For your TANDY Color Computer

Registered by Australian Post — Publication No. QBG 4009

NZ $3.95 PNG K5.95

$4.95

AUSTRALIAN RAINBOW

December, 1986

No.66

MERRY CHRISTMAS

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

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

RAINBOW Info

How To Read Rainbow

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

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

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

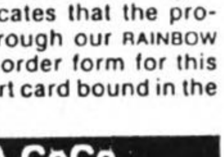
What's A CoCo

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

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

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

The Rainbow Check Plus



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

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

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

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

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

```
10 CLS:X=256*PEEK(35)+178
20 CLEAR 25,X-1
30 X=256*PEEK (35)+178
40 FOR Z=X TO X+77
50 READ Y:W=W+Y:PRINT Z,Y;W
60 POKE Z,Y:NEXT
70 IFW=7985THEN80ELSEPRINT
   "DATA ERROR":STOP
80 EXEC X:END
90 DATA 182, 1, 106, 167, 140, 60, 134
100 DATA 126, 183, 1, 106, 190, 1, 107
110 DATA 175, 140, 50, 48, 140, 4, 191
120 DATA 1, 107, 57, 129, 10, 38, 38
130 DATA 52, 22, 79, 158, 25, 230, 129
140 DATA 39, 12, 171, 128, 171, 128
150 DATA 230, 132, 38, 250, 48, 1, 32
160 DATA 240, 183, 2, 222, 48, 140, 14
170 DATA 159, 166, 166, 132, 26, 254
180 DATA 189, 173, 198, 53, 22, 126, 0
190 DATA 0, 135, 255, 134, 40, 55
200 DATA 51, 52, 41, 0
```

Using Machine Language

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

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

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

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

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

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

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

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

The Crew

Founder Greg Wilson
Publishers Graham & Annette Morphett
Managing Editor Graham Morphett
Accounts Annette Morphett
Assistant Editor 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,
Alex Hartmann, Michael Horn,
Darcy O'Toole, Martha Gritwhistle,
Geoff Fiala, John Redmond,
Sonya Young 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.

Merry Christmas from us!

Julie

Annette & Katie

Paul

Jim, Kate & Sheryl

Alex.

"The House!"

Sonya

Jim R

PRINT#-2

GEE! TIME FLIES WHEN you're having fun! Before you know it, the end of the year is here again! Some of you mightn't notice this, and then again, some others do. To some it seems we never take any holidays. (Or hardly ever.) Well, we do. We just don't announce it.

The truth of the matter is that our holidays are at the end of December, and they go for two weeks. Therefore if you decide to drop in/drop a line/drop in a letter etc, between the 19th and the 5th January, we won't be there to answer it.

And now for the news.

Geoff Fiala has made an advanced CoCoConnection. The newer model has an advantage over the older model in that it has two extra buffers (two extra PIA chips, not two extra outside input/output ports.). The price is still the same ($206) and if you read through this month's Rainbow, you'll find an article on applications for the CoCoConnection.

. Something for the CoCo III owners: we have made a disk of programs exclusively for the Colour Computer III. Some of the programs include converted graphics pictures, games from previous CoCoOz programs, a few programs showing the uses of the PALETTE command, a very basic game starter program for you to develop into your own game. This game starter program includes two ships (one at the top and the other at the bottom), score update and number of men left.

User Groups

I'm planning to see some of the User Groups in the general Brisbane area. So far I've only seen two groups in action, but on a spectrum from beginner to experienced, they're on opposite ends!

The first User Group is led by Colin North and his group is for the beginner and novice. They meet at his house at 1pm every fourth Sunday of the month. They have no formal planning of what they do, they just do whatever comes up.

For example, one user there asked about the DATA statement; ,what is it's purpose, where do the numbers come from in a typical data statement, etc. Colin can answer all this in an easy to understand manner, with plenty of programs to boot.

If you are interested in going to his group, ring Colin on (07) 824 2128, or, if you want, drop in on Colin at his home address, which is 4 Denice St, Birkdale.

The second User Group is led by Jack Fricker. His group meets on the first Saturday of the month at 1pm. This group is more for the experienced user. Like Colin North's group, they don't plan their day; they do what they feel like at the present moment. When I was there we talked about the CoCo III and showed some of the CoCo's colour & graphic capabilities. I found out some fairly interesting information. Did you know you can exchange any one PMODE 3 colour for any other one of the other 60 colours? For example, if you don't like green in your games, you can change it to black. Or purple. Or grey. Whatever you fancy. If you're interested, read 'Frickers Folleys' in this month's magazine.

Alternatively, if you want to give Jack a buzz, ring him on (07) 262 8869 and say "Alex suggested I call you!". Then again, if you want to drop in, his address is 63 Noble St, Clayfield, Qld.

CoCo III Programming Competition

Well, the programming competition for the best program for the Colour Computer III is going along well. I've had a few entries, some quite interesting, others even more. All I can say is keep 'em coming! Remember! There are lots of prizes to be won, like a pair of joysticks, a touch pad, a box of disks, and more!

Please note that the contest was incorrectly presented in Australian CoCo last month. There is only one prize and the winner will receive the joysticks, touch pad & disks.

Viatel: What's New

If you read "What's New" on page 4 of the Viatel Main Index, you'll notice there was an ad for MicroNews. What is MicroNews? MicroNews is Paris Radio's newest feature!

MicroNews is a weekly-updated board to tell the user what's new in the computing world. It is aimed at the CoCo, Amiga, Atari (520 & 1040), IBM PC AT/XT, 68000 based computers, and others!

It reports on new software, hardware, books and more. If you want to have a look, see Paris Radio's main index on node *64268# and find MicroNews. There are about 10-11 pages of information to read, and each page is 15c each.

Personally I think the idea of having a "What's New" in computers not such a bad idea. I had a look at MicroNews just the other day. Some of the information on there is really interesting! Sorry I can't tell you what it was.

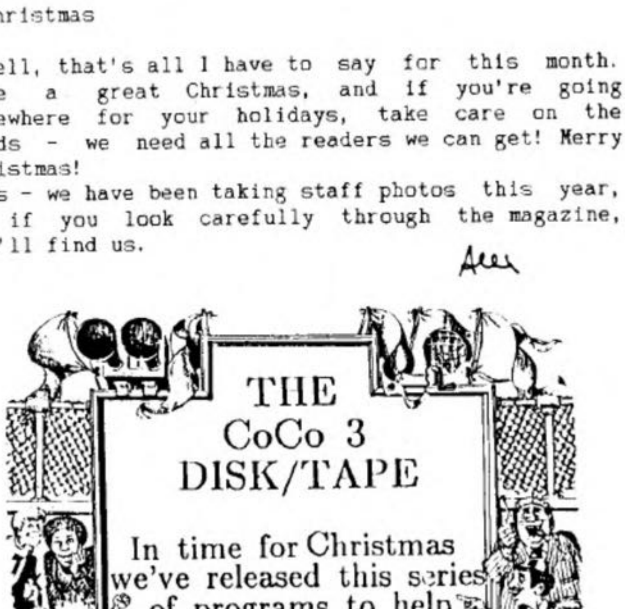
The Bulletin Boards. As most of the Viatellians will know, I'm your honorable Com Jok every Monday and Tuesday nights (if you're interested). So if you're ever on, message me and say hello!

Christmas

Well, that's all I have to say for this month. Have a great Christmas, and if you're going somewhere for your holidays, take care on the roads - we need all the readers we can get! Merry Christmas!

Ps - we have been taking staff photos this year, so if you look carefully through the magazine, you'll find us.

Alex



LETTERS

Dear Graham,

I dips my lid to M & J Spiller for their fine effort in CoCo Synthesiser (Rainbow July 86).

I've barely started to explore all the possibilities, but having messed around with music for some years my ear was slightly disconcerted by the tuning, particularly in the top octave.

So I wrote a little program that instantly turned me into an electronic piano tutor and after a bit of fiddling I came up with the enclosed amendments to the key-table frequency values.

As suggested by the authors, simply poke the new values, then you can save the re-tuned version for subsequent loading.

It may not be perfect as I also tuned my ear, but if you check the various notes against their corresponding octaves the results sound quite good.

Incidentally, that is a standard technique for tuning real stringed instruments. Anyway, I hope my labours can be of benefit to someone.

On another sour note, if you'll forgive the pun, you tantalisingly included in the same issue an article on MUSIC+.

This is a program I happily spent five hours typing in from the Rainbow July 1984 and is absolutely brilliant. The only short-coming was lack of facilities to modifiy the sound envelope which apparently has now been fixed.

POKES to re-tune "Piano"

Addr	Dec	Char
3187	30	B
318A	36	C
318D	39	D
3199	28	H
319E	25	J
319F	25	
31A5	21	L
31A7	24	M
31A8	23	
31AA	27	N
31AB	27	
31B3	169	Q
31B4	168	
31B9	44	S
31BA	44	
31C8	41	X
31C9	41	
31CB	100	Y
31CC	100	
31CE	46	Z
31CF	46	
3202	18	+
3204	22	<
3205	22	
320B	19	>

Bernard Besparis

*

Dear Graham,

I have seen reviews of Microcom software in your magazine. Could you please tell me the address please.

Blair Bryant.

Blair,

Write to:

Microcom Software,
P.O. Box 214,
Fairport NY 14450, USA.

Dear Graham,

I'd thought I'd write to you this time instead of via the telephone. First up, do you know of anyone who has successfully RUN Geoff Mackie's "Directory PRNT" using a DMP-110 printer?

I typed in the program initially and got a ?BS ERROR. I wrote to Geoff who told me to change Line 10 which I did.

I have since changed the control codes for my printer to the following as was required by the program:

Double width on:	27-16
Double width off:	27-19
Feedform:	12
Double strike:	27-18

This seemed to work, since the prompts followed right through until I was asked for "Disk Title".

I type this in-

AMER RAINBOW PROGRAMS AUG

and that was it - nothing. It would appear that the computer is not telling the printer to print, or alternatively the printer is not accepting the instruction to print the directory.

Any suggestions?

John Grigsby

Dear John,

This has been a saga for you I realise.

Perhaps one of our readers who has perfected this conversion can assist.

Graham.

*

Dear Graham,

I have a problem that you might be able to solve. When I use "LLIST" for my printer (Silver Reed EB50), it takes the lines without carriage return, so I end up with a single line with all the program on that line.

Are there any POKES or PEEKS to solve this problem?

Yours faithfully,
John Mikulski
NSW

John,

That's one we'll ask our readers about!

Graham

*

Dear Graham,

Re 'VEFAX PROGRAM'

Referring to Ross Platt's letter in the October issue I too found that the values for F1, F2 and F3 in lines 30,40 and 50 needed changing considerably from the values suggested in the program as published (P35 April Issue).

After a good deal of trial and error I found that the best values were 22, 483 and 712. All values are quite critical to get near perfect verticle picture alignment.

I have received FAX pictures of quite good quality from a number of stations - approximate frequencies are 22.53, 19.27, 18.94, 18.13, 17.06, 17.07, 16.23, 16.05, 16.06, 12.67, 8.07, 8.14, 5.80 and 5.09 Mhz.

I think the stations move slightly in frequency on a day to day basis, but with a little practice it is easy to pick the sound of FAX signals - a kind of whirr/whirr at line frequency making them quite destructive from other radio signals on the air.

Regards,
Peter Sweetingham
TAS

The following Questions & Answers were taken from the Tandy & OS9 Bulletin Boards in Goldlink this month.

*

Is there an expected date for release of CoCotex3?

If you know when please share it with me!!

The Frazz

Dear Frazz,

See the following letter.

*

Does anybody know when the CoCo3 version of CoCoTex will be out?

Richard

Dear Richard,

CoCotex3 will be released by the end of this month. CoCoTex3 upgrades will cost $20 plus the return of your old CoCoTex disk or tape. If you have a tape master and would like a disk master in return it wicost $25. With the new CoCoTex 3 package yu wil receive both a CoCo2 and a CoCo3 version plus a new handbook.

All restrictions to the setup of the users ID were removed in version 2.1. The ID can be set to any character up to a length o 16. Version 2.1 also provides an extended printer baud rate from 300 - 9600.

CoCoTex Kid

*

Any 'hidden' extras that the CoCo 3 has that we don't know about?

Answer:

The CoCo3 has a 64 Character mode but isn't supported by BASIC. Other modes the GIME support include:

640 by 192 4 AND 2 COLOR
512 by 200 4 AND 2 COLOR
320 by 210 16 4 AND 2 COLOR
256 by 210 16, 4 AND 2 COLOR
160 by 225 16 COLOR

CoCo3 64 Charcter mode POKEs. The width 80 command wil be replaced with 'Width64' command with the following POKE's.

```
POKE 63052,64: POKE 63105,64:
POKE 63112,44: POKE 63113,00:
POKE 63601,128: POKE 63605,43:
POKE 63606,128
```

*

To all serious OS9 users and others We are here to help with your queries re OS9 in general. If you need help or would like to offer some - HELP that is please leave a message on this BBS so we can reply.

Aust. OS9 User Group.

P.S. join our group for monthly OS9 Newsletter.

AUSTRALIAN PRODUCT REVIEWS

SPECTRUM DOS 1.0

Technical Information dept.
Title: Spectrum DOS 1.0
Classification: Utility Software
Cost: $???
Supplier: Computer Wizardry
Requirements: 64K ECB, Disk Drive

What it is dept.
Spectrum DOS is an advancement over the shortcomings of the CoCo 1 and 2. It fixes a number routines inside the CoCo's ROMs, adds 24 new commands and gives the user a choice of 32, 51 and 64 characters per line.

New features include:

* 35/40/80 track drives can be added to the system.
* Auto disk search: all drives are search for a program)
* One button text screen dump
* One button loading of a BASIC program ("BOOT/BAS")
* Lower case readable: all commands can be in lowercase.
* Auto key repeat
* New cursor: can be any printable character
* New Prompt: can be anything you want
* Reset protected.

Commands that have been fixed include:

* DIR - prints two directory entries at a time side by side and the number of free granules.
* DSKINI - lets you know what it is doing.

The 24 new commands cover such things like:

* DOS: boot OS9
* ERROR: 'on error goto' routine
* FLEX: boot flex
* RUNM: 'loadm & exec' ML files
* PPEEK & PPOKE: reads/writes 16 bit values
* AUTO (auto line numbering)
* INVERT: After this command text to the screen is inverted
* NORMAL: Rreturns the text screen to normal.
* WAIT: memory saving timer routine
* LMOVE: moves and destroys line numbers
* RATE: sets new stepping rate for drives
* TRACKS: sets number of tracks per drive
* HELP: lists all new commands
* HDIR: prints hard copy of DIR
* HIRES: selects 32, 51 & 64 column screen
* OLD: in a case of a NEW
* FKEY: Define up to 9 programmable keys.
* LCOPY: Duplicate a BASIC line
* BREAK: Disables BREAK key
* MEMO: a full text screen editor & screen dump
* FLIP: invert entire text screen.
* EXIT: gets out of hi-res screen
* ECHO: 'echo's everything out to the printer

Problems dept:
There are no 'real' software problems with this package; it is well done. Mind you, the following aren't problems, just something to be careful of.

* At present, you can have up to 4 drives, each capable of having 35 tracks. But the number of drives are lowered the more tracks you have for each drive. Viz, if you format 40 tracks, you can only have 3 drives to your system and if you format 80 tracks you can only have 2 drives.
* Not many programs are written in 51 or 64 column format; but nevertheless handy to have for presentation, etc.
* You can't undo the BREAK command.

General Comments:
If you have a CoCo 3, don't worry about buying this package; it has most of it built-in or already or improved. If you don't, and you like some of the things the new CoCo has, then I would consider getting this. It is very user friendly and by typing HELP will remind you of the commands available. Some of the commands in there I found very useful, some not at all useful, and others great potential.

Rating dept:
Workability: 7/10
Usefulness: 7/10
Overall: 7/10

by Alex

A DOS

Technical Information dept.
Title: ADOS
Classification: Utility Software
Cost: $???
Supplier: Computer Wizardry
Requirements: 64K ECB, Disk Drive

What it is dept.
ADOS is an enhancement over the present Disk Extended Color Basic. ADOS can be loaded in from disk or can be burned into a ROM. If you choose to burn it into ROM, you will find that there exists no incompatability with any software prior to and including RS DOS 1.1.

If you don't decide to burn it into an EPROM, then expect some incompatability to exist.

Some features include:

* Modified DIR commands
* Modified COPY & RENAME commands
* The 'head banger' and 'head settle' bugs have been fixed.
* Printer baud rate is set
* Verify is automatically on.
* New commands:
  - runm: Loads & executes a ML file
  - dos: boots OS-9
  - auto: Auto line numbering
  - ram: Puts you into 64K mode
  - rom: Gets you out of the 64K mode
  - prt on/off: Sends anything to the printer as well as the screen
  - scan: Returns the start, end, & exec addresses of an ML file
  - peep: Allows the view of a section of memory.
  - mon: Like Edtasm+'s ZBUG monitor; allows a view

at memory.
- disable: Turns off all ADOS's commands.

* Control Key abbreviations
The down arrow key while being held down while another key is pressed functions as a control key. They may be modified as well.

* The ability to customize ADOS completely to your tastes.

Problems dept:

There aren't any real problems with this package - it just depends on weather you need something that describes the above. If you do, and you like what you read above, then by all means, go out and get it! Some of the commands in there I find attractive; some commands in there are a REAL keystroke saver, ie the scan command and the modified copy and rename commands!

General Comments:

What I said above basically says it all. I find it interesting and attractive, but for my purposes, not necessary.

Rating dept:

Workability:	7/10
Usefulness:	6/10
Overall:	6/10

by Alex

A RECIPE TO FIX COCO FRIED CHIPS

IT IS VERY rare that controllers just spontaneously cease to work. In nearly all cases the reason is because the user has plugged, unplugged or wriggled the controller in the computer or multipack port socket while the power was on. What usually happens is that the positive and negative 12-volt lines (on old CoCo 1's and on Multipacks) contact the adjacent NMI and Halt line pins. Often this also burns out the CPU (the 6809) and/or the SAM (6883, also given as 74LS783 or 74LS785) in the computer itself. Had Tandy bothered, for about $1.50 worth of zener doides and SCR's it could have fully protected the computer from such abuse. But in all revisions of the CoCo circuit board so far, it has not intrduced such protective circuitry.

Fixing the burned out disk controller usually entails replacing the burned out chips. This is facilitated by knowing what chips are likely to get burned out and by having a full set of spares.

Usually, on the newer controllers only the main disk controller chip (1793-002, MB8877a or 1773), the disk ROM and the write protect precompenstation chip (if any) is socketted. Most of the small scale logic chips are soldered directly on th board. You should be reasonably adept in desoldering intergrated curcuits. You should on hand a full assortment of all chips found in your particular controller. A spare controller of the same make and model will give you access to the bigger socketted chips and the smaller chips are usually available at general IC supply houses.

On most models of CoCo disk controller, the 7416 (open collect buffer) is quite vunerable to damage from the -12 volt supply. In three CoCo 2 controllers I have fixed, both 7416's had to be replaced. These are U3 and U8 on the older type CoCo 2 controller with a 40-pin controller chip, and U8 and U6 on the newer Tandy controller that uses the 28-pin 1773 disk controller chip. The 74LS221 (one shot delay timer) seems to often burn out as well. Occasionally the main disk chip controller chip does, too.

On the older CoCo 1 controller from Tandy, the 74LS02 and the 74LS04 chips (U9 and U5 on that card) have a track record of blowing - sometimes spontaneously. Be sure to look for blowen 7416's and 74LS221's on that model. You should also have the associated Tandy technical service manual and a frequency counter. The potentiometers may need adjustment so you'll need the frequency counter to check for proper setting.

On third-party model controllers, the curcuitry is often similar to Tandy's, and thus the vunerable chips are likely to be the same. In the case of the old J&M controller, the disk chip is available only from J&M itself.

Oddly enough, the ROM chip on these controllers seldom seems to be affected. Indeed, once in the course of repairing a contrller I plugged in a ROM upside down and turned on the power. After realising my blunder, I turned off the power, inverted the ROM and tried again. Much to my amazement, the ROM functioned just fine.

In addition to these general tips, the serious trouble-shooter will want schematic diarams of the unit to be repaired. Tandy and HDS both supply such technical information on request and for a reasonable sum. J&M in the past was reluctant to release schematics, but may be changing its policies. DISTO agrees with the idea of releasing schematics of its products to the public, but to date has not made such information available. It is my impression that if enough customers insist on such information before buying any products from J&M or DISTO, both of these companies will quickly supply it.

BDOS BUG

SOME MONTHS ago I purchased a BDOS controller and have had some problems using the cassette and tape files while BDOS was place. Specifically, SKIPF causes the computer to hang up after it has SKIPFed, and cassette data files cause problems, especially if there is an I/O error.

I took the time to trace this bug, and there is a simple solution. If you own BDOS and you wish to use the cassette, type and enter the following poke (this is a two byte poke using the PPOKE function).

```
PPOKE &H1A4,&H8304
```

The cost is that BDOS's ability to accept BASIC commands in lower case will not work. If this function is of little importance to you, you could include this poke in your BOOT/SYS.

The explination is that BDOS uses the cassette input buffer to convert lower case commands into upper case. The cassette buffer is used by SKIPF and INPUT#-1.

Those with a disassembler can follow the bug:

CRUNCH BASIC HOOK $1A3 does a JMP to $DF14 (the usual hookis a JMP to $8304). The routine at $DF14 converts lower case letters to upper case, but when finished converting, it branches to $200 (in the cassette buffer) where there is a JMP $8304. This seems a remarkably stupid piece of programming, because it is the location of this single JMP which causes the bug.

The bug could be overcome in RAM mode by changing the instructon at $DF18 so it reads:

LBEQ $8304 (it now reads LBEQ $200).

I have no idea if the bug occurs with the BDOS you can buy on disk. If it does a PPOKE into $DF1A will do the job without losing the lower case ability (sorry, I haven't calculated what it should be).

I have had no other problems with BDOS and find some if it's features, particularly PPOKE and PPEEK commands very useful.

by John Carmichael

Hardware Review

PIXY 3, SILVER REED, EPSON : A4 PLOTTERS AVAILABLE

I HAVE NOT UNDERTAKEN any extensive survey of what plotters are available. I started out with the intention of getting a Silver Reed, but Big-W didn't have any when I went there. So I rang a couple of numbers in the Yellow Pages and found out a bit about the Epson and the Pixy. I brought the Pixy and stopped looking, but can provide a brief outline of these three.

PIXY 3

Is made by Tally, who changed their Australian agents earlier this year. The new agents are not carrying this plotter, and the old agents are selling out their stock. A bit of a risk for future support but normal flat bed plotters are not usually available at under $400. There may still be a few left.

It is a conventional flat bed plotter with a serial interface. It works with the same printer cable I use for a Brothers EP-44. Pin 20 on the RS-232 to pin 2 on the CoCo.

It can select one of 3 pens from a rack, and the other 2 can be changed while it is drawing with the third pen.

This plotter does not have a printer mode for direct use as a printer, but does provide the ability to print standard ASCII characters in any size, in 4 directions. I expect it should be fairly easy to write a routine for the CoCo to use it as a normal printer, but haven't actually done so yet. A variation of PSKIP should be sufficient.

SILVER REED

Silver Reed make a typewriter/printer/plotter whicl is a very versitile little machine. It has a 4 color pen, printing arrangement similar to the CGP-115, but in A4 size. The machine can be used as a stand alone typewriter/plotter with its own memory. Or it can be connected to a computer as a normal printer/plotter in much the same way as the CGP-115.

The problem with it is that the computer connection is Centronics, so you need a serial to parallel converter. These are available for the CoCo at a reasonable cost.

The Silver Reed is available from Big-W for $349. Silver Reed themselves quoted $449. Allowing an extra $90 for a serial to parallel converter, the total cost is still considerably less then any color dot matrix printer.

As noted, I intended to get one of these, but Big-W had none there when I went out there.

EPSON

The Epson plotter is available for around the same price as their cheapest color dot matrix printer ($850), so there is no price advantage with it over a dot matrix. Its a question of comparative quality and flexibility.

It is a neat little machine, has 4 pens available at any one time, and offers 10 colored pens, ball point or felt tip.

It is a neat little machine, has 4 pens available at any one time, and offers 10 colored pens, ball point or felt tip.

It is not a flat bed plotter, the Y axis is drawn by moving the paper. It has a standard printer mode to operate as a normal printer.

RUNNING COSTS

At this stage, I have no firm indication of the running costs of the Pixy. The pens are around $5 each, and I have not yet measured what I get from each one.

For color prints (and normal printing) there is little doubt that the normal dot matrix printer ribbon gives the lowest running costs. They have a long life and can be used well past their best quality with acceptable results for draft printing.

Other color printers are available which use a color wax ribbon and a thermal type print mechanism. Similar to the Brothers EP-44 and the Tandy portable, but with a multi colored ribbon. These are cheaper to buy then the normal dot matrix color printers, but the running costs for the ribbons is very high. The quality of print from the first time through the ribbon is quite good, but the second time through the ribbon, the print can be quite poor, even for draft use.

The third type of color printers are the ink-jets. These are more expensive printers and have high running costs for the ink.

I would expect the running costs for pens in a plotter to be somewhere between that for a normal dot matrix printer and the wax ribbon ink jet range. I feel the quality of the product from a plotter makes them worth it. As must be obvious, I am not at all impressed with the quality of color dumps from dot matrix printers.

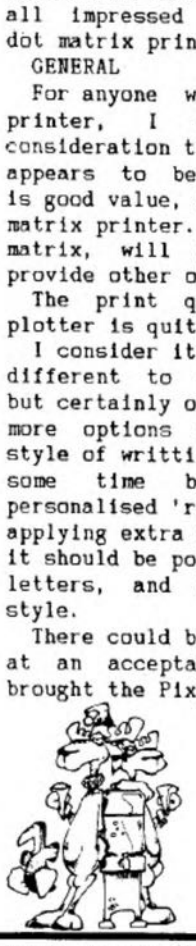
GENERAL

For anyone who might be thinking of a color printer, I would suggest they give serious consideration to a plotter instead. The Pixy 3 appears to be a one off thing, but the Silver Reed is good value, and a lot cheaper than any color dot matrix printer. The Epson at the same price as a dot matrix, will give a better quality result and provide other options as well

The print quality for normal printing from a plotter is quite good.

I consider it to be letter quality. A little different to typewriters and daisy wheel printers, but certainly of equal quality. It also provides more options for establishing your own personal style of writting (printing). There was an example some time back in Rainbow (June 85) of a personalised 'running' writting for the CGP. By applying extra program logic to letter combinations, it should be possible to get a smoother joining of letters, and develop your own 'printed' writting style.

There could be other A4 sized plotters out there at an acceptable price. I stoped looking when I brought the Pixy.



by George McLintock

Hardware Review

TELEMAT
ANTI–GLARE SPRAY

Technical Information dept.
Classification: "Hardware Modification"
Cost: Depends on the screen type.
$42 for computer and monitor screens
$35 for a standard & domestic TV
Supplier: Telemat, Pty Ltd
3 Fortril 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 advantages 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 problems. Clean it with Anti-Static polish once a week. Each can of antistatic costs $12.

5. Ok, so we let you apply the spray. What if we don't like it?

You can take it off quickly and easily, anytime.

"Problems" dept:

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

by Alex

USA PRODUCT REVIEWS

Software Review

CoCo-Util II: An Improved Way to Transfer Data

There is good news for those of you who need to transfer data between the Color Computer and an MS-DOS machine. Mark Data has released a sequel to its powerful programming utility, *CoCo-Util.*

CoCo-Util II provides the capability to migrate disk files from one system to the other. It requires an IBM/PC or compatible, 128K RAM, two floppy disk drives or a floppy and a hard drive and PC/MS-DOS Version 2.00 or higher.

As you can see from the requirements, you use *CoCo-Util* on an MS-DOS machine. So if you have an MS-DOS machine at work and a CoCo at home, you can create data files on your CoCo at home and transfer them to the MS-DOS format at work. When *CoCo-Util* is loaded into the MS-DOS machine, it adjusts itself for the type of video system and the color or graphics card that is active.

The new version has many enhancements. One is the improved use of colors and the screen layout, which includes an option to change the color scheme. If you want the change to be permanent, you can create a configure file that loads each time you call up *CoCo-Util.*

The *CoCo-Util* screen was designed to display as much information as possible without confusion. The layout includes seven areas. The first area, the Dir Info Box, gives information about the current directory that is loaded. This includes the free space left on the drive, and if it is MS-DOS, it displays the pathname to the current directory.

The Date Box displays the day of the week and the system date, along with the DOS version in use. The Files Info Box displays the number of files loaded and the maximum number of files that *CoCo-Util* has room to store in memory at one time. This number is dependent on the amount of RAM. A minimum system of 128K should show a maximum number of between 800 and 900 files. A full 640K has room for 9,999 files.

The Drive Assignment Box displays which drive is assigned to be a CoCo drive and which is the MS-DOS drive. This box is very important when you plan to format a CoCo disk. If you pick the wrong drive, all the data on the disk will be lost. The File Display Box is the largest area. This is where the filenames of the current directory are displayed. The display can show up to three columns of 15 (45 filenames at one time). If there are more than 45 files, you can page up or down through the files. You can also have the displayed files sorted. You have several sort sequence options.

The next area is the Message Line Area. This is the bottom line of the screen and is used to display messages from the program and error messages. The last area is the Menu Box, which is used to display the options available. Since most of the functions are menu driven, another level of options is displayed when many of the options are selected. When *CoCo-Util* first initializes, the primary menu is displayed. The primary menu consists of the following options; Copy, Dir, Erase, Format, Insert, Options, Print, Remove, Shell and View.

The Copy option brings up another menu that allows you to copy between MS-DOS and CoCo files. You can use the arrow keys to mark the files you want copied or you can use the wild card feature. The asterisk (*) and the question mark (?) are used the same way as in MS-DOS. These files will probably be ASCII files, but *CoCo-Util* also transfers binary files. This can be either an M/L program or a binary basic or data file. While a binary file may not run on a different machine, you can modify it in a word processor and transfer it back, or possibly use it for transferring over a modem. In any case, it will transfer it; it is up to you how you will use the file.

The Dir and Erase options are self-explanatory. The Format option allows you to format a CoCo compatible disk. I do have a suggestion here. The format is only 35 tracks, and there are many people who use 40 tracks. I would like to see an option to allow *CoCo-Util* to format either 35 or 40 tracks.

The Insert option is used to insert a line feed following each carriage return. Normally, CoCo text files have lines that are terminated with carriage returns only. In the MS-DOS world, a line feed is used, too. This option should only be used on ASCII text files because if it is used on a binary file, the file will be unusable.

The Remove option is the reverse of Insert, and again should only be used on ASCII text files. If Insert is used on a binary file, you may try to Remove them, but do not be too hopeful.

The Print option allows you to dump any file to your printer using either an ASCII or a Dump format. This is good for printing an ASCII file or for printing out ASCII text in a binary file. You Adventure gamers may find a use for that. The Dump format prints a file in ASCII and Hex, and prints the relative displacement of each byte. In both formats, non-printable characters are replaced with periods.

The Shell option allows you to temporarily leave *CoCo-Util* and perform something in MS-DOS. The View option is similar to Print, but the file goes to your screen.

I was very impressed with this new version of *CoCo-Util*. The screen layout and use of colors add a professional touch. I would like to see the 40/35 track option for the CoCo disk, but overall I liked the program. I recommend it highly to anyone who needs the capability of transferring data between computers. It is very useful if you have text files to transfer. It can also be used to transfer programs written in CoCo Extended BASIC to an MS-DOS machine, modify them, then compile them to run under MS-DOS. If you already have the original *CoCo-Util*, you can get an upgrade to *CoCo-Util II* for $12.95 including shipping and handling.

(Mark Data Products, 24001 Alicia Parkway, #207, Mission Viejo, CA 92691, $39.95)

— Dale Shell

Software Review

Ultra Telepatch Improves the 'Perfect' Word Processor

Telewriter-64 is, in my opinion, the most popular word processor available for the Color Computer. I base that on the number of program submission articles written with *TW64* that are sent to the RAINBOW by the CoCo community. I use *TW64* on almost a daily basis and have been delighted with its service.

Many reviews have appeared in the pages of the RAINBOW describing *Telewriter* and many of its enhancements. I recently reviewed *Telepatch II*, written by Bob van der Poel, and was pleased with the extra features it afforded. I honestly thought *Telewriter-64* had been perfected, but boy was I wrong!

Just about the time we think something is perfect, someone comes along and improves it. This is the case with the latest endeavor by Mr. van der Poel, called *Ultra Telepatch* Version 3.0. A lot has already been said about *Telewriter*, so I will just point out the main improvements of this latest effort.

Disk I/O — *Telepatch II* gave the option of calling the I/O from disk or memory. The reason for the option was that buffer space was used if you chose to use the memory option. In the *Ultra* version, the disk I/O is stored in memory with no loss of buffer space. This is the best of both worlds — speed *and* efficiency.

Word Delete — The original *TW64* features a character delete, but most of us think in words, not characters. The *Ultra* version features word delete. Just move the cursor to the blank space in front of the object word and press CLEAR and 'Y' (for yank). All characters in the word will be deleted to the next space or carriage return.

Insert Space — Pressing CLEAR and the space bar will now insert a space at the cursor position.

Braces — The special characters { and } can be generated by pressing CLEAR-'H' and CLEAR-'J' combinations.

Find and Global Replace — This enables searches and replacements of control characters as well as normal text characters.

Queuing Files — Now you can use a period (.) as well as a slash (/) for filenames.

The *Ultra Telepatch* disk also contains some new files of special interest that can be merged with the T/BAS boot program:

2COLDIR/BAS — Provides a two-column, on-screen directory format.

TODISK — Forces *TW64* to display the disk menu on start up. This is very helpful if, for example, you need to load in an initialization file.

TPRINT — Provides automatic printing of multiple copies of your text files. No longer do you need to sit in front of your CoCo pressing 'P' for each copy.

The boot program is fully remarked so that the program can be tailored to most individual needs. Here you can select your disk drive stepping rate, turn on and off key clicks and all the other useful features added in the earlier enhancement versions. Extra lines have also been added for the user's special requirements, such as defeating reset protection with `POKE 113,3` or maybe adding special printer control codes.

One other thing. Remember how you used to have to press CLEAR-UP-ARROW after reading in a file, so it would unfold on the screen? No more. Now the text unfolds automatically as soon as the file has read into the buffer.

I discovered one potential problem quite by accident. After a lot of frustrated searching for both hardware and software problems, I discovered that if either joystick is plugged in and is approximately in the 10 to 11 o'clock position, the computer appears to lock up while attempting to run the boot or patcher programs. This is not a flaw in the program, but apparently the `USR(X)` calls in these

continued on page 42

Portraits by BASIC

by Ann B. Mayeux

WHOSE face will you draw? With Draw Face, you can draw your friends, a clown, a vampire, a baby and any other face you can imagine. It's easy.

Using single-key selections, choose face shape, ears, hairstyle, eyes, eyebrows, mouth, nose, accessories and other features such as beard, glasses, or vampire cape. To make a selection, press the letter for the option wanted.

To help you, a box in the upper right-hand corner tells which features you need to choose. If you cannot remember the options, pressing the (/) brings up a list of the letters to use and a brief description (see figure).

After your picture is the way you want it, press the '@' key and type in a name across the bottom of the screen. If the name is five or six characters long it will be centered. Push the left arrow-key to allow for more letters. A tone sounds when the left margin is reached. Pressing the '@' key erases the name.

You can clear the screen at any time, and once ears are selected, the features can be changed by pushing the up-arrow. Repeatedly pressing the up-arrow erases previous features in turn, except the face shape. Accessories cannot be erased, but if you have started accessories and decide you want to change a feature, the up-arrow takes you back through. After typing the name, the up-arrow takes you back to accessories, although the box in the corner does not reappear.

After enjoying your picture, clear the screen and begin again.

Menu Options

Shape
B - Baby
C - Cadaverous
L - Long
O - Oval
R - Round
S - Square

Ears
B - Big
E - Normal ears
L - Little
N - No ears
S - Pointy

Hair
A - Afro
B - Boy's
C - Curly
D - Dutch boy
H - Hair
I - Curly infant
L - Long
M - Middle part
N - No hair
O - Old fringe
P - Pony tail
R - Red fringe
S - Girl's short
W - Widow's peak

Eyes
B - Big
C - Closed
E - Open eyes
I - Little
L - Eyes with lashes
M - Mad
O - Sleepy open
S - Surprised
T - Tired

Eyebrows
A - Arched
B - Bushy
C - Clown
I - Infant
M - Mad
N - None
Q - Quizzical
S - Surprised
T - Tilted

Mouth
B - Big
C - Clown
H - Happy
I - Infant rosebud
L - Lipstick
M - Straight mouth
O - Open
S - Sad
T - Teeth
V - Vampire

Nose
C - Clown
I - Infant
N - Narrow
P - Pug
S - Straight
W - Wide

Accessories
B - Beard
C - Cheeks
E - Earrings
F - Frown
G - Glasses
H - Hat
M - Mustache
N - Neckline
R - Hair bow
S - Shirt
T - Bow tie
V - Vampire's cape

150	161	1640	153
260	169	1810	94
420	49	1960	162
560	189	2120	213
770	64	2260	105
930	154	2460	227
1040	103	2570	142
1130	122	2690	30
1340	213	2770	194
1450	191	END	14

The listing: DRAWFACE

```
10 ' DRAW A FACE
20 ' BY ANN B. MAYEUX
30 ' KEY WEST, FL
40 Z$="          ":U$="     <^> TO CH
ANGE ":Y$="      <CLEAR> TO CLEAR
SCREEN"
50 '*SHAPE
60 PMODE3,1:PCLS:SCREEN1,1:MG$="
BM5,0R245D190L245U190R200D40R45"
:DRAWMG$
70 DRAW"BM20,20U10R5F2D6G2L5BR13
U10R6F2DG2L6R5F3D2BR6U6E4F4DNL8D
5BR7NU10E4NUF4NU10BR20U6E4F4DNL8
D5BR20U10NR7D5NR5D5BR13U6E4F4DNL
8D5BR15BU2G2L4H2U6E2R4NF2BR9NR7D
5NR5D5R7":GOTO90
80 LINE(209,3)-(248,38),PRESET,B
F:RETURN
90 DRAW"BM210,9R3U3L3U3R3;BR4D6U
3R4U3D6;BR4U4E2F2DNL4D3;BR4U6R4D
3L4D3;BR8U6R3BD3L3D3R3":CIRCLE(2
26,20),10,,.8,.41,.1:DRAW"BM218,
24F3D4F3R5E3U4E2U"
100 A$=INKEY$:IFA$=""THEN100 ELS
E LINE(10,10)-(200,30),PRESET,BF
110 IFA$="C"THENF=1:GOTO230
120 IFA$="O"THENF=2:GOTO270
130 IFA$="L"THENF=3:GOTO240
140 IFA$="R"THENF=4:GOTO250
150 IFA$="S"THENF=5:GOTO260
160 IFA$="B"THENF=6:GOTO220
170 IFA$="/"THEN200
180 GOTO100
190 ON F GOTO230,270,240,250,260
,220
200 CLS:SCREEN0,0:PRINT@74,"SHAP
E":PRINT:PRINTZ$+"<B> BABY":PRIN
TZ$+"<C> CADAVEROUS":PRINTZ$+"<L
> LONG":PRINTZ$+"<O> OVAL":PRINT
Z$+"<R> ROUND":PRINTZ$+"<S> SQUA
RE"
210 IFINKEY$=""THEN210 ELSESCREE
N1,1:GOTO100
220 CIRCLE(121,73),49,,.8,.42,.1
:CIRCLE(96,106),21,,1,.32,.61:CI
RCLE(146,106),21,,1,.89,.2:CIRCL
E(121,123),19,,1,.13,.37:DRAW"BM
90,124M110,135BR22M153,123":GOTO
290
230 CIRCLE(120,76),47,,1,.4,.12:
DRAW"BM87,102D20F20R28E20U20":GO
TO290
240 CIRCLE(120,85),43,,1.5,.95,.
6:CIRCLE(120,67),41,,1,.5,0:GOTO
290
250 CIRCLE(120,82),51,,1,.3,.2:C
IRCLE(122,119),20,,1,.1,.41:GOTO
290
260 CIRCLE(120,60),47,,.65,.48,.
02:CIRCLE(120,120),35,,.6,0,.5:D
RAW"BM167,62M154,120BL68M74,62":
GOTO290
270 CIRCLE(120,80),48,,1.1,.45,.
05:CIRCLE(120,119),30,,.9,.1,.4:
DRAW"BM167,94M143,136BL45M74,94"
280 '*EARS
290 GOSUB80:DRAW"C8BM215,17U6R4B
D3L4D3R4;BR4U6R3D3NL3D3;BR4U6R4D
3L4RF3;BR4R3U3L3U3R3BM232,27E2RF
2D3G2LH2BL11G2LH2U3E2RF2"
300 R$="R2E2R3F2D14G4L3H4":L$="G
4L3H4U14E2R3F2R2":B$="R2E3R4F5D1
4G4L7H6U2":I$="D2G6L7H4U14E5R5F3
R2"
310 S$="R5E10D25G8L4H8":P$="G8L4
H8U25F10R5":RB$="RE3R4F3D8G3L4H3
":LB$="LH3L4G3D8F3R4E3"
320 IFQ=1THENIFE=0THEN630ELSE ON
 E GOTO440,530,580,490
330 Q=0:A$=INKEY$
340 IFA$="E"THENE=1:GOTO440
350 IFA$="S"THENE=2:GOTO530
360 IFA$="B"THENE=3:GOTO580
370 IFA$="L"THENE=4:GOTO490
380 IFA$="N"THEN630
390 IFA$=CHR$(12)THENRUN
400 IFA$="/"THEN420
410 GOTO330
420 CLS:SCREEN0,0:PRINT@74,"EARS
":PRINT:PRINTZ$+"<B> BIG":PRINTZ
$+"<E> NORMAL":PRINTZ$+"<L> LITT
LE":PRINTZ$+"<N> NONE":PRINTZ$+"
<S> POINTY SPOCK":PRINT@448,Y$
430 IFINKEY$=""THEN430 ELSESCREE
N1,1:GOTO330
440 IFF=1ORF=6THENDRAW"BM164,83X
R$;BL83XL$;"
450 IFF=2THENDRAW"BM166,84XR$;BL
87XL$;"
460 IFF=3ORF=5THENDRAW"BM161,85X
R$;BL77XL$;"
470 IFF=4THENDRAW"BM169,82XR$;BL
93XL$;"
480 IFQ=2THEN290ELSE630
490 IFF=1ORF=2ORF=6THENDRAW"BM78
,87XLB$;BM163,87XRB$;"
500 IFF=3ORF=5THENDRAW"BM81,90XL
B$;BM159,90XRB$;"
510 IFF=4THENDRAW"BM75,90XLB$;BM
165,90XRB$;"
520 IFQ=2THEN290ELSE630
530 IFF=1ORF=5THENDRAW"BM162,86X
S$;BL73XP$;"
540 IFF=2ORF=6THENDRAW"BM165,84X
S$;BL79XP$;"
550 IFF=3THENDRAW"BM162,83XS$;BL
73XP$;"
560 IFF=4THENDRAW"BM169,80XS$;BL
87XP$;"
570 IFQ=2THEN290ELSE630
580 IFF=1ORF=6THENDRAW"BM164,83X
B$;BL80XI$;"
590 IFF=2THENDRAW"BM166,85XB$;BL
86XI$;"
600 IFF=3ORF=5THENDRAW"BM161,87X
B$;BL75XI$;"
610 IFF=4THENDRAW"BM169,82XB$;BL
90XI$;"
620 IFQ=2THEN290
630 IFF=4THEN640ELSEDRAW"BM149,1
27D20F20BL97E20U20":GOTO650
640 DRAW"BM150,121D15F20BL100E20
U15"
650 GOSUB80:DRAW"BM216,16U6D3R4U
3D6BR4U6R4D3NL4D3BR4NU6BR4U6R4D3
L4RF3;BM218,30U6E3R12F3D6L2U2H2L
10G2D2L2"
660 '*HAIR
670 A$=INKEY$
680 IFA$="A"THEN1020
690 IFA$="B"THEN940
700 IFA$="C"THEN1080
710 IFA$="D"THEN1130
720 IFA$="H"THEN1060
730 IFA$="I"THEN980
740 IFA$="L"THEN1040
750 IFA$="M"THEN1010
760 IFA$="N"THENB=1:GOTO1150
770 IFA$="O"THEN1110
780 IFA$="P"THEN920
790 IFA$="R"THEN1120
800 IFA$="S"THEN990
810 IFA$="W"THEN970
820 IFA$="/"THEN860
830 IFA$=CHR$(12)THENRUN
840 IFA$=CHR$(94)THEN890
850 GOTO670
860 CLS:SCREEN0,0:PRINT@74,"HAIR
":PRINT:PRINT"<A> AFRO        <L
> LONG        <B> BOYS        <M
> MIDDLE PART <C> CURLY       <N
> NONE        <D> DUTCH BOY   <O
> OLD FRINGE  <H> STRAIGHT    <P
> PONY TAIL"
870 PRINT"<I> INFANT       <R> RE
D FRINGE  <S> SHORT GIRLS <W> WI
DOW'S PEAK":PRINT@416,U$+"EARS":
PRINTY$
880 IFINKEY$=""THEN880ELSESCREEN
1,1:GOTO670
890 DRAW"C5M+0,0":Q=2:ON E GOTO4
40,530,580,490
900 GOTO330
910 IFINKEY$=""THEN910 ELSESCREE
N1,1:GOTO670
920 P$="E9R12F9D20F3E5D10G6L7H7U
13H4L8":O$="H9L12G9D20G3H5D10F6R
7E7U13E4R8"
930 IFF=3ORF=5THENDRAW"BM160,74X
P$;BL85BU10XO$;"ELSEDRAW"BM165,7
0XP$;BL95BU10XO$;"
940 CIRCLE(96,6),68,,1,.15,.25:D
RAW"BM163,83L6U6H5L3H10U8BD20BL4
5L6G3D6L6"
950 IFF=2ORF=4THENDRAW"BM169,83L
9BL80L9"
960 GOTO1150
970 DRAW"BM163,83L5U13H12L11G15H
15L11G12D13L5":GOTO950
980 HR$="R8F4D4G3R8E4U5H3":DRAW"
BM75,86F3R7E2U4GL3H3U3E7R3XHR$;X
HR$;XHR$;R9F5D10G2L3H2D4F3R5E6":
GOTO950
990 CIRCLE(90,94),10,,2,.2,.65:C
IRCLE(150,94),10,,2,.85,.4:DRAW"
BM70,108E9BR80F9"
1000 CIRCLE(80,108),13,,.9,0,.55
:CIRCLE(160,108),13,,.9,.95,.5:G
OTO940
1010 CIRCLE(92,34),33,,1,.1,.35:
CIRCLE(145,32),33,,1,.15,.4:GOTO
1150
1020 B=1:FORH=98TO146STEP6:CIRCL
E(H,41),8:CIRCLE(H,36),8:NEXTH:F
ORH=108TO132STEP6:CIRCLE(H,50),8
:NEXTH
1030 CIRCLE(88,43),8:CIRCLE(152,
43),8:CIRCLE(85,48),8:CIRCLE(156
,48),8:FORV=53TO70STEP5:CIRCLE(8
0,V),8:CIRCLE(162,V),8:NEXTV:GOT
O1150:REMCIRCLE(81,58),8:CIRCLE(
161,58),8:GOTO760
1040 CIRCLE(40,139),22,,1.2,0,.3
:CIRCLE(200,139),22,,1.2,.2,.5
1050 CIRCLE(120,105),66,,1.4,.45
,.07:DRAW"BM206,162G5L30BL99L30H
5":GOTO940
1060 CIRCLE(122,112),70,,.5,.65,
.85:IFF=1THENDRAW"BM166,84L6"
1070 GOTO950
1080 B=1:FORH=91TO152STEP8:FORV=
37TO54STEP8:CIRCLE(H,V),11:NEXTV
:NEXTH:FORH=100TO140STEP8:CIRCLE
(H,30),11,,1,.5,0:NEXTH
1090 CIRCLE(88,62),11:CIRCLE(151
,62),11:CIRCLE(96,62),8:CIRCLE(1
43,62),8
1100 FORV=50TO105STEP8:CIRCLE(79
,V),11:CIRCLE(160,V),10:NEXTV:FO
RV=69TO99STEP8:CIRCLE(72,V),12:C
IRCLE(169,V),12:NEXTV:GOTO1150
1110 B=1:DRAW"BM164,85L9H4U4E4R9
BL85R9F4D4G4L9":GOTO1150
1120 B=1:FORH=73TO90STEP5:FORV=7
0TO75STEP3:CIRCLE(H,V),7:CIRCLE(
240-H,V),7:NEXTV:NEXTH:CIRCLE(80
,65),7:CIRCLE(160,65),7:GOTO1150
1130 IFF=4THENDRAW"BM74,73F5R79E
9"ELSEDRAW"BM77,73F5R73E9"
1140 '*EYES
1150 GOSUB80:DRAW"BM215,16U6R3BD
3L3D3R3BR8U3NH3E3BR4D6R3BU3L3U3R
3BR7L3D3R3D3L3;BM216,29E3R3F3BR6
E3R3F3BL4C6UBL13DC8"
```

```
1160 A$=INKEY$
1170 IFA$="B"THEN1400
1180 IFA$="C"THEN1390
1190 IFA$="E"THEN1370
1200 IFA$="I"THEN1380
1210 IFA$="L"THEN1360
1220 IFA$="M"THEN1340
1230 IFA$="O"THEN1410
1240 IFA$="S"THEN1420
1250 IFA$="T"THEN1350
1260 IFA$=CHR$(12)THENRUN
1270 IFA$="/"THEN1300
1280 IFA$=CHR$(94)THEN1330
1290 GOTO1160
1300 CLS:SCREEN0,0:PRINT@42,"EYE
S":PRINT:PRINTZ$+"<B> BIG":PRINT
Z$+"<C> CLOSED":PRINTZ$+"<E> OPE
N":PRINTZ$+"<I> LITTLE":PRINTZ$+
"<L> EYES WITH LASHES":PRINTZ$+"
<M> MAD":PRINTZ$+"<O> SLEEPY OPE
N":PRINTZ$+"<S> SURPRISED":PRINT
Z$+"<T> TIRED"
1310 PRINT@416,U$+"HAIR":PRINTY$
1320 IFINKEY$=""THEN1320 ELSESCR
EEN1,1:GOTO1160
1330 PCLS:DRAWMG$:Q=1:GOTO190
1340 DRAW"BM97,89R7C6D6U6C8F6BR2
0E6C6D6U6C8R7":GOTO1450
1350 V$="FR9EBG3NL6BE3":DRAW"C6B
M99,95XV$;BR21XV$;C8":GOTO1410
1360 DRAW"BM96,88F3E2R2H3F3R5H3B
R32G3R5E2G3R3F2E2"
1370 E$="E4R3C6D5U5C8R3F5":DRAW"
BM98,92XE$;BR15XE$;":GOTO1450
1380 E$="E3R2C6D5HU2R2D2LU4R3C8F
3G2L6H2":DRAW"BM98,93XE$;BR30XE$
;":GOTO1450
1390 CIRCLE(102,89),13,,.7,.1,.4
:CIRCLE(138,89),13,,.7,.1,.4:GOT
O1450
1400 B$="H3U5E3R3F3D5G3L3C6U2H1U
2E1R2F1D2G1L2C8D2":DRAW"BM102,97
XB$;BR34XB$;":GOTO1450
1410 CIRCLE(104,91),5:CIRCLE(136
,91),5:CIRCLE(104,91),3,6:CIRCLE
(136,91),3,6:CIRCLE(104,97),13,,
.8,.6,.9:CIRCLE(136,97),13,,.8,.
6,.9:GOTO1450
1420 CIRCLE(104,91),5,6:CIRCLE(1
36,91),5,6:CIRCLE(104,91),2,7:CI
RCLE(136,91),2,7
1430 CIRCLE(104,92),9,,1,.5,0:CI
RCLE(136,92),9,,1,.5,0
1440 '*EYEBROWS
1450 GOSUB80:DRAW"C8BM213,19U8R2
F2G2NLF2G2L2BR8U8R3FD2GL3RF3DBR4
U8R4D8L4BR8BU8D6F2E2NU3F2E2U6BM2
18,29E2R3FBR6ER3F2"
1460 Q=0:DRAW"C8":A$=INKEY$
1470 IFA$=CHR$(12)THENRUN
1480 IFA$="B"THENY=1:GOTO1700
1490 IFA$="S"THENY=2:GOTO1660
1500 IFA$="I"THENY=8:GOTO1670
1510 IFA$="M"THENY=3:GOTO1690
1520 IFA$="N"THENY=9:GOTO1730
1530 IFA$="Q"THENY=4:GOTO1710
1540 IFA$="A"THENY=5:GOTO1680
1550 IFA$="T"THENY=6:GOTO1640
1560 IFA$="C"THENY=7:GOTO1650
1570 IFA$=CHR$(94)THEN1630
1580 IFA$="/"THEN1600
1590 GOTO1460
1600 CLS:SCREEN0,0:PRINT@42,"EYE
BROWS":PRINT:PRINTZ$+"<A> ARCHED
":PRINTZ$+"<B> BUSHY":PRINTZ$+"<
C> CLOWN":PRINTZ$+"<I> INFANT":P
RINTZ$+"<M> MAD":PRINTZ$+"<N> NO
NE":PRINTZ$+"<Q> QUIZZICAL":PRIN
TZ$+"<S> SURPRISED":PRINTZ$+"<T>
 TILTED"
1610 PRINT@416,U$+"EYES":PRINTY$
1620 IFINKEY$=""THEN1620ELSESCRE
EN1,1:GOTO1460
1630 LINE(88,84)-(150,99),PRESET
,BF:GOTO1150
1640 DRAW"BM90,90E8R5BR33R5F8":G
OTO1730
1650 DRAW"C7BM94,89E9ND5F9BR17E9
ND5F9C8":GOTO1730
1660 CIRCLE(102,90),12,,1.1,.6,.
9:CIRCLE(138,90),12,,1.1,.6,.9:G
OTO1730
1670 DRAW"BM98,85R11BR20R11":GOT
O1730
1680 DRAW"BM94,90E7R8F3BR17E3R8F
7":GOTO1730
1690 DRAW"BM100,81R7F9BR9E9R7":G
OTO1730
1700 DRAW"BM94,87E4R17FL18GR17BR
12R17HL18ER17F4":GOTO1730
1710 DRAW"BM94,82E4R11F3BD5BR17E
3R13F4":GOTO1730
1720 '*MOUTH
1730 IFQ=2THEN1450ELSEGOSUB80:DR
AW"BM209,17U6F2E2D6BR4U6R4D6L4BR
8NU6R4U6BR4R4L2D6BR6U6D3R4U3D6;B
M223,28F3NR6FR4E4"
1740 A$=INKEY$
1750 IFA$="B"THEN2000
1760 IFA$="C"THEN1960
1770 IFA$="H"THEN1970
1780 IFA$="I"THEN1980
1790 IFA$="L"THEN2030
1800 IFA$="M"THEN1950
1810 IFA$="O"THEN2010
1820 IFA$="S"THEN1990
1830 IFA$="T"THEN2020
1840 IFA$="V"THEN1940
1850 IFA$=CHR$(12)THENRUN
1860 IFA$="/"THEN1890
1870 IFA$=CHR$(94)THEN1920
1880 GOTO1740
1890 CLS:SCREEN0,0:PRINT@42,"MOU
TH":PRINT:PRINTZ$+"<B> BIG":PRIN
TZ$+"<C> CLOWN":PRINTZ$+"<H> HAP
PY":PRINTZ$+"<I> INFANT":PRINTZ$
+"<L> LIPSTICK":PRINTZ$+"<M> STR
AIGHT"
1900 PRINTZ$+"<O> OPEN":PRINTZ$+
"<S> SAD/MAD":PRINTZ$+"<T> TEETH
":PRINTZ$+"<V> VAMPIRE":PRINT@41
6,U$+"EYEBROWS":PRINTY$
1910 IFINKEY$=""THEN1910ELSESCRE
EN1,1:GOTO1740
1920 Q=2:DRAW"C5":ON Y GOTO1700,
1660,1690,1710,1680,1640,1930,16
70,1450
1930 IFY=7THENDRAW"BM94,89E9ND5F
9BR17E9ND5F9":GOTO1450ELSEGOTO14
50
1940 DRAW"C7BM107,120R26L5D7H2U5
L12D5G2U7C8":GOTO2060
1950 DRAW"C7BM110,118R20BG3L12C8
":GOTO2060
1960 CIRCLE(120,114),24,7,.7,.01
,.49:CIRCLE(120,113),9,7,.7,.05,
.45:DRAW"C7BM96,115U3E4R9F5BR14E
4R9F4D3C8":PAINT(120,129),8,7
1970 CIRCLE(120,115),16,7,.5,.05
,.45:GOTO2060
1980 DRAW"C7BM112,121RE3UERF2E2R
FDF3NRG3L8H3BR6R3C8":GOTO2060
1990 CIRCLE(120,120),13,7,.3,.5,
.99:GOTO2060
2000 DRAW"C7BM107,120E5R16F5G4L1
8H4R26C8":GOTO2060
2010 CIRCLE(120,119),8,7:GOTO206
0
2020 DRAW"C7BM105,118F9R12E9L30R
10C8D4R5U4D4R5U4":GOTO2060
2030 CIRCLE(121,119),11,7,.5,.08
,.47:CIRCLE(121,119),13,7,.7,.08
,.47
2040 DRAW"C7BM108,119R10F2E2R10L
2H3L4G3H3L4G3L2C8"
2050 '*NOSE
2060 GOSUB80:DRAW"BM213,15U6F6U6
BR4R4D6L4U6BR12L4D3R4D3L4BR8U6R3
BD3L3D3R3BM227,27D4G2R6H2U4"
2070 A$=INKEY$
2080 IFA$=CHR$(12)THENRUN
2090 IFA$="S"THEN2210
2100 IFA$="W"THEN2200
2110 IFA$="C"THEN2230
2120 IFA$="I"THEN PSET(117,109):
PSET(123,109):GOTO2260
2130 IFA$="N"THEN2240
2140 IFA$="P"THEN2220
2150 IFA$="/"THEN2180
2160 IFA$=CHR$(94)THENPAINT(120,
125),5,5:LINE(105,110)-(135,127)
,PRESET,BF:GOTO1730
2170 GOTO2070
2180 CLS:SCREEN0,0:PRINT@74,"NOS
E":PRINT:PRINTZ$+"<C> CLOWN":PRI
NTZ$+"<I> INFANT":PRINTZ$+"<N> N
ARROW":PRINTZ$+"<P> PUG":PRINTZ$
+"<S> STRAIGHT":PRINTZ$+"<W> WID
E":PRINT@416,U$+"MOUTH":PRINTY$
2190 IFINKEY$=""THEN2190 ELSESCR
EEN1,1:GOTO2070
2200 DRAW"BM112,110U2E2R2FNU4BE2
R5BF2NU4ER2F2D2BL5LBL7L":GOTO226
0
2210 DRAW"BM120,95D12BF3R2BL8L2"
:GOTO2260
2220 DRAW"BM124,107F3BL4LBL4LBL3
E3":GOTO2260
2230 CIRCLE(120,104),8,7:PAINT(1
20,105),7,7:GOTO2260
2240 DRAW"BM118,95D10G2D4E1R1F1R
2E1R1F1U4H2U10"
2250 '*ETC.
2260 GOSUB80:DRAW"BM218,20U6R4BD
3L4D3R4BR6U6L2R4BR4NR4D6R4BR4RUL
"
2270 A$=INKEY$
2280 IFA$="B"THEN2500
2290 IFA$="C"THEN2480
2300 IFA$="E"THEN2580
2310 IFA$="H"THEN2490
2320 IFA$="S"THEN2520
2330 IFA$="R"THEN2530
2340 IFA$="G"THEN2540
2350 IFA$="M"THEN2550
2360 IFA$="N"THEN2560
2370 IFA$="F"THEN2590
2380 IFA$="T"THEN2600
2390 IFA$="V"THEN2610
2400 IFA$=CHR$(12)THENRUN
2410 IFA$="@"THEN2620
2420 IFA$="/"THEN2450
2430 IFA$=CHR$(94)THENLINE(110,1
12)-(130,95),PRESET,BF:GOTO2060
2440 GOTO2270
2450 CLS:SCREEN0,0:PRINT@10,"ACC
ESSORIES":PRINT:PRINT" <B> BEARD
      <M> MUSTACHE       <C> CHEEK
S     <N> NECKLINE       <E> EARRI
NGS   <R> HAIR BOW       <F> FROWN
      <S> SHIRT          <G> GLASS
ES    <T> TIE            <H> HAT
      <V> VAMPIRE CAPE"
2460 PRINT@384,U$+"NOSE":PRINT"<
@> STOP DRAWING AND ENTER NAME "
+Y$
```

continued on page 18

bombs away

by Allen Drennan

BOMBS AWAY is a nuclear war between two opponents. Both sides fire missiles in an attempt to destroy each other.

Each player takes his turn by typing the angle (0-90) followed by a comma and the velocity. But be warned: If the velocity is too great, your base falls appart from the stress. If the velocity is too little, the missle blows up your base. A range of zero to 200 works best. Some experimentation is required to determine the best velocity for your computer. As you fire, remember to compensate for wind and terrian.

To win the battle, you must destroy the opponent's base. A game is won when a certain number of battles have been fought. The number of battles to be played is determined at the beginning of each battle.

The game is simple to use and the prompts are all user friendly. ENTER is used as a toggle switch to go from the text screen to the graphics screen and back again during a player's turn. Bombs Away is written in Extended Colour BASIC as a high resolution game. It should run in 16K with no modifications.

The Listing: BOMBAWAY

```
100 ' BOMBS AWAY
110 '
120 ' ALLEN DRENNAN
130 ' 1986 COLOR CLOUD
140 ' 19506-D INDUSTRIAL DR.
150 ' SONORA, CA. 95370
160 ' (209) 533-3477
170 '
180 DIM H(139):GOTO 1320
190 SCREEN 1,1:FOR I=1 TO 254 ST
EP 2:LINE(I,(H(I/2)+1))-(I,159),
PSET:NEXT
200 RETURN
210 REM
220 REM ** CREATE BASES
230 REM
240 CLS:PCLS:PRINT:PRINT:PRINT:P
RINT:PRINT:PRINT:PRINT"      NUCLE
AR BASES ACTIVE ...":N1=2:FOR I=
1 TO 1000:NEXT I:CLS
250 PMODE 4,1:SCREEN 1,1:PCLS:SC
REEN 1,1:PCLS
260 X1=20+FND(30):X2=80+FND(40):
L(1)=10+FND(X1-10):L(2)=X2+FN D(
120-X2)
270 N=158-FND(58):FOR I=0 TO X1:
H(I)=N:NEXT:N=1:GOSUB 1190
280 N=158-FND(58):FOR I=X2 TO 13
9:H(I)=N:NEXT:N=2;GOSUB 1190
290 FOR KK=1 TO 1000:NEXT KK
300 X3=X1+FND(X2-X1-20)+10:H(X3)
=50+FND(100):N=H(X3)/2:D1=N-H(1)
/2:D2=N-H(139)/2
310 REM
320 REM ** SCOREBOARD
330 REM
340 CLS:PRINT:PRINT"THE SCORE:
";P$(1);:PRINT" =";S(1):PRINT "
            ";P$(2);:PRINT" =";S(
2)
350 IF S(1)+S(2)=0 THEN 390
360 REM
370 REM ** CREATE ILLUSION
380 REM
390 A=180:R=180/(X3-X1+1):N=H(1)
+D1
400 FOR I=X1+1 TO X3-1:A=A-R:H(I
)=COS(A*.0174533)*D1+N:NEXT
410 A=0:R=180/(X2-X3+1):N=H(139)
+D2
420 FOR I=X3+1 TO X2-1:A=A+R:H(I
)=COS(A*.0174533)*D2+N:NEXT
430 CLS:GOSUB 190
440 REM
450 REM ** DISPLAY WIND FACTOR
460 REM
470 W=FND(100)-50:PRINT:PRINT:PR
INT:PRINT:PRINT:PRINT "  WIND FA
CTOR";:IF W<=0 THEN PRINT W+(W*-
2);"TO THE WEST." ELSE PRINT W;"
TO THE EAST."
480 FOR I=1 TO 1500:NEXT I:CLS:G
OSUB 1230:N=N1
490 N=3-N:S=5*N-4
500 REM
510 REM ** PLAYER PROFILE
520 REM
530 CLS:PRINT "LAST SHOTS :";:FO
R I=0 TO 3:PRINT A(S+4-I);:NEXT:
PRINT
540 PRINT"                 ";:FOR I=0
 TO 3:PRINT V(S+4-I);:NEXT I:PRI
NT
550 PRINT "FIRING :";
560 IF N=1 THEN PRINT "LEFT" ELS
E PRINT "RIGHT"
570 PRINT "WIND FACTOR";
580 IF W<=0 THEN PRINT W+(W*-2);
"TO THE WEST." ELSE PRINT W;"TO
THE EAST."
590 ANG$="":V$="":PRINT P$(N);
600 LINE INPUT " CALL YOUR SHOT
";YU$:IF YU$="" THEN 690
610 FOR I=1 TO LEN(YU$):IF MID$(
YU$,I,1)="," THEN 640
620 ANG$=ANG$+(MID$(YU$,I,1))
630 NEXT I
640 FOR I=I+1 TO LEN(YU$):V$=V$+
(MID$(YU$,I,1)):NEXT I
650 ANG=VAL(ANG$):V=VAL(V$):GOTO
 740
660 REM
670 REM ** PERFORM TOGGLE
680 REM
690 SCREEN 1,1
700 A$=INKEY$:IF A$=""THEN 730
710 IF A$=CHR$(13) THEN 530
720 GOTO 700
730 GOTO 700
740 CLS:SCREEN 1,1
750 IF V<350 THEN 790
760 PRINT:PRINT"YOUR BASE BLEW A
PART FROM TO     MUCH PRESSURE AN
D FORCE.":FOR I=1 TO 1500:NEXT I
770 FOR I=1 TO 1000:NEXT I
780 GOTO 1120
790 GOSUB 1220
800 REM
810 REM ** FIRE MISSLE
820 REM
830 IF N=2 THEN ANG=180-ANG
840 V1=COS(ANG*.0174533)*V:V2=-S
IN(ANG*.0174533)*V
850 MN=L(N):X=L(N)*2:Y=H(MN)
860 IF N=1 THEN X=X+1 ELSE X=X-6
870 X=X-7*(N=2)
880 PSET(X,Y,3):OX=0
890 XO=X:YO=Y
900 X=X+V1/10:V1=V1+(W-V1)/30:IF
 X<0 OR X>254 THEN 490
910 Y=Y+V2/10:V2=V2+6
920 IF Y<1 THEN 960
930 IF OX THEN PSET(OX,0,3):OX=0
940 PLAY"T255;O1;1;1"
950 LINE(XO,YO)-(X,Y),PSET:XO=X:
YO=Y:GOTO 970
960 OX=X
970 IF H(X/2)-Y>2 THEN 900
980 Y=H(X/2)+2
990 PSET(X,Y,3)
1000 IF ABS(X/2-L(3-N))<3 THEN 1
080
1010 IF ABS(X/2-L(N))>3 THEN 107
0
1020 PRINT:PRINT " YOU DESTROYED
 YOURSELF ";P$(N):FOR I=1 TO 150
0:NEXT I:GOSUB 1720
1030 GOTO 1120
1040 REM
1050 REM ** SOUND EFFECTS
1060 REM
1070 PLAY"T200;O1;V31;8;8;8;V25;
6;6;6;V20;4;4;4;V15;2;2;2;V10;1;
1;1":CIRCLE(X,Y),2,3:GOTO 490
1080 PLAY"O1;T255;V10;4;4;4;V15;
6;6;6;V20;8;8;8;V25;10;10;10;V31
;12;12;12;V25;10;V20;8;V15;6;V10
;4":GOSUB 1720
1090 WIN=N
1100 FOR KK=1 TO 1000:NEXT KK
1110 N=3-N
1120 S(3-N)=S(3-N)+1:IF S(3-N)+S
(N)=GN THEN 1580
1130 FOR I=1 TO 10:A(I)=0:V(I)=0
:NEXT:N1=3-N1:PCLS:GOTO 260
1140 GOTO 1580
1150 END
1160 REM
1170 REM ** DRAW BASES
1180 REM
1190 X=L(N)*2:Y=H(L(N))-1:FOR I=
-2 TO 3:LINE(X+I,Y+1)-(X+I,Y-2),
PSET:NEXT
1200 LINE(X-4,Y-4)-(X-4,Y-2),PSE
T:LINE(X-3,Y-4)-(X-3,Y-2),PSET:L
INE(X,Y-4)-(X,Y-2),PSET:LINE(X+1
,Y-4)-(X+1,Y-2),PSET
```

```
1210 LINE(X+4,Y-4)-(X+4,Y-2),PSE
T:LINE(X+5,Y-4)-(X+5,Y-2),PSET:R
ETURN
1220 NN=5*N:FOR J=1 TO 4:K=NN-5+
J:A(K)=A(K+1):V(K)=V(K+1):NEXT:V
(NN)=V:A(NN)=ANG:RETURN
1230 LINE(142-W,150)-(143+W,150)
,PSET
1240 SS=-SGN(W)
1250 FOR I=1 TO 5:Y=150-I:X=140+
W+SS*I
1260 LINE(X,Y)-(X+1,Y),PSET
1270 Y=150+I:LINE(X,Y)-(X+1,Y),P
SET:NEXT
1280 RETURN
1290 REM
1300 REM ** MAIN DISPLAY
1310 REM
1320 CLS:PRINT:PRINT:PRINT:PRINT
"        ";:A$="*** BOMBS AWAY ***
"
1330 FOR I=1 TO LEN(A$):PRINT MI
D$(A$,I,1);
1340 POKE&HFF21,&H3C:POKE&HFF21,
&H34:FOR QW=1 TO 30:NEXT QW:NEXT
1350 PRINT:PRINT:PRINT "           B
Y ALLEN DRENNAN                  1
986 COLOR CLOUD"
1360 FOR I=1 TO 2000:NEXT I
1370 CLS:A$="GAME RULES ARE SIMP
LE:":GOSUB 1540:PRINT
1380 PRINT
1390 A$="BLOW UP YOUR OPPONENT B
Y FIRING AT THE RIGHT ANGLE AND
VELOCITY,":GOSUB 1540
1400 PRINT
1410 A$="WHILE COMPENSATING FOR
WIND AND TERRAIN.  EACH GUNNER M
UST ENTER"
1420 A$=A$+"THE GUN ANGLE AND SH
ELL POWER.":GOSUB 1540
1430 PRINT:PRINT
1440 A$="THE ANGLE MUST BE BETWE
EN (0-90)":GOSUB 1540
1450 PRINT
1460 PRINT " PLAYERS NAMES :"
1470 FOR P=1 TO 2
1480 PRINT "PLAYER ";P;:LINE INP
UT", ";P$(P):IF LEN(P$(P))>10 TH
EN CLS:PRINT "10 CHARACTERS ONLY
!":GOTO 1460
1490 NEXT
1500 DEF FND<X)=RND(X)
1510 INPUT "HOW MANY BATTLES TO
PLAY ";GN
1520 IF GN<1 THEN PRINT"PLEASE,
DONT JOKE AROUND!":GOTO 1510
1530 GOTO 240
1540 FOR I=1 TO LEN(A$):PRINT MI
D$(A$,I,1);
1550 POKE &HFF21,&H3C:POKE &HFF2
1,&H34:FOR QW=1 TO 30:NEXT QW:NE
XT
1560 FOR I=1 TO 700:NEXT I
1570 RETURN
1580 CLS
1590 REM
1600 REM ** GAME REPORTS
1610 REM
1620 PRINT "THE SCORE:  ";P$(1);
:PRINT" =";S(1):PRINT P$(2);:PRI
NT" =";S(2)
1630 IF S(1)>S(2) THEN WI=1
1640 IF S(1)<S(2) THEN WI=2
1650 IF S(1)=S(2) THEN 1680
1660 PRINT "WINNER"
1670 GOTO 1700
1680 PRINT
1690 PRINT "A TIE !!!"
1700 PRINT "BETTER LUCK NEXT TIM
E !!!"
1710 END
1720 SCREEN 1,1:FOR RT=1 TO 15
1730 CIRCLE(X,Y),RT,3,1,.50,0
1740 NEXT RT
1750 FOR I=1 TO 1500:NEXT I:RETU
RN
```

Portraits by BASIC

continued from page 16

```
2470 IFINKEY$=""THEN2470 ELSE SC
REEN1,1:GOTO2270
2480 DRAW"BM96,105U3E3R3F3D3G3L3
H3BR41U3E3R3F3D3G3L3H3":PAINT(99
,105),8,8:PAINT(145,105),8,8:GOT
O2270
2490 B=1:CIRCLE(120,55),70,7,.26
,.8,.7:CIRCLE(120,45),40,7,1,.45
,.05:PAINT(120,50),7,7:GOTO2270
2500 BD$="BM86,76D33F22R24E22U33
R9D45G30L25H30U45R9":IFB=1THEN25
10ELSEPAINT(124,54),8,8:REM
*BEARD*
2510 DRAW"C7XBD$;C8":PAINT(120,1
40),8,7:DRAWBD$:GOTO2270
2520 DRAW"BM90,138F30D10U10E30D1
5G20H10G10H20U15":GOTO2270
2530 B=1:DRAW"C6BM116,40H8L10G2D
18F2R10E8R6F8R10E2U18H2L10G8L6C8
":PAINT(116,43),6,6:GOTO2270
2540 CIRCLE(102,91),14,7,.7:CIRC
LE(138,91),14,7,.7:DRAW"C7BM79,8
0F10BR27R10BR27E10C8":GOTO2270
2550 DRAW"BM98,118E6R32F6H6L4G1L
18H1":GOTO2270:REM**MUSTACHE**
2560 IFF=4THENCIRCLE(120,120),40
,,.7,.1,.4ELSECIRCLE(120,130),40
,,.7,.1,.4
2570 GOTO2270
2580 CIRCLE(75,105),5,6:CIRCLE(1
69,105),5,6:GOTO2270
2590 DRAW"BM118,82D5BR5U5":GOTO2
270
2600 DRAW"BM108,156C6RF7E2R4F2E7
RD16LH7G2L4H2G7LU16C8":PAINT(120
,163),6,6:GOTO2270
2610 DRAW"C7BM47,167E25H15R30M97
,152G15L35BR149H25E15L30M144,152
F15R35C8":PAINT(82,157),7,7:PAIN
T(148,152),7,7:FORH=99TO145STEP7
:CIRCLE(H,152),4:NEXTH:GOTO2270
2620 LINE(200,1)-(249,50),PRESET
,BF:LINE(6,185)-(245,170),PRESET
,BF
2630 DRAW"C5BM0,185BR85C6":L=85
2640 IFL<10THENL=10:SOUND15,1:DR
AW"BM10,185"ELSEIFL>240THENSOUND
1,1:GOTO2820
2650 A$=INKEY$:IFA$=""THEN2650
2660 IFA$="@"THEN2620
2670 IFA$="^"THENDRAW"C8":GOTO22
70
2680 IFA$="A"THENDRAW"M+0,0U10E4
F4D3NL8D7BR9"ELSEIFA$="B"THENDRA
W"M+0,0U14R5F2D2G2L5R6F2D4G2L5BR
16"
2690 IFA$="C"THENDRAW"BM+3,0H3U8
E3R2F3BD8G3L2BR13"ELSEIFA$="D"TH
ENDRAW"M+0,0U14R5F3D8G3L5BR17"
2700 IFA$="E"THENDRAW"M+0,0U14R8
BD7BL2L6D7R8BR8"ELSEIFA$="Y"THEN
DRAW"BM+4,0U7H4U3BR8D3G4D7BR12"
2710 IFA$="V"THENDRAW"BM+0,-14D1
0F4E4U10BD14BR8"ELSEIFA$="I"THEN
DRAW"M+0,0R2U14L2R4BD14L2BR9":L=
L-4
2720 IFA$="M"THENDRAW"M+0,0U14F5
E5D14BR8":L=L+2ELSEIFA$="O"THEND
RAW"BM+3,0H3U8E3R2F3D8G3L2BR13"
2730 IFA$="L"THENDRAW"NU14R8BR8"
ELSEIFA$="N"THENDRAW"U14M+8,14NU
14BR6"ELSEIFA$="R"THENDRAW"M+0,0
U14R6F2D3G2NL6F2D5BR8"
2740 IFA$="Z"THENDRAW"M+0,0BU14R
8D3G8D3R8BR8"ELSEIFA$="T"THENDRA
W"BM+4,0U14L4R8BD14BR8"ELSEIFA$=
"H"THENDRAW"M+0,0U14D7R8U7D14BR8
"
2750 IFA$="F"THENDRAW"M+0,0U14R8
BD7BL2L6D7BR16"ELSEIFA$="G"THEND
RAW"BM+2,0H2U10E2R4F2BD6NL3D4G2L
4BR14"
2760 IFA$="J"THENDRAW"BM+3,0NH3R
2E3U11BD14BR8"ELSEIFA$="K"THENDR
AW"M+0,0U14BR8G8E4F4D6BR8"ELSEIF
A$="P"THENDRAW"M+0,0U14R4F3D3G3L
4D5BR16"
2770 IFA$="Q"THENDRAW"BM+3,0H3U8
E3R3F3D8G3L3R2BU4F4BR8"ELSEIFA$=
"S"THENDRAW"BM+0,-3F3R2E3U2H3L2H
2U2E2R3F3BD11BR8"
2780 IFA$="U"THENDRAW"BM+0,-14D1
1F3R3E3U11BD14BR8"ELSEIFA$="W"TH
ENL=L+2:DRAW"BM+0,-14D14E5F5NU14
BR8"ELSEIFA$="X"THENDRAW"M+8,-14
BL8M+8,14BR8"
2790 IFA$=CHR$(8)THENDRAW"BM+0,0
BL17":L=L-32ELSEIFA$=CHR$(32)THE
NDRAW"BM+0,0BR17"ELSEIFA$=CHR$(1
2)THENRUN
2800 IFA$="."THENDRAW"BM-2,0RULD
BR8"ELSEIFA$="/"THEN2860
2810 L=L+16:GOTO2640
2820 A$=INKEY$
2830 IFA$="@"THEN2620
2840 IFA$=CHR$(12)THENRUN
2850 GOTO2820
2860 CLS:SCREEN0,0:PRINT@42,"ENT
ER NAME":PRINT:PRINT" IF THE NAM
E IS 4 OR 5 LETTERS  LONG, JUST
TYPE IT IN.  IF IT ISLONGER, HIT
 LEFT ARROW ONCE FOR EACH TWO LE
TTERS MORE  THAN 5."
2870 PRINT:PRINT"<SPACE BAR> WIL
L ENTER A SPACE.":PRINT" <^> WIL
L TAKE YOU BACK TO        ACCESSOR
IES.":PRINTY$
2880 IFINKEY$=""THEN2880ELSESCRE
EN1,1:GOTO2650
```

CORRECTION

OOPS

by Rick Adams & Dale Lear

....we glitched! Last month we loaded some CoCo 3 programs into a CoCo 2 with a 1.4 DOS. No wonder they look funny!

Anyway, here are the proper versions on those two programs, this time loaded into a CoCo 3. Sorry for any inconvenience.

The Listing:

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

The Listing:

```
10 '**************************
20 '*        "WAGON WHEEL"      *
30 '*     DEMO TO SHOW USE      *
40 '*  OF PALETTE REGISTERS     *
50 '*     IN ANIMATATION        *
60 '*BY RICK ADAMS & DALE LEAR*
70 '**************************
80 '
90 '**************
100 ' SET UP
110 '**************
120 POKE &HFFD9,0
130 HSCREEN 2
140 HCLS(1)
150 PALETTE 0,24
160 '
170 '*********************
180 ' DRAW OUTSIDE OF WHEEL
190 '*********************
200 HCIRCLE (160,96),90,0
210 HPAINT (0,0),0,0
220 '
230 '*********************
240 ' DRAW SPOKES
250 K=14*8
260 '*********************
270 FOR I=0 TO K-1
280 X=90*SIN(I*3.14/K)
290 Y=90*COS(I*3.12/K)
300 HCOLOR 2+14*(I/14-INT(I/14))
,1
310 HLINE (160+X,96+Y)-(160-X,96
-Y),PSET
320 NEXT I
330 FOR I=1 TO 30
340 HCIRCLE (160,96),I,0
350 NEXT I
360 '
370 '*********************
380 ' SET ALL PALETTE
390 ' COLORS TO WHITE
400 ' EXCEPT ONE
410 '*********************
420 FOR I=1 TO 15
430 PALETTE I,255
440 NEXT I
450 '
460 '************************
470 ' ROTATE WHEEL BY SETTING
480 ' ONE PALETTE REGISTER
490 ' AT A TIME TBLACK
500 '************************
510 K=2
520 KK=K+1
530 IF KK=16 THEN KK=2
540 PALETTE K,255:PALETTE KK,0
550 K=KK
560 GOTO 520
570 '
580 '*************************
590 ' RESTORE PALETTE ON BREAK
600 '************************
610 PALETTE RGB
620 STOP
```

FRICKERS FOLLIES

continued from page 51

from the default values when you go to run the program. You do this by setting up the registers from &HFFB0 to &HFFBF.

To give you the address and the result in order. These numbers are all in hex which means that you will have to add the "&H" prefix.

```
FFB0=background in pmode3 screen 1,0
FFB1-FFB3=fore  in pmode3 screen 1,0
FFB4=background in pmode3 screen 1,1
FFB5-FFB7=fore  in pmode3 screen 1,1
FFB8=background in pmode4 screen 1,0
FFB9=foreground in pmode4 screen 1,0
FFBA=background in pmode4 screen 1,1
FFBB=foreground in pmode4 screen 1,1
FFBC=background in 32 col screen 0,0
FFBD=foreground in 32 col screen 0,0
FFBE=background in 32 col screen 0,1
FFBF=foreground in 32 col screen 0,1
```

To use these on your existing programs you can either modify the program which means adding more code in an assembly program or using a basic loader program which will poke the colours that you want into the registers or if it is a BASIC program add the necessary lines.

One problem with this is that the registers are reset to the original values ahen you reset the computer which is a problem with some machine language arcade games.

PROGRAMMING UTILITY

Let CoCo take the tedium out of Adventure writing and leave the creativity to you

The Adventure Processor

Developing an Adventure game is not as difficult as one might imagine. Adventures are simply a collection of data and a series of true and false tests comparing the player's inputs to a list of data stored in memory of the computer.

The simplest way to store long lists of data in a computer's memory is through the use of arrays. Writing an Adventure game generally requires that information such as room descriptions, object lists, authorized user inputs and key responses be read into arrays so that they may be called upon quickly.

Arranging the data, formatting the text screen and processing of standard commands is required in all Adventures. Some programmers accomplish it differently. I prefer to keep it simple and easy to follow, since BASIC programs should be a learning experience.

About the Program

Imagine a program that could actually write most of the code for you, automatically! A program that could save you hours of tedious writing, testing and debugging — a program that would function error-free, and in a matter of minutes save you more than 50 percent of the work in putting together your dream program.

ADV-PRO, or Adventure Processor, is a utility to save you hours of tedious typing of repetitive code. It simply writes a "shell" of an Adventure for you. It provides the following possibilities:

Up to 100 rooms, 60 objects and 30 commands.

A separate help message for every location in the game.

Individual score values for each object found.

Customized responses for each object "examined."

Randomized object placement, if desired, to make your game play differently every time.

Scroll-protected split screens.

Save game in progress/load previous unfinished game capability.

Operates with memory-stretching PCLEAR ZERO.

Outputs to tape or disk.

The first step in writing an Adventure is mapping it out on paper. On your map you should indicate the major compass points at the top, bottom and sides of the sheet. N, S, E, W, Up, Down, should all be indicated for ease in laying out the Adventure.

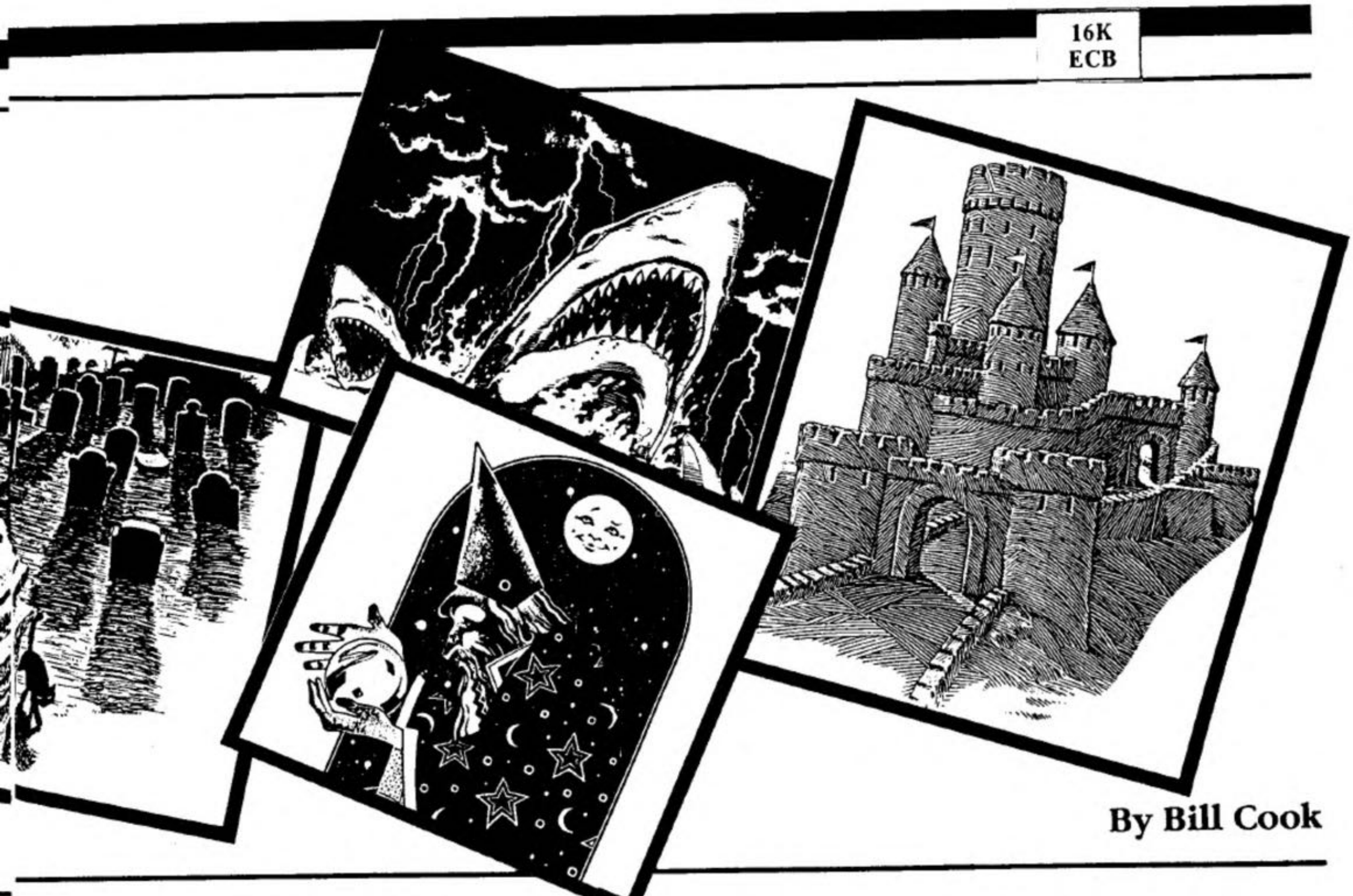
Each location should have a number as should each object you plan to place in the game. Have a good idea of the vocabulary (verbs and nouns) you want the program to recognize; two word sentences are the standard. Once you have completed this, jot down how many rooms you'll have, the number of objects and the number of commands (verbs).

Using the Program

ADV-PRO asks you for the number of items and limits you to 100 rooms, 60 objects and 30 commands. You are also asked for the room number in which you want the game to begin and for a filename. You are then asked whether to direct the output to tape or disk. After answering these questions *ADV-PRO* goes to work and creates an Adventure "shell."

Within a few minutes, you will be over half done with creating an efficient and versatile Adventure game. *ADV-PRO* writes to disk or tape, an ASCII file that is a loadable BASIC program. The pre-written coding sets up a machine language anti-scroll routine, frees the maximum available memory, in-

Bill Cook is a manager for the Navy Exchange in Whidbey Island, Washington. He is the author of The Adventure Generator *and wrote his first Adventure in 1982. He uses the CoCo extensively for business applications and as a management aide.*



16K
ECB

By Bill Cook

itializes and reserves line numbers for all of your room descriptions, legal movement directions, help messages, object descriptions, noun lists, object score values, initial locations and verb lists. You simply modify the program with your customized data.

Here's a sample room description DATA line as generated: 10 DATA ROOM # 1 DESCRIPTION,0,0,0,0,0,0,HELP MESSAGE HERE.

If you want room one to be described to the user: YOU ARE IN THE LIVING ROOM, simply change the line as follows: 10 DATA IN THE LIVING ROOM ,0,0,0,0,0,0,HELP MESSAGE HERE. Notice that the "you are" is not necessary. The program automatically precedes each room description with "you are."

Next you decide in which directions the player will be able to move from this room. Let's assume that moving north takes you to room three, south to room four, east to room six, west to room 10, up and down lead nowhere. These locations should replace the series of zeroes that come next in the above data statement. Rooms that lead nowhere remain at zero. The line would now read like this:

```
10 DATA IN THE LIVING ROOM,3,4,
6,10,0,0,HELP MESSAGE HERE
```

Now for the help message. Assume that the player is in room one and he enters the command HELP. The remainder of the DATA statement should contain whatever response you would like the player to receive. For example, YOU SENSE A PRESENCE HERE. This phrase becomes the final part of the DATA statement:

```
10 DATA IN THE LIVING ROOM
,3,4,6,10,0,0,YOU SENSE A PRES
ENCE HERE
```

If you would like no help to be given to the user, simply leave off the phrase with the comma preceding it. This causes an automatic response of NO HELP HERE. Here's how the line would look:

```
10 DATA IN THE LIVING ROOM
,3,4,6,10,0,0
```

The standard format for object data looks like this:

```
152 DATA OBJECT # 1 DESCRIPTION,
KEYWORD,0,0,RESPONSE WHEN
EXAMINED
```

As with the room descriptions, modify the line to enter your object description. Let's assume your first object is a small rusty knife, you want it located in room six, it is worth 10 points if carried and if the player says EXAMINE KNIFE you want the game to respond with IT HAS A PEARL HANDLE. Here's how your modified line should look:

```
152 DATA A SMALL RUSTY KNIFE,
KNIFE,6,10,IT HAS A PEARL
HANDLE.
```

If you would like an object to be placed in the player's inventory initially, use location -1. If you would like an object to be placed in a randomized location (unknown), use location -2. Use of randomized object locations will make your game play differently every time.

Commands

The first seven commands the game recognizes are already built in to the game. They are: EXAMINE, INVENTORY, QUIT, SCORE, HELP, SAVE and LOAD. All the necessary coding for these commands to function is already written into your program. You can, of course, modify the code, but it will function as is. The remaining commands (if you specified more than seven) are represented in the program as null strings. The line would look like this:

```
508 V$( 8)=""
```

If you want the eighth command to be GET, simply change the line as follows:

```
508 V$( 8)="GET"
```

After making the changes to include the entire verb list, you are finished with the data portion of the Adventure.

Verb processing and conditional statements are the toughest parts of Adventure programming, and the most time-consuming. Let's still assume verb eight is GET. Processing for verb eight is accomplished between lines 7500 and 7990. This is the area where you process the different possibilities of reactions to the player's use of the verb GET. This is where you exercise your own programming talent and creativity. Here is a brief sample of what could be done:

```
7510 IF LO(N)=-1 THEN PRINT"YOU
  ALREADY HAVE IT.":GOTO 60000
7520 IF LO(N)< >L THEN PRINT"I
  DON'T SEE IT.":GOTO 60000
7530 IF CA=5 THEN PRINT"YOUR
  ARMS ARE FULL.":GOTO 60000
7540 LO(N)=-1:CA=CA+1:PRINT
  "OKAY. YOU HAVE IT.":GOTO 60000
```

Significant Variables
(In order of appearance)

R	Total number of rooms in the game
R$(n)	Description of room n
D(n,nn)	Authorized directions from room n
H$(n)	Help messages when in room n
O	Total number of objects
O$(n,1)	Description of object n
O$(n,2)	Keyword in description for object n
LO(n)	Room location of object n
SC(n)	Score value of object n
O$(n,3)	Response when object n is examined
V$(n)	Command (verb) n
NV	Total number of verbs
V1$	String containing first four characters of each verb
N1$	String containing first four characters of each object
C$(d)	Labels for directions
L	Player's current location
L5	Temporary location storage flag
LN	Line counter
Z	Temporary flag for inventory test
P	Test location for anti-scrolling
TURNS	Turn counter
I$	User's input
V2$	User's command (verb)
N2$	User's object (noun)
V$	Truncated verb
N$	Truncated noun
V	Verb number
N	Noun number
SC	Score counter
MX	Possible score
DV	Device number for loading/saving -1=tape 1=disk
F$	Filename for saving/loading

Line 7510 checks to see if the object is already in the player's inventory and, if so, responds. Line 7520 checks to see if the object is in the current room and, if not, responds. Line 7540 places the object in inventory, increments the number of objects carried by one, and responds that you have the object.

With a little experimentation and patience, you will be writing professional quality Adventures in no time. I look forward to seeing your contributions in future issues of this magazine and wish you happy Adventuring.

520	32	1040	220
560	241	1115	217
605	130	1200	214
720	143	1290	213
808	40	1350	30
890	38	END	7

The listing: ADV-PRO

```
1 'ADV-PRO
2 '(C) 1986 ALL RIGHTS RESERVED
3 'PROGRAM BY BILL COOK
4 'ISLAND SOFTWARE
5 '
6 'AN ADVENTURE PROCESSOR
7 '
10 GOTO63950
20 'initialize
100 CLEAR1500:CS$=CHR$(142):EL$=
STRING$(32,32):SG$=STRING$(32,21
7):X=0
110 DIMV(31),V$(30):V$(1)="EXAMI
NE":V$(2)="INVENTORY":V$(3)="QUI
T":V$(4)="SCORE":V$(5)="HELP":V$
(6)="LOAD":V$(7)="SAVE"
170 GOTO500
172 A$=STR$(LN)+A$:PRINT#DV,A$:P
RINT@128,A$:PRINTEL$;EL$;EL$:LN=
LN+G:RETURN
320 'centering routine
330 T=LEN(T$):PRINTTAB(INT(32-T)
/2);T$:RETURN
470 'title routine
480 CLS:T$="ADVENTURE PROCESSOR"
:GOSUB330:T$="(C) 1986 BY BILL C
OOK":GOSUB330:PRINTSG$:RETURN
500 GOSUB480
510 INPUT"NUMBER OF ROOMS (1-100
)";RM:IFRM=0 OR RM>100 THEN510
520 INPUT"NUMBER OF OBJECTS (1-6
0)";NO:IFNO=0 OR NO>60 THEN 520
530 INPUT"NUMBER OF VERBS INCLUD
ING THE   7 BUILT-IN (1-30)";NV:
IFNV=0 OR NV>30 THEN 530
531 GOSUB480:INPUT"ADVENTURE TO
START IN WHICH ROOMNUMBER";L:IFL
<1 OR L>RM THEN 531
532 GOSUB480:LINEINPUT"FILENAME
(8 CHARS.MAX.):";F1$:IFLEN(F1$)>
8 THEN532 ELSE IFINSTR(F1$,".")>
0 OR INSTR(F1$,"/")>0 THEN532
534 LINEINPUT"OUTPUT TO DISK OR
TAPE (D/T)?";DT$:IFDT$="D"THENDV
=1:F1$=F1$+"/BAS": ELSE IFDT$="T
"THENDV=-1 ELSE 534
535 IFDV=-1THENLINEINPUT"PRESS E
NTER WHEN TAPE READY";Z$
536 OPEN"O",#DV,F1$
540 LN=0:G=1
541 REM process initialization
546 A$="GOTO63950":GOSUB172
550 A$="CLEAR600,&H7FB5:IFPEEK(&
H7FB6)=57THEN4":GOSUB172
560 A$="Y=0:DX$="+CHR$(34)+"BE01
68AF8C0C308C0CBF01688639A78CEF39
55550234170D6F26109E888C05E02D09
810D270A8C05FF270535176E9CE2A68C
E1C6203DC30400308C0934101F013416
7EA34E0A8920E2"+CHR$(34):GOSUB17
2
570 A$="FORP=1TOLEN(DX$) STEP2:A
$="+CHR$(34)+"&H"+CHR$(34)+"+MID
$(DX$,P,2):A=VAL(A$):POKE&H7FB6+
Y,A:Y=Y+1:NEXT:EXEC&H7FB6":GOSUB
172
580 A$="POKE&H7FCA,8":GOSUB172
590 A$="DIMR$(100),RM(100),D(100
,6),H$(100),V$(30),O$(60,3),LO(6
0),SC(60),C$(6)":GOSUB172
591 A$="REM FORMAT FOR ROOM DATA
=DESCRITION,DESTINATIONS(N,S,E,W
,U,D),HELP RESPONSE":GOSUB172
592 LN=10:G=1:FORQP=1TORM:A$="DA
TA ROOM #"+STR$(QP)+" DESCRIPTIO
N,0,0,0,0,0,0,HELP MESSAGE HERE"
:GOSUB172:NEXT
595 LN=150:G=1
600 A$="R="+STR$(RM)+":FORI=1TOR
:READR$(I):FORA=1TO6:READD(I,A):
NEXT:READH$(I):NEXT":GOSUB172
601 A$="REM FORMAT FOR OBJECT DA
TA=DESCRIPTION,KEYWORD,ROOM #LOC
ATION,POINT #VALUE,RESPONSE WHEN
 EXAMINED":GOSUB172
602 FORQP=1TONO:A$="DATA OBJECT
#"+STR$(QP)+" DESCRIPTION,KEYWOR
D,0,0,RESPONSE WHEN EXAMINED":GO
SUB172:NEXT
605 LN=500:G=1
610 A$="O="+STR$(NO)+":FORI=1TOO
:READO$(I,1),O$(I,2),LO(I),SC(I)
,O$(I,3):NEXT:T=RND(-TIMER):FORI
=1TOO:IFLO(I)=-2 THENLO(I)=RND(O
):NEXT:ELSENEXT":GOSUB172
612 FORQP=1TONV:A$="V$("+STR$(QP
)+")="+CHR$(34)+V$(QP)+CHR$(34):
GOSUB172:NEXT
620 LN=1000:G=10
630 A$="NV="+STR$(NV)+":FORI=1TO
NV:V1$=V1$+LEFT$(V$(I),4):NEXT":
GOSUB172
640 A$="FORI=1TOO:N1$=N1$+LEFT$(
O$(I,2),4):NEXT":GOSUB172
650 LN=3050
660 A$="DATANORTH,SOUTH,EAST,WES
T,UP,DOWN:FORDD=1TO6:READC$(DD):
NEXTDD":GOSUB172
670 LN=3100:G=10
690 A$="L="+STR$(L)+":L5="+STR$(
L)+":T=0:SG$=STRING$(32,217):EL$
=STRING$(32,32):CLS:LN=0":GOSUB1
72
700 REM process adv screen
710 L1=LN:A$="PRINT@0,"+CHR$(34)
+"YOU ARE "+CHR$(34)+"R$(L)"+CHR
$(34)+"."+CHR$(34):GOSUB172
720 A$="PRINT"+CHR$(34)+"YOU SEE
:"+CHR$(34)+";":GOSUB172
730 A$="Z=0:FORA=1TOO":GOSUB172
```

```
740 A$="IFLO(A)=L AND POS(0)+LEN
(O$(A,1))>32 THENPRINT":GOSUB172
750 A$="IFLO(A)=L THENPRINTO$(A,
1)+CHR$(44);:Z=1":GOSUB172
760 A$="NEXT:PRINTCHR$(8);"+CHR$
(34)+"."+CHR$(34)+";":GOSUB172
770 A$="IFZ=0THENPRINTCHR$(8)+"+
CHR$(34)+":NOTHING OF INTEREST."
+CHR$(34):GOSUB172
780 A$="PRINT:PRINT:PRINT"+CHR$(
34)+"OBVIOUS EXITS LEAD: "+CHR$(
34):GOSUB172
790 A$="FORG=1TO6:IFD(L,G)<>0THE
NPRINTC$(G)+CHR$(32);":GOSUB172
800 A$="NEXT:PRINT:PRINTSG$;:P=P
EEK(136)*256+PEEK(137)-1024:POKE
&H7FCA,INT(P/32)":GOSUB172
802 A$="FORI=P+1024 TO 1504 STEP
32:IFPEEK(I)=217THEN LN=1:ELSE N
EXTI":GOSUB172
803 A$="IFLN>0 THENFORJ=P TO LN-
1024 STEP32:PRINT@J,EL$;:NEXTJ:L
N=0":GOSUB172
804 REM welcome
805 MG$="WELCOME TO THE WONDERFU
L WORLD  OF ADVENTURE. GOOD LUCK
!"
807 A$="IFTURNS=0THENPRINT@480,"
+CHR$(34)+MG$+CHR$(34):GOSUB172
808 REM player input
810 L2=LN:A$="PRINT@480,;:TURNS=
TURNS+1:I$="+CHR$(34)+CHR$(34)+"
:LINEINPUT"+CHR$(34)+"WHAT NOW?
"+CHR$(34)+";I$":GOSUB172
820 A$="IFI$="+CHR$(34)+CHR$(34)
+"THENPRINT"+CHR$(34)+"WHAT?"+CH
R$(34)+":GOTO"+STR$(L2):GOSUB172
830 A$="IFI$="+CHR$(34)+"LOOK"+C
HR$(34)+"THEN"+STR$(L1):GOSUB172
840 A$="IFLEN(I$)>1THEN"+STR$(LN
+40):GOSUB172
850 A$="L5=L":GOSUB172
860 A$="G=INSTR("+CHR$(34)+"NSEW
UD"+CHR$(34)+",I$):IFG=0THENPRIN
T"+CHR$(34)+"I DON'T UNDERSTAND.
"+CHR$(34)+":GOTO"+STR$(L2):GOSU
B172870 A$="IFD(L,G)>0THEN L5=D(
L,G):L=L5:GOTO"+STR$(L1)+":ELSEP
RINT"+CHR$(34)+"YOU CAN'T GO THA
T WAY."+CHR$(34)+":GOTO"+STR$(L2
):GOSUB172
880 A$="I$=I$+"+CHR$(34)+" "+CHR
$(34)+":SP=INSTR(I$,CHR$(32))":G
OSUB172
890 A$="V2$=LEFT$(I$,SP-1):N2$=M
ID$(I$,SP+1):V$=LEFT$(V2$,4):N$=
LEFT$(N2$,4):V=INSTR(V1$,V$):N=I
NSTR(N1$,N$)":GOSUB172
900 A$="IFV=0THENPRINT"+CHR$(34)
+"I DON'T UNDERSTAND."+CHR$(34)+
":GOTO"+STR$(L2)+":ELSEV=(V-1)/4
+1":GOSUB172
910 A$="IFN=0THENPRINT"+CHR$(34)
+"I DON'T UNDERSTAND."+CHR$(34)+
":GOTO"+STR$(L2)+":ELSEN=(N-1)/4
+1":GOSUB172
915 REM on goto
920 L3=LN:A$="ON V GOTO"
930 LL=4000
940 FORI=1TONV
950 LL$=STR$(LL):T=LEN(LL$):LL$=
RIGHT$(LL$,T-1)
960 A$=A$+LL$+","
970 V(I)=LL:LL=LL+500
980 NEXT
990 T=LEN(A$):A$=LEFT$(A$,T-1)
992 GOSUB172:GOSUB480
1000 FORI=1TONV
1010 LN=V(I):A$="REM VERB #"+STR
$(I)+" "+V$(I):GOSUB172
1011 REM default each verb
1012 LN=V(I)+490:A$="PRINT"+CHR$
(34)+"I DON'T UNDERSTAND."+CHR$(
34)+":GOTO"+STR$(L2)
1015 GOSUB172
1020 NEXT
1025 REM examine
1030 LN=V(1)+10
1040 A$="IF LO(N)<>-1 AND LO(N)<
>L THENPRINT"+CHR$(34)+"YOU CAN'
T EXAMINE SOMETHING YOU DO NOT H
AVE OR CANNOT SEE."+CHR$(34)+":G
OTO"+STR$(L2):GOSUB172
1042 A$="IFO$(N,3)="+CHR$(34)+CH
R$(34)+"THENPRINT"+CHR$(34)+"NOT
HING SPECIAL."+CHR$(34)+":GOTO"+
STR$(L2):GOSUB172
1043 A$="PRINTO$(N,3):GOTO"+STR$
(L2):GOSUB172
1045 REM inventory
1050 LN=V(2)+10
1060 A$="PRINT"+CHR$(34)+"YOUR I
NVENTORY:"+CHR$(34)+":NH=0":GOSU
B172
1070 A$="FORI=1TOO:IFLO(I)=-1THE
NNH=1:PRINTO$(I,1)":GOSUB172
1080 A$="NEXT:IFNH=0THENPRINT"+C
HR$(34)+"NOTHING."+CHR$(34):GOSU
B172
1090 A$="GOTO"+STR$(L2):GOSUB172
1095 REM quit
1100 LN=V(3)+10
1110 A$="SC=0:PRINT"+CHR$(34)+"G
AME ENDS AFTER"+CHR$(34)+"TURNS"
+CHR$(34)+"TURNS."+CHR$(34)+":FO
RI=1TOO:IFLO(I)=-1THENSC=SC+SC(I
):NEXT:ELSENEXT":GOSUB172
1115 A$="PRINT"+CHR$(34)+"YOU SC
ORED"+CHR$(34)+"SC"+CHR$(34)+"PO
INTS."+CHR$(34)+":POKE&H7FCA,0:P
OKE&HBA,PEEK(&HBC):POKE&HB7,PEEK
(&HBC)+6:END":GOSUB172
1118 REM score
1120 LN=V(4)+10
1130 A$="SC=0:MX=0:FORI=1TOO:IFL
O(I)=-1THENSC=SC+SC(I):MX=MX+SC(
I):NEXT:ELSEMX=MX+SC(I):NEXT":GO
SUB172
1140 A$="PRINT"+CHR$(34)+"YOU HA
VE SCORED"+CHR$(34)+"SC:PRINT"+C
HR$(34)+"OUT OF A POSSIBLE"+CHR$
(34)+"MX:GOTO"+STR$(L2):GOSUB172
1145 REM help
1150 LN=V(5)+10
1160 A$="IFH$(L)="+CHR$(34)+CHR$
(34)+"THENPRINT"+CHR$(34)+"NO HE
LP HERE."+CHR$(34)+":GOTO"+STR$(
L2)+" ELSEPRINTH$(L):GOTO"+STR$(
L2):GOSUB172
1165 REM check for get or drop
1170 LN=60000:A$=" IFV$="+CHR$(3
4)+"GET"+CHR$(34)+"THEN"+STR$(L1
):GOSUB172
1180 A$=" IFV$="+CHR$(34)+"DROP"
+CHR$(34)+"THEN"+STR$(L1):GOSUB1
72
1190 A$=" GOTO"+STR$(L2):GOSUB17
2
1195 REM load
1200 LN=V(6)+10
1210 A$="LINEINPUT"+CHR$(34)+"FI
LENAME TO LOAD:"+CHR$(34)+";F$":
GOSUB172
1220 A$="IFLEN(F$)>8THENPRINT"+C
HR$(34)+"TOO LONG."+CHR$(34)+":G
OTO"+STR$(LN-10):GOSUB172
1230 A$="PRINT"+CHR$(34)+"TAPE O
R DISK? (T/D)"+CHR$(34):GOSUB172
1240 A$="A$=INKEY$:IFA$="+CHR$(3
4)+CHR$(34)+"THEN"+STR$(LN)+" EL
SE A=INSTR("+CHR$(34)+"TD"+CHR$(
34)+",A$):IFA=0 THEN"+STR$(LN)+"
 ELSE IFA=1 THENDV=-1 ELSEDV=1":
GOSUB172
1250 A$="IFDV=-1THENPRINT"+CHR$(
34)+"READY TAPE, PRESS ENTER.."+
CHR$(34)+";:LINEINPUTZ$":GOSUB17
2
1260 A$="PRINT"+CHR$(34)+"LOADIN
G "+CHR$(34)+";F$:OPEN"+CHR$(34)
+"I"+CHR$(34)+",DV,F$:FORI=1TOO:
INPUT#DV,LO(I):NEXT:INPUT#DV,L,T
URNS,CA":GOSUB172
1270 A$="CLOSE:GOTO60000":GOSUB1
72
1275 REM save
1280 LN=V(7)+10
1290 A$="LINEINPUT"+CHR$(34)+"FI
LENAME FOR SAVING:"+CHR$(34)+";F
$":GOSUB172
1300 A$="IFLEN(F$)>8THENPRINT"+C
HR$(34)+"TOO LONG."+CHR$(34)+":G
OTO"+STR$(LN-10):GOSUB172
1310 A$="PRINT"+CHR$(34)+"TAPE O
R DISK? (T/D)":GOSUB172
1320 A$="A$=INKEY$:IFA$="+CHR$(3
4)+CHR$(34)+"THEN"+STR$(LN)+" EL
SE A=INSTR("+CHR$(34)+"TD"+CHR$(
34)+",A$):IFA=0THEN"+STR$(LN)+"
ELSEIFA=1THENDV=-1 ELSEDV=1":GOS
UB172
1330 A$="IFDV=-1THENPRINT"+CHR$(
34)+"READY TAPE, PRESS ENTER.."+
CHR$(34)+";:LINEINPUTZ$":GOSUB17
2
1340 A$="PRINT"+CHR$(34)+"SAVING
 "+CHR$(34)+";F$:OPEN"+CHR$(34)+
"O"+CHR$(34)+",DV,F$:FORI=1TOO:P
RINT#DV,LO(I):NEXT:PRINT#DV,L,TU
RNS,CA":GOSUB172
1350 A$="CLOSE:GOTO60000":GOSUB1
72
1355 REM pclear zero
1360 LN=63950
1370 A$="POKE&H3C0,&H5F:POKE&H3C
1,&H5C":GOSUB172
1380 A$="POKE&H3C2,&H96:POKE&H3C
3,&HBC":GOSUB172
1390 A$="POKE&H3C4,&H1F:POKE&H3C
5,&H02":GOSUB172
1400 A$="POKE&H3C6,&H7E:POKE&H3C
7,&H96:POKE&H3C8,&HA3":GOSUB172
1410 A$="EXEC&H3C0:GOTO1":GOSUB1
72
9999 END
63949 'pclear zero
63950 POKE&H3C0,&H5F:POKE&H3C1,&
H5C
63960 POKE&H3C2,&H96:POKE&H3C3,&
HBC
63970 POKE&H3C4,&H1F:POKE&H3C5,&
H02
63980 POKE&H3C6,&H7E:POKE&H3C7,&
H96:POKE&H3C8,&HA3
63990 EXEC&H3C0:GOTO20
```

Vastly increase GET and PUT speeds

PUT Speedy GETzales to Work

By H. Allen Curtis

It is pointed out in Radio Shack's manual, *Going Ahead With Extended Color BASIC*, that in simulating motion, the GET and PUT statements can move objects faster than any other combination of ECB commands. Unfortunately, the GET/PUT movement of relatively large objects is far from being fast enough.

The goal of this article is to significantly increase the CoCo's PUT speed to permit fast and smooth GET/PUT movement of large objects. I have added two commands to CoCo's BASIC vocabulary. The two commands are new varieties of GET and PUT and will be referred to as *GET and *PUT, respectively. *PUT executes twenty times faster than PUT. This allows a BASIC programmer to generate graphics displays (stationary or animated) at machine language speeds.

The format of *GET is much like that of GET, but streamlined. Gone are the parentheses, minus sign and full graphics indicator, 'G'. The format is as follows: `*GETx1,y1,x2,y2,d`, where x1 and y1 form the x1,y1 coordinate of the upper-left corner of a rectangular area on the display; x2 and y2 form the x2,y2 coordinate of the lower-right corner of the same rectangular area and 'd' is a letter A to Z denoting the destination memory area at which a copy of the rectangular area is stored.

*GET does *not* store the rectangular display information in array form. For maximum speed, it stores the information directly in high RAM. *GET automatically reserves the required amount of protected high RAM. The destination letter is not a variable but merely a label identifying the area in which the display information is stored.

For increased speed, *GET and *PUT were designed to work in PMODE4 only. Limiting *GET and *PUT to PMODE4 is no real disadvantage because of the many techniques that have been developed to paint in a multitude of colors in PMODE4.

In PMODE4 there are 256 picture elements (pixels) in a display line. Each line is composed of 32 bytes containing eight pixels each. For increased speed, the whole byte in which the 'xi' (i=1,2) point is contained is transferred from display memory to RAM. For instance, `*GET28,5,154,25,A` would store in 21 partial lines (five through 25), each consisting of 16 bytes containing points 24 through 159.

The format of *PUT is as follows: `*PUTx1,y1,x2,y2,s,a`, where x1 and y1 form the x1,y1 coordinate of the upper-left corner of a rectangular area of the display; x2 and y2 form the x2,y2 coordinate of the lower-right corner of the same rectangular area; 's' is a letter A to Z denoting the source memory area containing the data to be displayed and 'a' is one of three possible actions — PSET, AND, OR.

The three actions are defined as follows: PSET, set each display point in the source memory area; AND, form the logical AND of each byte in the rectangular display area with each corresponding byte of the source memory area and write it on the display; and OR, form the logical OR of each byte in the rectangular display area with each corresponding byte of the source memory area and write it on the display.

The three actions are not optional. One of the three must be specified for each *PUT given. *PUT does not support PRESET and NOT actions in the interest of increased speed. Consistent with *GET, no partial bytes of *PUT are written on the display from the source memory. Every whole byte containing an x1 or x2 point is written on the display. The whole byte requirement is an important factor in the extremely fast *PUT execution rate.

Your computer's ECB ROM contains machine language routines for the execution of the statements GET and PUT. To add *GET and *PUT to CoCo's BASIC command repertoire, analogous machine language routines must be generated and stored in RAM. Listing 1, called *Star Getput*, does the required machine language routine generation and storage.

In the listing, the DATA values of lines 100 through 240 contain the 289 bytes comprising the *GET and *PUT machine language routines. Lines 10 and 20 provide a check on the accuracy of your typing of the DATA values. Lines 30 through 50 inform you when to save *Star Getput*. Lines 60 through 80 store the machine language routines. Line 90 makes sure the routines are compatible with your system. Special typing care should be taken with lines 2, 60, 70, 80 and 90; errors in lines with POKEs can cause program self-destruction.

After you have correctly typed and saved *Star Getput*, run it. The program will stop at Line 50. To resume execution, type CONT and press ENTER. When *Star Getput* has completed execution, it consists of only two lines — the REM statements of lines 1 and 2. Line 90 caused the deletion of all but those two lines. Hidden from listing view in the greatly shortened *Star Getput* are its *GET and *PUT routines, which are safely stored immediately after Line 2. Adding your own lines of programming to *Star Getput* will not overwrite the machine language routines, but merely move them to a position immediately following the last line of BASIC programming.

To activate the machine language routines, you must delete the full word REM and nothing else from Line 2. After making the deletion, save the two-line version of *Star Getput*. It will necessarily be the basis of any program you write containing *GETs and *PUTs. The two-line version of *Star Getput* must always be used with the same system configuration as the one on which it was generated.

If you used *Rainbow Check PLUS* as an aid in the accurate typing of *Star Getput*, turn off your computer now. This will erase *Rainbow Check PLUS* from your computer's high RAM which will be needed shortly. Then turn your CoCo on again and load the two-line version of *Star Getput*.

Adding a few lines of BASIC programming to the two-line *Star Getput* yields Listing 2. This program and edited versions thereof will be used to illustrate the workings of *GET and *PUT.

Run Listing 2. Lines 10 through 40 serve to draw and paint the design in the lower-left quarter of the display. The *GET statement of Line 50 stores the design. The *PUT of Line 60 retrieves the design and rapidly places it in the upper-right quarter of the display. The rapidity of *PUT execution accentuates the slowness of BASIC's PAINT command. For a much faster method of painting consistent with *PUT's speed, see my article, "Festive CoCo" [July 1986, Page 46].

To compare the speed of PUT with that of *PUT, stop the program by pressing the BREAK key and add the following lines to the program:

```
45 GOTO400
400 DIMA(308):GET(0,96)-(12
  7,191),A,G
410 PUT(128,0)-(255,95),A,P
  SET
```

Run the changed program and notice how slowly the design is formed on the upper-right quarter of the display.

Delete the entire GET statement from Line 400 and rerun the program. This time PUT forms a black rectangle in the upper quarter of the display. Without a previous GET, the 'A' array contains all zeros corresponding to black pixel codes. The program did not remember the design stored in the previous run of the program.

Surprisingly, once display information has been stored by *GET, *Star Getput* can retain this information on subsequent runs with *GET deleted. To verify this, delete Line 50 containing *GET. Also, delete lines 45, 400 and 410 to remove the remains of the GET/PUT part of the program. Then run the program to see that it accomplishes the same design transfer as before.

Next, edit Line 60 by replacing the *PUT action, PSET, with AND. Running the program again shows that AND works just as fast as PSET. If you are familiar with how AND functions with PUT, you will immediately realize that it works the same way with *PUT but much faster. Replacing AND with OR in Line 60 and running the program another time reveals that OR executes as fast as PSET or AND and otherwise functions as it does with PUT.

It is possible to *PUT a portion of the display information stored by *GET. To illustrate this, edit Line 60 by changing the y1 value from zero to 48; also change OR to AND. Then run the program. This demonstrates that when the difference between y2 and y1 in *PUT is less than a similar y-ordinate difference in *GET, *PUT will write a proportional part of the stored information on the screen. However, making the difference between x2 and x1 less in *PUT than in *GET results in a scrambling of the display information. This can be verified by changing 128 to 228 in Line 60 and running the program.

Thus far, we have only discussed a single *GET, *PUT combination. A program may have several such combinations. To show this, make the following program changes: Restore x1 and x2 in *PUT to their original values by changing 228 to 128 and 48 to zero in Line 60. Delete Line 20. Delete the first CIRCLE command in Line 30. Delete from Line 40 all but the final POKE and PAINT. Insert Line 50 as follows:

```
50 *GET16,26,199,166,Z:*PUT56,
  50,239,190,Z,AND
```

then run the program.

The next example exhibits a peculiarity of the *GET command. Add the following three lines to the program and run it:

```
70 S$="TEST":T$="THIS IS A "+
  S$
80 *GET160,80,223,100,Q
90 FORJ=1TO2000:NEXT:SCREEN0,
  1:CLS:PRINTT$
```

After the graphics display is complete, there will be a slight pause and gibberish will be printed on the text screen. However, running the program again produces the expected text screen message, THIS IS A TEST.

There is an explanation for this curious phenomenon. When *Star Getput* assigns a high RAM area to each *GET command, it overwrites whatever is in the memory area where strings are stored. *Star Getput* assigns another area for strings but does not try to recover the lost string information. However, in rerunning the program, T$ was stored in the new string storage area which was left untouched by *Star Getput* because it had no new *GET areas to assign.

The fact that the assignment of *GET memory areas will destroy string information would appear to be a serious defect. However, it is easily overcome. Merely write a line or two of dummy *GETs early in your program before forming any strings. The *GET areas will be assigned early. Later, when the corresponding real *GETs are executed, no new *GET memory areas will be assigned and there will be no string information loss.

Remember, in a earlier example it was shown that the program would run without *GETs after their display information had been stored. This *GET capability can be turned to advantage to eliminate the string loss possibility, as well as providing memory and time savings.

A final example will lead into a discussion of the means of turning *GET's memory retention to advantage: Delete lines 30, 40 and 80. Change the *GET in Line 50 to *PUT and then append a comma and PSET to the end of that changed statement. Running the program shows that the same designs are displayed as previously, even without the commands initially used to draw and paint them. Moreover, the designs are generated faster now.

What this all implies is that if the *GET information stored in high RAM can be loaded along with the BASIC program, neither *GETs nor the commands used to generate the stored display information need to be included in the BASIC program. The elimination of all these commands will make the BASIC program both shorter and faster. Furthermore, *Star Getput* will not destroy any string information because it will no longer need to assign *GET storage areas.

With disk systems it is a simple matter to load the *GET information with the BASIC program. It can be accomplished by the following steps:

- Run the BASIC program with the *GETs present to make sure all *GET storage areas are assigned and the needed information stored.
- Remove the *GET statements as well as all those commands which are now unnecessary for drawing and painting.
- Save the *GET storage information by typing and entering: SAVEM"*STORE",256*PEEK(39)+PEEK(40)+1,256*PEEK(116)+255,0.

  The address 256*PEEK(39)+PEEK(40)+1 is the lowest address in the *GET storage area. The address 256*PEEK(116)+255 is the highest RAM address.
- Now, determine precisely what the lowest *GET storage address is by typing and entering: PRINT256*PEEK(39)+PEEK(40)+1
- Add to the BASIC program 3 IFPEEK(39)=PEEK(116) AND PEEK(40)>252THENCLEAR100,1a:LOADM"*STORE" where 1a has been used to represent the address determined in Step 4. Therefore, when typing Line 3, insert the address determined in Steo 4 in place of the letters 1a. In the case of our example program, the inserted address should be 11427 or 27811 depending on whether you have a 16K or 32K byte RAM.
- Save the BASIC program with the newly added line.

If you try this six-step procedure on the example program, turn your computer off and on following the last step. This removes the *GET information from high RAM. Then type and enter RUN"EXAMPLE" where it is assumed that EXAMPLE is the filename you specified for the example program. The RUN command causes the program to be loaded; the program in turn loads the *GET storage area, and then executes. If the program is stopped and rerun, it will not go through the now unnecessary process of loading the *GET display information.

With cassette-based systems it is less straightforward to load *GET information along with the BASIC program. It can be readily accomplished with the aid of a machine language program such as *Link* which combines program files — BASIC and data — on tape to allow sequential loading with a single CLOADM command. To learn about *Link*, see my article, "Link," which appeared in the January 1985 issue [Page 58].

In using *Link*, you need to know the lowest address of the *GET storage area as well as the highest RAM address. The addresses must be expressed in hexadecimal. The lowest address serves as both the entry and first address in *Link*'s address scheme. The addresses can be derived as follows: After running to completion your BASIC program with *GETs, type and enter: PRINT HEX$(256*PEEK(39)+PEEK(40)+1), HEX$(256*PEEK(116)+255). Before applying *Link*, be sure to remove from your BASIC program the *GET statements and any commands that are no longer required for drawing and painting. Then save the shortened BASIC program.

For many applications, *Star Getput* will be ideal for generating graphics and animating them smoothly and quickly. However, there may be occasions when the shapes and sizes of display objects will depend on the program's current input data. For instance, the size and orientation of wedges in pie charts will be dependent on the data being entered during the current running of a program. Pre-stored *GET display information cannot be relied on in such cases.

To increase the speed of graphics generation in programs of the foregoing type, I developed the program, *Star Paint*, in the previously mentioned article, "Festive CoCo." *Star Paint* adds the command *PAINT to CoCo's repertoire of BASIC commands. *PAINT not only colors objects at speeds consistent with *PUT's execution rate, but also conveniently colors objects in a multitude of hues.

The programs, *Star Getput* and *Star Paint* cannot be employed together. Listing 3 was written to remedy this situation. With this program, called *Star Getputpaint*, you can make full use of all three commands, *GET, *PUT and *PAINT, in your BASIC programs.

The purpose of Listing 3 is to produce a two-line version of *Star Getputpaint*, the analogue of the *Star Getput* two-liner.

If there are any remains of the two-line version of *Star Getput* in your computer (such as stored *GET information), turn the computer off and on again before typing Listing 3. To shorten your typing chore, you may want to load Listing 1 and take advantage of the close similarity of lines 1 through 240 in listings 1 and 3.

After you have correctly typed Listing 3, save and run it. When the program has completed execution, it will be in two-line form. Without changing the rest of Line 2, delete the word REM. Then save the two-liner.

Adding a few lines to the two-line version of *Star Getputpaint* yields Listing 4. This program illustrates the combined use of *GET, *PUT and *PAINT. It executes much like Listing 2 but faster.

The use of *Star Getput* or *Star Getputpaint*, places a small restriction on cassette-based systems: The functions USR8 and USR9 must not be employed in programs containing *GET, *PUT or *PAINT.

In conclusion, *GET going, and *PUT your new graphics capabilities to work.

(You may direct questions about this program to Mr. Curtis at 172 Dennis Drive, Williamsburg, VA 23185, 804-229-7086. Please enclose an SASE when writing.) □

```
70 ........32
130 .......42
200 .......90
END ......46
```

Listing 1: GETPUT1

```
1 REM *** STAR GETPUT ***
      BY H. ALLEN CURTIS
      COPYRIGHT (C) 1985
2 REMPOKE334,158:POKE335,27:POKE
336,110:POKE337,26:POKE401,126:P
OKE402,1:POKE403,78
10 FORI=0TO288:READD$:D=VAL("&H"
+D$):C=C+D:NEXT:CLS
20 IFC<>33110THENPRINT"DATA ERRO
R":STOP
30 PRINT@162,"IF YOU HAVE NOT AL
READY SAVED   STAR GETPUT, DO SO
 NOW.
40 PRINT:PRINT"  IF YOU HAVE SAV
ED STAR GETPUT,  TYPE CONT AND P
RESS ENTER."
50 STOP
60 X=256*PEEK(27)+PEEK(28)+289:A
=INT(X/256):B=X-256*A
70 POKE474,A:POKE475,B:POKE27,PE
EK(474):POKE28,PEEK(475):CLEAR
80 X=256*PEEK(27)+PEEK(28):M=X-2
89:FORI=M TOM+288:READD$:D=VAL("
&H"+D$):POKEI,D:NEXT
90 FORJ=0TO2:POKEM+4+J,PEEK(401+
J):NEXT:DEL10-
100 DATA 81,AD,27,3,7E,C2,4D,9D,
9F,81,C4,27,4,81,C5,26,F3,34,2
110 DATA 86,4,97,7C,9D,9F,BD,B7,
3D,34,10,BD,B2,6D,A,7C,26,F4,97
120 DATA 7C,9D,9F,E6,67,54,54,54
,E7,67,EC,62,54,54,54,E0,67,2B
130 DATA 3D,5C,DD,41,CC,20,20,D0
,42,D7,50,E6,65,3D,E3,66,D3,BA
140 DATA 1F,2,E6,61,E1,65,25,24,
C1,BF,22,20,E0,65,5C,D7,43,96,42
150 DATA 3D,DD,44,9E,27,30,1,A6,
80,91,7C,27,6C,84,C0,81,40,26,B
160 DATA EC,81,30,8B,20,EE,32,69
,7E,B4,4A,A6,68,81,C4,26,F5,10
170 DATA DF,7D,DC,21,93,7D,DD,7D
,DC,27,93,44,83,0,3,DD,27,DD,23
180 DATA DC,21,93,44,83,0,3,DD,2
1,93,7D,1F,3,9E,7D,DD,7D,35,4,E7
190 DATA C0,30,1F,8C,0,0,26,F5,1
0,DE,7D,DE,27,33,41,96,7C,A7,C0
200 DATA 9E,44,AF,C1,D6,42,A6,A0
,A7,C0,30,1F,5A,26,F7,8C,0,0,27
210 DATA 48,96,50,31,A6,20,EA,1F
,13,A6,68,81,C4,26,A,AE,C1,9C,44
,25,92,9E,44,20,D8
```

Listing 2: GETPUT2

```
1 REM *** STAR GETPUT ***
      BY H. ALLEN CURTIS
      COPYRIGHT (C) 1985
2 POKE334,158:POKE335,27:POKE336
,110:POKE337,26:POKE401,126:POKE
402,1:POKE403,78
10 PMODE4,1:PCLS1:COLOR0,1:SCREE
N1,1
20 LINE(0,96)-(127,191),PSET,B
30 CIRCLE(64,144),40:CIRCLE(128,
96),70
40 POKE178,1:PAINT(64,144),,0:PO
KE178,5:PAINT(120,90),,0:POKE178
,139:PAINT(120,100),,0
50 *GET0,96,127,191,C
60 *PUT128,0,255,95,C,PSET
500 GOTO500
```

```
70 .......116
130 .......41
200 .......89
260 .......44
350 ......169
410 ........4
END .....114
```

Listing 3: GETPUT3

```
1 REM *** STAR GETPUTPAINT ***
        BY H. ALLEN CURTIS
        COPYRIGHT (C) 1985
2 REMPOKE334,158:POKE335,27:POKE
336,110:POKE337,26:POKE401,126:P
OKE402,1:POKE403,78
10 FORI=0TO698:READD$:D=VAL("&H"
+D$):C=C+D:NEXT:CLS
20 IFC<>76450THENPRINT"DATA ERRO
R":STOP
30 PRINT@162,"IF YOU HAVE NOT AL
READY SAVED   STAR GETPUTPAINT,
DO SO NOW.
40 PRINT:PRINT"  OTHERWISE, TYPE
 CONT AND          PRESS ENTER."
50 STOP
60 X=256*PEEK(27)+PEEK(28)+699:A
=INT(X/256):B=X-256*A
70 POKE474,A:POKE475,B:POKE27,PE
EK(474):POKE28,PEEK(475):CLEAR
80 X=256*PEEK(27)+PEEK(28):M=X-6
99:FORI=M TOM+698:READD$:D=VAL("
&H"+D$):POKEI,D:NEXT
90 FORJ=0TO2:POKEM+7+J,PEEK(401+
J):NEXT:DEL10-
100 DATA 16,1,1B,81,AD,27,3,7E,C
2,4D,9D,9F,81,C4,27,4,81,C5.26,E
C,34,2
110 DATA 86,4,97,7C,9D,9F,BD,B7,
3D,34,10,BD,B2,6D,A,7C,26,F4,97
120 DATA 7C,9D,9F,E6,67,54,54,54
,E7,67,EC,62,54,54,54,E0,67,2B
130 DATA 3D,5C,DD,41,CC,20,20,D0
,42,D7,50,E6,65,3D,E3,66,D3,BA
140 DATA 1F,2,E6,61,E1,65,25,24,
C1,BF,22,20,E0,65,5C,D7,43,96,42
150 DATA 3D,DD,44,9E,27,30,1,A6,
80,91,7C,27,6C,84,C0,81,40,26,B
160 DATA EC,81,30,8B,20,EE,32,69
,7E,B4,4A,A6,68,81,C4,26,F5,10
170 DATA DF,7D,DC,21,93,7D,DD,7D
,DC,27,93,44,83,0,3,DD,27,DD,23
180 DATA DC,21,93,44,83,0,3,DD,2
1,93,7D,1F,3,9E,7D,DD,7D,35,4,E7
190 DATA C0,30,1F,8C,0,0,26,F5,1
0,DE,7D,DE,27,33,41,96,7C,A7,C0
200 DATA 9E,44,AF,C1,D6,42,A6,A0
,A7,C0,30,1F,5A,26,F7,8C,0,0,27
210 DATA 48,96,50,31,A6,20,EA,1F
,13,A6,68,81,C4,26,A,AE,C1,9C,44
,25,92,9E,44,20,D8
220 DATA 9E,44,33,42,9D,9F,81,B0
,27,7,81,B1,27,6,86,A6,8C,86,A4
230 DATA 8C,86,AA,A7,8C,4,D6,42,
A6,A4,A6,C0,A7,A0,5A,26,F7,A,43
240 DATA 27,6,96,50,31,A6,20,EB,
9D,9F,32,6B,39
250 DATA 81,C3,27,25,8B,79,81,AA
,27
260 DATA 7,44,81,55,10,26,FE,DE,
97,50,9E,BA,33,89,18,0,DF,51,9D,
9F,A6
270 DATA 84,98,50,A7,80,9C,51,26
,F6,35,90,86,3,97,7C,9D,9F,BD
280 DATA B7,3D,34,10,32,61,BD,B2
,6D,A,7C,26,F2,BD,B7,3D,9F,42
290 DATA 35,54,D7,42,D7,45,1F,10
,C6,20,3D,D3,BA,1F,3,1F,10,54
300 DATA 54,54,8D,2B,F,44,1F,32,
8D,11,3,7C,D6,50,8D,1F,33,A8,E0
310 DATA D6,43,D7,45,C6,FF,D7,44
,1F,31,DC,BA,C3,17,E1,DD,7D,9C
320 DATA 7D,25,1,39,1F,30,93,BA,
2A,9,1F,98,DD,50,5A,4C,DD,52,39
330 DATA D6,51,3A,A6,84,5C,D1,52
,26,59,5A,D1,53,26,17,81,FF,27
340 DATA EC,84,3,81,3,27,5E,A6,1
,81,FF,27,6,84,C0,81,C0,27,52,39
350 DATA 5A,D1,53,22,26,81,FF,27
,48,E6,1F,C1,FF,26,4,C,53,20,DA
360 DATA 84,C0,81,C0,26,8,A,51,A
,52,30,1F,20,30,A6,84,84,3,81,3
370 DATA 26,CA,20,26,81,FF,27,22
,A,51,30,1F,A6,84,81,FF,27,18,A
380 DATA 52,20,D2,5A,5A,D1,53,22
,E,81,FF,27,A,E6,1,C1,FF,26,A0,C
390 DATA 51,30,1,96,51,97,53,E6,
84,C1,FF,26,3C,D6,45,E7,84,30,1F
400 DATA A,53,2A,F0,D6,51,1F,31,
5C,3A,D7,52,E6,84,C1,FF,26,4B,D6
410 DATA 45,E7,80,C,52,C6,20,D1,
52,26,EE,33,C8,E0,D,7C,26,3,33
420 DATA C8,40,3,44,26,3,96,42,8
C,96,43,97,45,16,FF,26,E6,84,57
430 DATA 24,C9,57,24,C6,57,24,F,
57,24,C,57,24,C,57,24,9,E6,84,D4
440 DATA 45,20,9,C6,F0,8C,C6,C0,
DA,45,E4,84,E7,84,20,A7,E6,84,58
450 DATA 24,BC,58,24,B9,58,24,F,
58,24,C,58,24,C,58,24,9,E6,84,D4
460 DATA 45,20,9,C6,F,8C,C6,3,DA
,45,E4,84,E7,84,20,9A,16,FD,4B,0
,0,0
```

Listing 4: GETPUT4

```
1 REM *** STAR GETPUTPAINT ***
        BY H. ALLEN CURTIS
        COPYRIGHT (C) 1985
2 POKE334,158:POKE335,27:POKE336
,110:POKE337,26:POKE401,126:POKE
402,1:POKE403,78
10 PMODE4,1:PCLS1:COLOR0,1:SCREE
N1,1
20 LINE(0,96)-(127,191),PSET,B
30 CIRCLE(64,144),40:CIRCLE(120,
96),70
40 *PAINT84,164,170,170:*PAINT12
0,60,116,248:*PAINT114,150,170,8
5:*PAINT60,104,170,85
50 *GET0,96,127,191,C
60 *PUT128,0,255,95,C,PSET
500 GOTO500
```

GRAPHICS 16KECB

COLORS OF THE SPECTRUM

by Bill Bernico

YOU'VE probably seen ROY G.BIV mentioned before in THE RAI-NBOW. For those of you who do not know, ROY G. BIV represents the six primary and secondary colors of the spectrum that makes up the rainbow: Red, Orange, Yellow, Green, Blue, Indigo and Violet.

The following program demonstrates the combining of any two of the primary colors (red, yellow, blue) to make a secondary color (green, orange, violet). Just answer the computer's prompts.

The Listing: SPECTRUM

```
10 'SPECTRUM
20 'BY BILL BERNICO
30 '708 MICHIGAN AVE.
40 'SHEBOYGAN, WI 53081
50 '(414) 459-7350
60 'IDEA BY DAVID POLONSKY
70 '
80 R$=CHR$(191):Y$=CHR$(159):B$=
CHR$(175):BL$=CHR$(128)
90 CLS0:PRINT@71,"RED";:PRINT@76
,"YELLOW";:PRINT@84,"BLUE";
100 PRINT@102,STRING$(5,191);STR
ING$(8,159);STRING$(6,175);
110 PRINT@165,"CHOOSE ANY two OF
 THESE";:PRINT@197,"COLORS TO SE
E WHAT COLOR";:PRINT@229,"THEY M
AKE WHEN COMBINED.";
120 PRINT@323,"CHOICE 1 (R,Y,B):
";:INPUT C$(1):SOUND191,1
130 PRINT@387,"CHOICE 2 (R,Y,B):
";:INPUT C$(2):SOUND150,1
140 IF C$(1)=C$(2)THEN 90
150 IF C$(1)="R"AND C$(2)="B"THE
N C$(3)=CHR$(239):GOTO210
160 IF C$(1)="R"AND C$(2)="Y"THE
N C$(3)=CHR$(255):GOTO210
170 IF C$(1)="Y"AND C$(2)="R"THE
N C$(3)=CHR$(255):GOTO210
180 IF C$(1)="Y"AND C$(2)="B"THE
N C$(3)=CHR$(143):GOTO210
190 IF C$(1)="B"AND C$(2)="R"THE
N C$(3)=CHR$(239):GOTO210
200 IF C$(1)="B"AND C$(2)="Y"THE
N C$(3)=CHR$(143):GOTO210
210 PRINT@344,BL$+BL$+BL$+C$(3)+
C$(3)+C$(3)+C$(3)+C$(3);:PRINT@3
77,"="+BL$+C$(3)+C$(3)+C$(3)+C$(
3)+C$(3);:PRINT@408,BL$+BL$+BL$+
C$(3)+C$(3)+C$(3)+C$(3)+C$(3);
220 GOSUB 230:GOTO 90
230 PRINT@484,"HIT ANY KEY TO CO
NTINUE";:EXEC44539:RETURN
```

16K Disk

16K ECB Mod.

Use low resolution graphics to create sharp logon messages

Graphically Speaking: The Artistic BBS

By Eric Bailey

The world of telecommunications is expanding rapidly and the CoCo is growing with it. One of the things coming our way soon is telecommunicating with graphics. No one has produced a terminal program for the CoCo that can transfer high resolution graphics, yet. But low resolution graphics are possible through a modem.

I have tried to send graphics of the highest resolution for the CoCo, but it takes over five or 10 minutes for a simple picture. I have experimented to find a way to transfer the high resolution graphics, but have not found a way to make it easy to add to a BBS (Bulletin Board System). A new terminal program and BBS software would have to be written.

I decided to write a program that allows you to create and edit low resolution graphics. Then, if you have a BBS, you can use the data files to create logon messages.

Some bulletin board systems create graphics with text. They use the slashes, plus and minus signs, etc. The idea is good, and the systems using graphics seem to attract more people. I used this idea and added a little more.

For a remote terminal to see these graphics, it must be using a CoCo and the terminal program must show the character strings 128 through 255. Some of the new communications packages are in high resolution and do not show these character strings, so the graphics won't appear correctly.

My program, *LWRSEDIT*, creates the graphics with the SET and RESET commands. The save routine PEEKs each character of the screen and saves it in ASCII format. These graphics are in low resolution (64 by 32 pixels), but it is still possible to make some very nice pictures. Pictures can really add excitement to your bulletin board.

Type in the program listing and save it. When run, it asks whether you want to see a command summary or start. The command summary lists all the commands you can use while the program is running.

The program asks for a color. This color is just to start with; you may change it anytime while in the edit mode. It then asks for the name of the picture to edit. After these questions are answered, the screen turns black and there is a flashing cursor in the color you chose at the first prompt.

To move the cursor, use the four arrow keys. To make a dot the same color as the cursor, press the space bar. The color can be changed by pressing 'C'. When the cursor stops blinking, press the number of the color wanted. Use the colors listed in the main menu. These are the same as the values the CoCo uses in BASIC.

After the color has been changed, some problems may occur. When the cursor is moved over another color, the other colors flash on and off. This is to warn you that if you press the space bar (to make a dot), then all those blinking colors will change too. This is because the CoCo can only mix a color with black. This only happens in a block of four pixels. My advice is to carefully space your picture if you plan to use many colors.

For the text mode, press 'T'; the program offers text with your graphics. Use lowercase for the characters to mix with the background.

To save a picture, press 'S'. It uses the last name you used. To change the filename, use 'F'.

Finally, there is the load command; press 'L', which loads any picture already created, or it will load the first 512 characters of any data file in ASCII format.

Modifications

With a few modifications, you can change the program to work on a cassette system. Change the save and load routines starting at lines 360 and 420, respectively. Change each expression of #1 to read #-1. Line 470 needs to be changed to a REM statement. It should not be removed, because it begins a subroutine.

The following listing is a short subroutine that can be inserted in a BBS to add graphics. Change the name of the data file to your needs. A short prompt added to the login of your BBS will tell if you should send graphics to them. The question could be, "Are you using a CoCo in 8-bit mode?" This ensures people with other computers won't receive garbage characters.

If the program does not work, try changing Line 40 to 40 PRINT CHR$(A);.

```
10 OPEN "I",1, "TEST/DAT
20 FOR X = 1 TO 512
 30 INPUT #1,A
 40 POKE 1023 + X,A
 50 NEXT X : CLOSE
```

(You may direct questions about this program to the author at 2016 Vawter #4, Urbana, IL 61801, 217-384-5083. Please enclose an SASE when writing.) □

150	204
410	84
580	140
END	128

The listing: LWRSEDIT

```
1ø ' LOW-RES GRAPHICS EDITOR
     COPYRIGHT 1986
     BY ERIC BAILEY
2ø CLS:PRINTTAB(3)+STRING$(25,19
1)+STRING$(7," ")+CHR$(191)+"LOW
-RES GRAPHICS EDITOR"+CHR$(191)+
STRING$(7," ")+STRING$(25,191)
3ø PRINTTAB(8)"BY ERIC BAILEY":P
RINT
```

continued on page 38

GAME 16K DISK

Checkers with a modem offers a new twist for an old favorite

Long Distance Draughts

By Greg Miller and Erik Gavriluk

EACH day more and more CoCo users are becoming interes--ted in telecommunications and are purchasing modems to explore this exciting new world.

We are proud of our new program, McCheckers, which combines both modem programming and some of the graphics programming tricks we learned while writing McPaint. We are also very pleased to be able to share this program with a larger audience than was possible before.

The Program

McCheckers is a machine language checkers game two people can play over the modem. This means any two people having this program and a modem can play, whether they live across town, or across the country.

To make the file necessary to play McCheckers, you need to use two programs. The first, shown in Listing 1, is a BASIC program that draws the graphics checkerboard on which the game is played. Type in and RUN this program. After the display is generated, press any key to save the graphics screen. Be sure to save a copy of the BASIC program as well.

Listing 2 is a BASIC program to generate the machine language checkers game. If you get the Checksum Error message, check the data lines, because it is likely that one or more of them contain an error. Also be sure to save a copy before you run the program; an error in typing could crash the computer.

After running Listing 2, save the completed program on cassette by typing CLOADM"CHEKBRD",&H800 and press ENTER. Then type CSAVEM" CHECKERS",&HE00,&H3300,&H2600 and press ENTER. For disk, type LOADM"CHECKBRD" and press ENTER. Then type
SAVEM"CHECKERS",&HE00,&H3300,&H260 and press ENTER.

How To Play

Load the game and type EXEC. You will see a banner, along with a prompt "Originate or Answer?" The person using the answer mode on his modem should use Answer; the other person should use Originate. The person using Originate goes first.

Next, you are put into the type mode, where you can send commands to your modem (if it reponds to commands like a Hayes Smartmodem). If you have not already done so, you must now establish carrier between you and your opponent. Press BREAK to begin the game.

Both players move the white pieces on the bottom of the board. The program automatically displays the other player's pieces as black.

McCheckers is a complete implementation of checkers; the usual rules apply. Here's a brief overview:

* Pieces only move diagonally forward. A piece may be moved backward only if it is a king. A piece becomes a king when it reaches the last row of the opposing player (the top row on the sreen).

* A piece must "jump" if at all possible. (This is an official rule of checkers, but is most often ignored in casual play.)

* The game ends when one player has captured all his opponent's pieces, or when a player has no possible move, then the other player wins.

To move a piece, point the arrow to the piece you want to move, and then to the destination square. If you make an illegal move, you are told so. You can only move a piece when the arrow appears on the screen. If the arrow does not appear on your screen, it means that the other player is in the process of moving. You must wait for the arrow to appear before you can move. When it is your turn (the arrow appears on the screen), you may send a short message to the other player by pressing CLEAR and then and typing your message. Messages are displayed on the top of the screen. If you recieve a message, press the joystick button after reading the message; the other player will not be able to continue his turn until after you have done so.

At the end of a game each player is notified as to whether he won or not, and is again put into the type mode, where pressing BREAK begins a new game.

240	242
490	207
590	128
END	155

Listing 1: MCDRAW

```
1 ' BASIC PROGRAM TO DRAW
2 ' CHECKERBOARD FOR McCheckers
3 '
10 PMODE 4,1:PCLS1:SCREEN 1,1
20 DIM B(500),B2(500)
30 FOR Y=0 TO 30 STEP 6
40 LINE(0,Y)-(255,Y),PRESET
50 LINE(0,Y+1)-(255,Y+1),PRESET
60 NEXT Y
70 X1=58:Y1=45:X2=195:Y2=180
80 LINE(X1,Y1)-(X2,Y2),PRESET,B
90 LINE(X1+1,Y1+1)-(X2-1,Y2-1),P
RESET,B
100 LINE(62,48)-(191,177),PRESET
,BF
110 GET (0,40)-(13,53),B
120 FOR X=64 TO 190 STEP 32
130 FOR Y=50 TO 160 STEP 32
140 PUT (X,Y)-(X+13,Y+13),B,PSET
150 NEXT Y,X
160 FOR X=80 TO 176 STEP 32
170 FOR Y=66 TO 176 STEP 32
180 PUT(X,Y)-(X+13,Y+13),B,PSET
190 NEXT Y,X
200 FOR Y=32 TO 44
210 IF Y/2=INT(Y/2) THEN A=204 E
LSE A=51
```

```
220 LC=&HE00+Y*32
230 FOR T=0 TO 31:POKE LC+T,A:NE
XT
240 NEXT Y
250 GET(0,33)-(255,44),B
260 PUT(0,181)-(255,192),B
270 GET(0,32)-(57,44),B,G
280 GET(196,32)-(255,44),B2,G
290 FOR Y=32 TO 180 STEP 12
300 PUT(0,Y)-(57,Y+12),B,PSET
310 PUT(196,Y)-(255,Y+12),B2,PSE
T
320 NEXT Y
330 FOR Y=4 TO 26
340 LC=&HE00+Y*32
350 FOR A=7 TO 24
360 READ B:POKE LC+A,B:NEXT A
370 NEXT Y
380 A$=INKEY$:IF A$="" THEN 380
390 CLS:PRINT"SAVING..."
440 A=PEEK(&HC000)
450 IF A=68 THEN SAVEM"CHEKBORD"
,&HE00,&H25FF,&HA027:END
460 CSAVEM"CHEKBOARD",&H600,&H1D
FF,&HA027:END
470 DATA 255,252,15,255,255,192,
24,31,255,255,255,255,255,255,25
5,255,255,255
480 DATA 255,249,136,31,255,31,3
,31,255,255,255,255,255,255,255,
255,255,255
490 DATA 255,243,131,31,254,113,
134,31,255,255,255,192,255,255,2
55,255,255,255
500 DATA 255,247,135,31,252,192,
198,31,255,255,255,216,255,255,2
55,255,255,255
510 DATA 255,231,143,31,249,128,
204,31,255,255,255,152,255,255,2
55,255,255,255
520 DATA 255,237,155,24,3,0,204,
63,192,15,0,176,252,0,3,255,255,
255
530 DATA 255,205,155,3,195,5,140
,1,159,132,120,48,249,248,96,6,0
,127
540 DATA 255,217,179,14,102,12,1
2,240,48,193,204,48,243,12,55,19
2,252,63
550 DATA 255,153,179,24,54,28,25
,152,96,99,6,48,6,6,60,99,134,63
560 DATA 255,177,227,48,54,28,27
,12,96,102,6,96,198,6,56,6,6,63
570 DATA 255,49,230,48,102,63,21
9,12,192,198,12,97,140,12,48,12,
12,63
580 DATA 255,97,198,96,6,63,222,
12,195,140,0,103,12,56,48,12,0,6
3
590 DATA 255,97,198,97,6,63,222,
12,222,12,32,108,13,224,96,231,1
92,63
600 DATA 254,97,134,97,6,63,222,
13,240,12,32,120,31,0,97,224,120
 127
610 DATA 254,193,134,99,246,31,2
20,12,192,12,126,240,12,0,97,224
,12,63
620 DATA 254,192,12,99,243,28,28
,24,192,44,126,216,76,2,99,240,6
,63
630 DATA 252,192,12,98,3,1,156,2
4,192,12,64,204,12,0,195,224,6,6
3
640 DATA 253,132,12,96,193,195,2
4,24,192,204,24,198,12,12,195,23
6,6,63
650 DATA 253,135,236,49,128,126,
24,48,99,134,48,195,6,56,195,231
,12,63
660 DATA 253,135,224,31,0,0,24,4
8,62,3,224,193,131,224,199,225,2
48,63
670 DATA 252,15,224,0,4,0,0,128,
0,0,0,0,0,0,7,224,0,63
680 DATA 252,15,224,0,15,0,0,129
,0,0,0,0,0,0,7,240,0,127
690 DATA 252,15,255,192,31,255,1
92,129,128,56,2,4,8,2,7,252,0,25
5
```

201 221	216 153
205 48	219 14
208 133	223 8
212 86	END 85

Listing 2: MCLOAD

```
1 ' BASIC loader for McCheckers
2 '
3 GOTO 10
4 GOTO 20
10 CLEAR 1000:PCLEAR 8:GOTO 4
20 CLS:AD=&H2600
30 FOR T=29 TO 1 STEP -1
40 PRINT T;
50 READ A$
60 Z$=LEFT$(A$,2):A$=MID$(A$,3)
70 V=VAL("&H"+Z$):CK=CK+V
80 POKE AD,V
90 AD=AD+1:IF A$<>"" THEN 60
100 NEXT T
110 PRINT:PRINT
120 IF CK<>285767 THEN PRINT"CHE
CKSUM ERROR" ELSE PRINT"DATA COR
RECT"
130 END
200 DATA BDA9287F09867FFF408E2C1
6BDB99CBDA1765F814F2706814126F4C
6FFF72C16F72C06BDA9288634B7FF038
E2C6ABDB99CBD2B6D8635B7FF03B62C1
6B72C06BD2FC4C6408E2B7D108E2BBDA
680A7A05A26F9CC0000FD2C14BD29C7A
6842603BD297FFC2C144C810826024F5
CFD2C14C10826E5BD
201 DATA 2B1A8E290FC60C3404EC81F
D2C144FBD295C6AE426F3C60CE7E4EC8
1FD2C148601BD295C6AE426F23261732
C067D2C061026008D8601C603FD2C00B
D2A7F10270605BD2AB3102705FE7D315
82AFB7F3158B6315B812126037E2E25C
C0707B0315DF0315EFD2C14BD29C7A68
46F84B72C0FBD297F
202 DATA CC0707B0315FF03160FD2C1
4BD29C7B62C15810726058603B72C0FB
62C0FA7844ABD295CFC31614D102BFF8
A3406CC0707A0E0E0E0FD2C14BD29C76
F84BD297F7D31631027FF6F732C067E2
69ECC0204FD2C00BD2A7F1027059FBD2
AB310270598BD274F7E269E8E2764BF2
ECC8E2765BF2ECA8E
203 DATA 2DF1BF2FBC7E2EDD39FC2FB
AFD2C14BD29927D2C12102B0645FC2C1
2FD2C14FD2C0DFD2C07BD29C7A684102
706342BC38101102706308103102706 2
AB72C0FBF2C108E27A7BF2ECA7E2EDDF
C2FBAFD2C14BD29927D2C12102B0603B
E2C12BF2C14BF2C0BBD29C7A68410260
5FDFC2C07FD2C14BD
204 DATA 29C75FA68481042702C6FFF
72C027F2C0386FFB72C09B72C0AB62C0
7B02C0B4D2A01408101276F810210220
5BDB62C08B02C0C4D2A0140810210220
5ADB62C07B02C0BF62C08F02C0C2A087
D2C0227037E2DCA340686016DE42A028
6FFC6016D612A02C6FFEDE4B62C07A0E
0B72C14F62C08E0E0
205 DATA F72C15BD29C7A6848103270
68101102605646F84BD297FBE2C14BF2
C0986FFB72C03BD29D526077D2C03102
7055CB62C08B02C0C2A15FC2C07FD2C1
4BD29C7A6848101270681021023053BF
C2C0B5D2616FC2C07FD2C14BD29C7A68
4810324028B02A784B72C0FBE2C0BBF2
C14BE2C106F84BD29
206 DATA C7B62C0FA7844ABD295CFC2
C0DFD2C14BD297F7F290B7D2C032717F
C2C0BFD2C14CC0707FD2C04BD29EB260
6732C0673290BBE2C07BF2905BE2C0BB
F2907BE2C09BF29098E2905C607BD2FC
B3900000000000000000000010003000
50007000001020104010601010203020
50207020005020504
207 DATA 0506050106030605060706 0
0070207040706078E1468F62C14583AB
62C1527B8308902004A26F9392E822E6
A2EB22E9A347634028DDD108E2954A6E
04810AEA6C60C3404ECA1ED843088206
AE426F5326135F634768DBC4FC60CA78
4A7013088205A26F635F6FC2C1481402
52581C02421C13325
208 DATA 1DC1AF24198040C03344444
444B72C124FC1102505C0104C20F7B72
C133986FFB72C12B72C1339F62C15585
858FB2C148E2BBD3A398E0000BF2C048
602B72C008604B72C01BD2B415D262BB
D29C7A6848103251086FFC6018D2C261
D8601C6018D2426158601C6FF8D1C260
D86FFC6FF8D142605
209 DATA 20CF1CFB391A04393510BF2
C141A04390000FD2A24BE2C143410ABE
4810722E8B72C14EB61C10722DFF72C1
5BD29C7A6848101270481032 6CF86027
D2A242A0286FEC6027D2A252A02C6FEA
BE4EB61810722B5C10722B1FD2C14BD2
9C7A6841026FFA53510BF2C141CFB39C
C00003406ECE4FD2C
210 DATA 14BD29C7A684B12C00271BB
12C012716ECE44C8108250B4F5CC1082
50532621A0439EDE420D632621CFB39C
C0000FD2C04BD2B415D2658BD29C7FC2
C14FD2BFEA684B72BFD8102271A7A2C1
47C2C15BD29C7A68427367C2C147C2C1
4BD29C7A6842729B62BFD81012720FC2
BFEFD2C147A2C147A
211 DATA 2C15BD29C7A684270F7C2C1
47C2C14BD29C7A684270220A51CFB391
A0439B7FFC0B7FFC3B7FFC586FFB7FF2
28E0E0034108EFFC686061F89686169E
459A7854A2AF4326239FC2C04FD2C144
C810826074F5CC108260139FD2C04BD2
9C7A68427E481FF27E0B12C002705B12
C0126D65F39AD9FA0
212 DATA 00270781032705BD309E20F
139FF01FF01FF01FF0101FF01FF01FF0
1FFFF01FF01FF01FF0100FF00FF00FF0
0FFFF00FF00FF00FF0002FF02FF02FF0
2FFFF02FF02FF02FF0202FF02FF02FF0
2FF00000000000000000000000000000
00000000000000000000000000000000
0000000000000000
213 DATA 0000000000000000000000
```

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000020202020202020202
04D63436865636B65727320312E300D0
D42792047726567204D696C6C6572206
16E64204572696B2047617672696C756
B0D3C4F3E72696769
214 DATA 6E617465206F72203C413E6
E737765723F20000D202020202020202
020202D54595045204D4F44452D0D202
0202020505245535353203C425245414B3
E20544F20424547494E0D00202020202
02D2D594F552057494E2054484953204
7414D452D2D0D008E2C9EBD2CEA7E262
620202020202D2D59
215 DATA 4F55204C4F5345205448495
32047414D452D2D0D008E2CC3BD2CEA7
E26263410BDA9288E05009F883510BDB
99C396060606060606060606060494E564
14C4944604D4F5645606060606060606
0606060606060606060604E4F60504
94543456054484552456060606060606
06060606060606054
216 DATA 484154604953604E4F54605
94F55526050494543456060606060606
06060606043414E6754604D4F5645605
44F6054484552456060606060606060604
F4E4C59604B494E47536043414E604D4
F5645604241434B57415244536060605
94F556048415645604160564l4C49446
04A554D5060544F60
217 DATA 4D414B4560608E2CFA108E2
D1A108E2D3A108E2D5A108E2D7A108E2
D9ABD325D7E274F454E544552204D534
72C203332204348415253204D41582E0
D00B6FF00854027F9BDA9288E2DD6BDB
99CBDA3908E02DDC6217F02FEBD2FCBB
D2B1A7D31582AFB7F3158B6315D81062
6F17E2EDD007F2E24
218 DATA 8E315DC6207D2E242604A68
426078660B72E2420128160250480602
00A81402406812025028B40A7805A26D
98E315DBD325D8606B702DD8E02DDC60
2BD2FCB7E26A803001FE03FF03FF07FF
87FF87FF87FF83FF03FF01FE00300030
01FE0303020106018400840086018201
030301FE003000300
219 DATA 1FE03FF03CF078787338733
878783CF03FF01FE0030003001FE0303
0231067984CC84CC86798231030301FE
0030000000000BE2F22BD2F47BD2FBE2
7FB6E9F2ECABD2FA3BD2F24BF2F22BD2
F61BD2FBE27E086FDB7FF02B6FF00844
02607BD2F476E9F2FBCBD2FA3BD2F24B
C2F2227DD3410BE2F
220 DATA 22BD2F47AD9F2ECC3510BF2
F22BD2F6120C70000F62FBBC1B62505C
6B6F72FBB8E0E00F62FBA5454543AF62
FBB2601393088205A26F739C609108E2
F58A6A0A7843088205A26F6390000000
0000000000108E2F913410C609CE2F5
8A684A7C0A6A0A484A7843088205A26F
03510108E2F9AC609
221 DATA A6A0A884A7843088205A26F
4399F8F878381838180F060504844424
452790FAD9FA00AB6015AC6043DF72FB
AB6015BC6033DF72FBB3900000000B6F
F008401398E30C7BF010D3934771A501
08E304E10BF31598600B7FF20B6FF228
50126F98602B7FF20B6FF22850127F91
08E00003414A68031
222 DATA A65A26F93420C63C5A26FDA
662BD309EE662AE63A680BD309E5A26F
8A6E4BD309EA661BD309E8602B7FF20C
60AF73155BD3055C602F73155B731568
1151027FFC532653577B631568106102
6FF855F5A26FDBD309E39326535777E2
FCB34751A508608B73157F631558E000
0B6FF2244240F301F
223 DATA 26F65A26F3357532626E9F3
1595F8D168D10B6FF2244567A315726F
41F988D0235F58D008D003402B6315C2
1FE4A26FB358234771A5034024FB7FF2
08DE4C60864E425024F8C8602B7FF208
DD55A26F032618602B7FF208DC935F7B
6FF02B6FF224424013B8E314DBF31598
600B7FF20B6FF2244
224 DATA 24FA8602B7FF20BD3055B73
15B1F898E315DBD3055A7805A26F8860
2B7FF20BD3055B73156BD3055F631561
E89FD31538E315DF6315B108E0000A68
031A65A26F910BC3153261F8606BD309
EC605F73155BD3055C602F731557F315
88106260D86FFB731583B8615BD309E2
09A8602B7FF203B00
225 DATA 00020000000000005900000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000
226 DATA 00000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000
227 DATA 00000000000000000000000
00000000000000000000000000000000
00000000000B6FF00850127F98636B7F
F03B7FF01B7FFC3B7FFC586FFB7FF22B
D32938634B7FF014CB7FF03B6FF02B6F
F001CAFB6FF00850127F939C620108E0
E00A680A7A05A26F91A50B6FF0084012
60BC6208E0E006F80
228 DATA 5A26FB39B6FF027DFF032AF
BB7FFC2B7FFC47FFF22C631B6FF007DF
F012AFB34763576125A26F0B7FFC3B7F
FC586FFB7FF227E32A2FF00002600
```

CORRECTION

The Adventure Processor

Lines 630 and 640 were inadvertently left out. These lines create strings consisting of the first four characters of each verb and object keyword. For proper operation, each verb and keyword must contain at least four characters. The program can be corrected by making the following changes:-

```
630 A$="NV="+STR&(NV)+":FORI=1TO
NV":GOSUB172
631 A$="IFLEN(V$(I))<4 THEN V$(I
)=V$(I)+CHR$(32):GOTO"+STR$(LN):
GOSUB172
632 A$="V1$=V1$+LEFT$(V$(I),4):N
EXT":GOSUB172
640 A$="FORI=1TO O":GOSUB172
641 A$="IFLEN(O$(I,2))<4 THENO$(
I,2)=O$(1,2)+CHR$(32):GOTO"+STR$
(LN):GOSUB172
642 A$="N1$=N1$+LEFT$(O$(I,2),4)
:NEXT":GOSUB172
```

A previously generated Adventure can be corrected by adding spaces to verb strings consisting of less than four characters and appear in data statements must be enclosed with quotes and additional spaces added to bring that total number of characters to four.

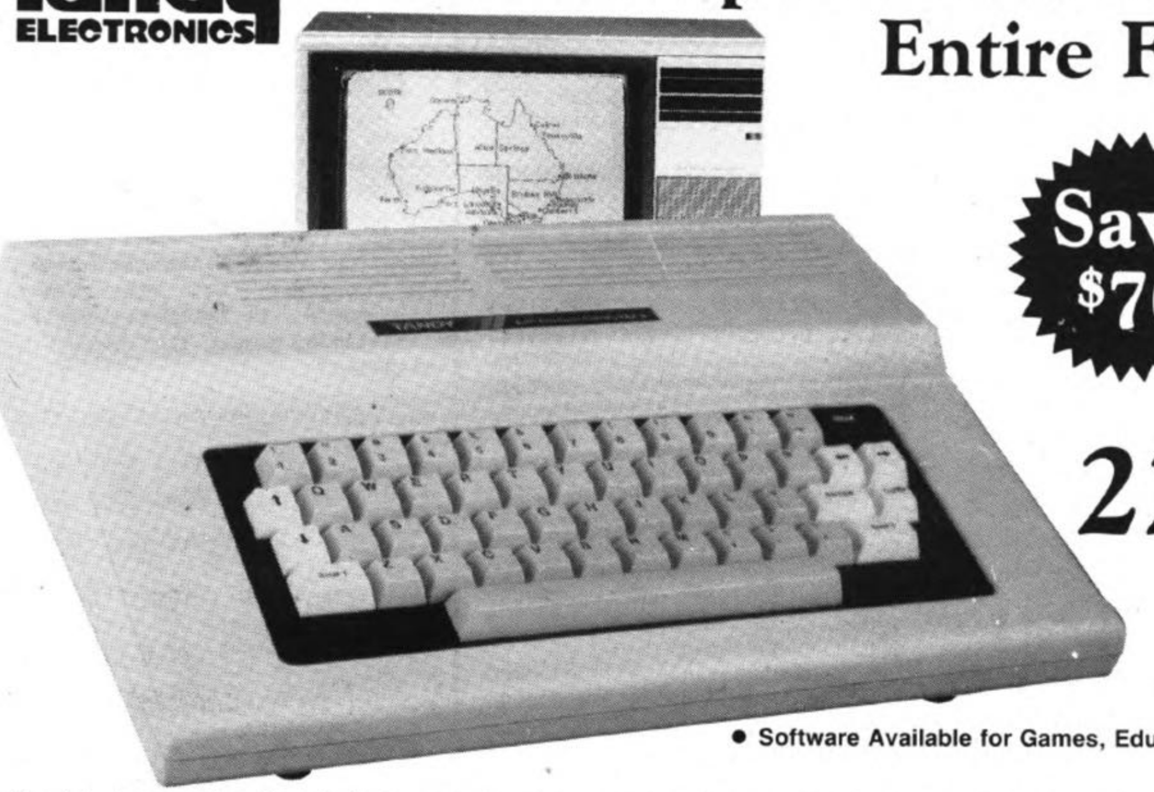
NOTICE TO AUTHORS

Below is a list of words that are reserved by Disk BASIC. We ask that you refrain from using these words as they muck up the works - to use the correct techinal explanation!

BACK UP	TO	CLOSE
COPY	CVN	DIR
DRIVE	DSKINI	DSKI$
DSKO$	EOF	FIELD
AS	FILES	FREE
GET	PUT	KILL
LOAD	LOADM	LOC
LOF	LSET	MERGE
MKN$	OPEN	RENAME
RSET	SAVE	SAVEM
UNLOAD	VERIFY	WRITE

Tandy ELECTRONICS

Color Computer Fun for The Entire Family



Save $70

Reg 239.95

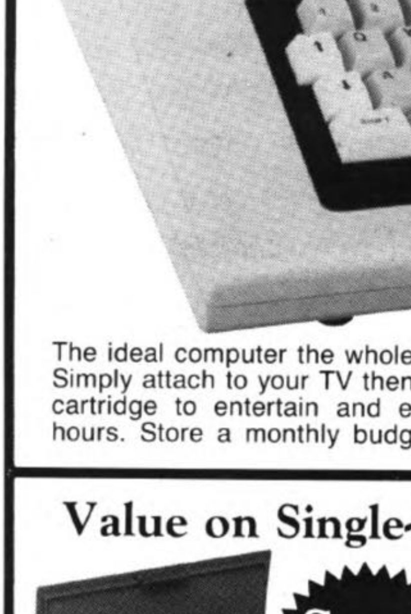
229⁹⁵

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

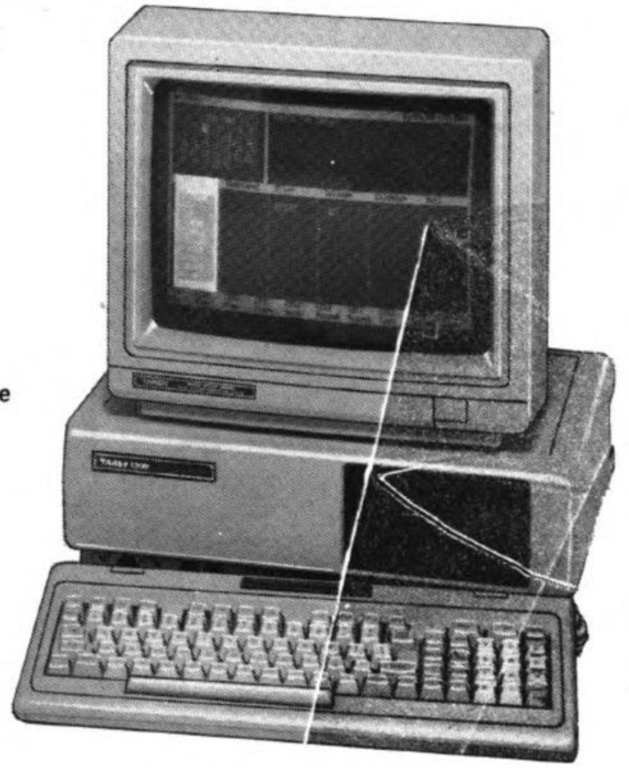
Save $50

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



Save $899

Reg 2598.00

$1699

- 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

64K Disk

CoBBS Message Editor

A useful update for the CoBBS system

By Richard Duncan

The *CoBBS* system was presented in November 1985, and quite a few BBSs use it. There are editors for just about every file in use on the system, except for handling the message base for certain problems or desires. The *CoBBS Message Editor* is written to handle some of the problems that occasionally occur during normal BBS operation.

SMH/EDI goes into the message base and allows the user to modify information contained in the header of that message (e.g., menu number it was posted to, active/deleted, private/public, or the to/from/subject information). In addition, there is a renumbering routine that allows a sequential assignment of message numbers starting at whatever the SysOp designates.

The routine can be used offline and does not require that the driver be loaded. There is no error trapping or checking for carrier detect. This editor would normally only be used by the SysOp and would not require these features. Error trapping and carrier check could easily be added to the routine if required.

Booting Up

On running *SMH* the routine checks the drives assigned to three files: MENU/SYS, HDR/SYS and MSG/SYS. These values will not be correct if the editor is run before the BBS system itself has been booted. When this occurs, you are asked which drive each file is on. Once this is taken care of, the system loads in the name of the various menus you have on the board and then goes directly into the editor.

A menu is displayed before any message information. This is the help menu and can be seen or redisplayed by using the H command. There are 16 commands available.

Move Options

There are five commands that affect movement around the message base. Two commands, + and –, move forward or backward respectively through the message base. The GOTO command locates a particular message number within the base if available. ALL messages are displayed whether they are active or deleted.

Initially, every message is displayed when entering the editor. If you only want to view one particular area (menu number), use the SET MENU command. When prompted, enter the menu number you want to look at. Return to viewing all messages by using this command and responding with ALL instead of a menu number.

There is a search feature that thumbs through the header information for a match. The SEARCH option prompts for the string you want a match to. This is a global search that checks the to, from and subject at one time. The search starts from the current location to the end of the file. If a complete search of the message base is desired, go to the first message in the base before starting the search.

Toggles

Four toggle commands are available. The PUBLIC/PRIVT toggle determines whether the message is public or private. A message may be deleted or reactivated by the use of the K and A commands. The KILL command is used to delete the current message and the ACTIVATE command allows a deleted message to be reinstated. A message that has been received can be changed back to waiting on the user to call again by the TOGGLE RCVD command.

Modifiers

Occasionally, there is the need to change the header information of who the message is from, to or its subject. The CHANGE HEADER command allows this. Each particular part of the address is displayed showing what the original information is and requesting the change. If just ENTER is pressed, the current information will be retained, while entering anything on this line causes that part of the address to be changed.

A message can be re-posted to a different menu number by use of the BOARD POST command. Type the new menu number and press ENTER.

There is always a difference in opinion about how a message board should be operated. *CoBBS* was originally set up to sequentially count the number of messages entered from day one. Some operators like to limit or change the overall message count from time to time. Using the RENUMBER command allows the SysOp to change each message in a sequential order. It is best to do this when there are no deleted messages in the base or the renumbering will appear broken to a user and will disappear at the next message purge.

Other Commands

Once the message header appears, the SysOp can then view the message text by using the MESSAGE command. As the text appears, pressing any key stops the scroll and pressing any key again restarts the viewing. Pressing the 'S' key stops the display and returns to the command menu. Scroll control is only available from the keyboard.

The complete message may be dumped at one time to either the screen or the printer with the DUMP MESSAGE command. After selecting this command, choose whether a screen print or printer dump is desired.

The QUIT command allows termination of the message editor and the option of going into BASIC or returning the BBS.

The editor can be used online with the *CoBBS* system. Again, note that there is no error trap or carrier detect while in the message editor. To modify

CoBBS for online access to this file, follow this procedure: Load *COBBS/SYS*. Retype Line 34 to read `34 LOAD"SMH/EDI",R`. Save *COBBS/SYS*.

The message editor returns to *CoBBS* by using the QUIT command and answering 'Y' to the option. There are no other modifications required. Save this editor on Drive 0 under the filename SMH/EDI.

After the modification is made to *COBBS/SYS*, the editor can be called from the BBS by using a type ″ command. No additional data is required.

The *CoBBS* software originally came out last fall in RAINBOW [November 1985, Page 135] and is available through back issues of RAINBOW and RAINBOW ON TAPE.

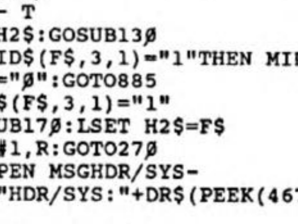
If you are operating a CoBBS system, please send me your phone number, BBS number and hours of operation because I am compiling a directory of *CoBBS* systems. In a future article I will correct some problems in the original *CoBBS* and look at some new additions. My address is 9821 Margie Circle, Little Rock, AR 72209-6521. Please enclose an SASE when writing. All letters will be answered as soon as possible. You can also find me on Delphi (username RICH4COBBS). □

100	137
215	123
300	142
360	29
470	59
590	115
720	159
840	160
END	63

The listing: SMH

```
ø 'SMH/EDI BY RICHARD DUNCAN
1ø CLEAR 5øøø
15 DIM BN$(255)
2ø DR$(ø)="ø":DR$(1)="1":DR$(2)=
"2":DR$(3)="3":NU$=CHR$(ø):MD=-1
25 CLS:PRINT"        SYSTEM MESSAGE
 EDITOR"
3ø FOR X=467ø TO 4673:IFPEEK(X)<
4 THEN 6ø
35 PRINT:PRINT"ERROR IN DRIVE NU
MBERS"
4ø INPUT"MENU/SYS DRIVE";X:POKE4
673,X
45 INPUT" HDR/SYS DRIVE";X:POKE4
67ø,X
5ø INPUT" MSG/SYS DRIVE";X:POKE4
671,X
55 GOTO65
6ø NEXTX
65 PRINT:PRINT:PRINT"WAIT....":P
RINT"GETTING MENU: "
7ø REM - MENU/SS
75 GOSUB945
8ø FOR R=1 TO K2 STEP 4
85 GET#2,R:BN$=M5$:BN=ASC(M1$)
9ø PRINTBN;TAB(1ø);M5$
95 K=INSTR(BN$,NU$):IFK=øTHEN K=
LEN(BN$)+1:BN$(BN)=LEFT$(BN$,K-1
)
1øø BN$(BN)=LEFT$(BN$,K-1):NEXT
R:BN$(255)="SYSOP MSG"
1ø5 CLOSE:GOSUB895:GOSUB92ø
11ø GET#1,1:RE=CVN(H1$):MH=RE
115 FORB=2 TO K1:GET#1,B:F$=H2$:
GOSUB13ø:IFMID$(F$,2,1)<>"1"THEN
 RS=CVN(H1$) ELSE NEXT B:RS=RE
12ø ML=RS:GET#1,1:R=1
125 GOSUB2ø5:GOTO27ø
13ø REM
135 F=ASC(F$):E=128:F$=""
14ø FOR Q=1 TO 8
145 J=INT(F/E)
15ø IF J=ø THEN F$=F$+"ø"ELSEF$=
F$+"1"
155 F=F-(E*J):E=E/2
16ø NEXT Q
165 RETURN
17ø REM
175 E=1:F=ø
18ø FOR Q=8 TO 1 STEP -1
185 IFMID$(F$,Q,1)="1"THEN F=F+E
19ø E=E*2:NEXTQ:F$=CHR$(F)
195 RETURN
2øø GOSUB295
2ø5 PRINT:PRINT
21ø IF MD=-1 THENPRINT"ALL MENUS
" ELSE PRINT"MENU:";MD;"/ ";BN$(
MD)
215 PRINT"MESSAGES";ML;"TO";MH
22ø PRINT"<+>NEXT MSG #    <G>OTO
"
225 PRINT"<->LAST MSG      <M>ESS
AGE TEXT"
23ø PRINT"<?>SEARCH        <P>UBL
IC/PRIVT"
235 PRINT"<K>ILL MSG       <D>UMP
 MESSAGE"
24ø PRINT"<A>CTIVATE MSG  <R>ENU
MBER"
245 PRINT"<B>OARD POST     <S>ET
MENU"
25ø PRINT"<H>ELP MENU      <T>OGG
LE RCVD"
255 PRINT"<C>HANGE HDR     <Q>UIT
26ø RETURN
265 GOSUB295
27ø LINEINPUT"COMMAND: ";CH$
275 IFCH$=""THEN265
28ø ON INSTR("+-GM?PKDARCQHSBT",
CH$)+1 GOTO 27ø,37ø,395,42ø,45ø,
5ø5,545,575,6øø,655,68ø,735,285,
125,795,825,865
285 CLEAR1øø:LINEINPUT"RETURN TO
 BBS? ";A$
29ø X$=LEFT$(A$,1):IFX$="Y"ORX$=
"y"THEN CLOSE:UNLOAD:LOAD"COBBS/
SYS",R ELSE END
295 REM-HDR PRINT
3øø IF MD>-1ANDASC(H8$)<>MD THEN
RETURN
3ø5 PRINT#SC:PRINT#SC
31ø IF R=1 THENPRINT"SYSTEM RECO
RD #1!"
315 Mø=CVN(H1$):KF=INSTR(H5$,NU$
):MF$=LEFT$(H5$,KF-1)
32ø F$=H2$:GOSUB13ø:IFMID$(F$,2,
1)="1"THENPRINT#SC;"-DELETED-"
325 KT=INSTR(KF+1,H5$,NU$):XX=KT
-KF-1:IFXX<1THEN MT$="":GOTO33øE
LSE MT$=MID$(H5$,KF+1,XX)
33ø KS=INSTR(KT+2,H5$,NU$):XX=KS
-KT-1:IFXX<1THEN MS$="":GOTO335E
LSE MS$=MID$(H5$,KT+1,XX):PF=ø
335 IFMID$(F$,1,1)="1"THENPRINT#
SC,"PRIVATE"
34ø IFMID$(F$,3,1)="1"THEN X$="
<RCVD>" ELSE X$=""
345 F5=1:PRINT#SC,"MESSAGE #";Mø
;" ";BN$(ASC(H8$))
35ø PRINT#SC,RIGHT$(STR$(ASC(LEF
T$(H3$,1))),2);"/";RIGHT$(STR$(A
SC(MID$(H3$,2,1))),2);"/";RIGHT$
(STR$(ASC(RIGHT$(H3$,1))),2);"
  ";STR$(ASC(LEFT$(H4$,1)));":";
355 A$=RIGHT$(STR$(ASC(RIGHT$(H4
$,1))),2):IFVAL(A$)>9THENPRINT#S
C,A$ELSEMID$(A$,1,1)="ø":PRINT#S
C,A$
36ø PRINT#SC,"FROM: ";MF$:PRINT#
SC,"  TO: ";MT$;X$:F5=1:F4=1:PRI
NT#SC,"SUBJ: ";MS$:PRINT#SC
365 RETURN
37ø REM- +
375 R=R+1:IF R>K1 THEN R=1
38ø GET#1,R:SA=ø
385 IF MD>-1 AND ASC(H8$)<>MD TH
EN 375
39ø GOTO265
395 REM- -
4øø R=R-1:IF R<1 THEN R=K1
4ø5 GET#1,R:SA=ø
41ø IF MD>-1 AND ASC(H8$)<>MD TH
EN 4ø5
415 GOTO265
42ø REM- G
425 PRINT:LINEINPUT"MESSAGE #";N
$
43ø N=VAL(N$):IF N<ML OR N>MH TH
EN PRINT"OUT OF RANGE.":GOTO27ø
435 FOR X=1 TO K1
44ø GET#1,X:IF CVN(H1$)=N THEN R
=X:GOTO265
445 NEXTX:PRINT"MSG NOT AVAILABL
E":GOTO27ø
45ø REM- M
455 GOSUB46ø:SC=ø:GOTO27ø
46ø R1=CVN(H6$):R2=CVN(H7$)
465 FOR X=R1+1 TO R2
47ø GET#2,X:A$=MG$:IFA$=STRING$(
8ø,255)THENPRINT#SC:RETURN
475 FOR Y=1 TO LEN(A$)
48ø PRINT#SC,MID$(A$,Y,1);:X$=IN
KEY$:IFX$=""THEN49ø
485 IFX$="S"ORX$="s"THEN5øøELSEI
FINKEY$=""THEN485
49ø NEXTY
495 NEXTX
5øø PRINT#SC:RETURN
5ø5 REM- ?
51ø U=R:PRINT:LINEINPUT"SEARCH S
TRING: ";S$
515 IF S$=""THEN27ø
52ø FOR X=U+1 TO K1
525 GET#1,X
53ø IFINSTR(H5$,S$)>ø THEN R=X:G
OTO265
535 NEXT X:GET#1,U
54ø PRINT"NOT FOUND.":GOTO27ø
545 REM- P
55ø F$=H2$:GOSUB13ø
555 IFMID$(F$,1,1)="1"THENMID$(F
$,1,1)="ø":GOTO565
56ø MID$(F$,1,1)="1"
565 GOSUB17ø:LSET H2$=F$
57ø PUT#1,R:PRINT:GOTO27ø
575 REM- K
58ø F$=H2$:GOSUB13ø
585 MID$(F$,2,1)="1":GOSUB17ø
59ø LSET H2$=F$:PUT#1,R
595 GOTO27ø
6øø REM- D
6ø5 PRINT:PRINT
61ø PRINTTAB(5);"1-SCREEN"
615 PRINTTAB(5);"2-PRINTER"
62ø LINEINPUT"      >";CH$
625 ONINSTR("12",CH$)+1 GOTO 27ø
,63ø,635
63ø SC=ø:GOTO64ø
635 SC=-1
64ø GOSUB295
645 GOSUB46ø
65ø GOTO27ø
655 REM- A
66ø F$=H2$:GOSUB13ø
665 MID$(F$,2,1)="ø":GOSUB17ø
```

```
67Ø LSET H2$=F$:PUT#1,R
675 GOTO27Ø
68Ø REM- R
685 PRINT:PRINT"MESSAGE RENUMBER
!"
69Ø LINEINPUT"STARTING NUMBER: "
;S$:S=VAL(S$)
695 IF S=Ø THEN 27Ø ELSE ML=S
7ØØ S=S-1
7Ø5 FOR X=2 TO K1
71Ø GET#1,X:S=S+1:LSET H1$=MKN$(
S)
715 PUT#1,X:NEXTX
72Ø GET#1,1:LSET H1$=MKN$(S):PUT
#1,1
725 MH=S
73Ø GET#1,2:GOTO125
735 REM- C
74Ø PRINT:PRINT"FROM: ";MF$
745 LINEINPUT"FROM: ";CH$
75Ø IFCH$=""THEN755ELSEMF$=CH$
755 PRINT"  TO: ";MT$
76Ø LINEINPUT"  TO: ";CH$
765 IFCH$=""THEN77ØELSEMT$=CH$
77Ø PRINT"SUBJ: ";MS$
775 LINEINPUT"SUBJ: ";CH$
78Ø IFCH$=""THEN785ELSEMS$=CH$
785 LSET H5$=MF$+CHR$(Ø)+MT$+CHR
$(Ø)+MS$+STRING$(8Ø,Ø)
79Ø SA=1:GOTO27Ø
795 REM- S
8ØØ PRINT:LINEINPUT"MENU # OR AL
L: ";CH$
8Ø5 IFCH$=""THEN27Ø
81Ø IFCH$="ALL" THEN MD=-1 ELSE
MD=VAL(CH$)
815 IF MD<Ø OR MD>255 THENPRINT"
MENU OUT OF RANGE.":MD=Ø
82Ø GOTO27Ø
825 REM- B
83Ø PRINT:PRINT"PRESENT MENU: ";
BN$(ASC(H8$))
835 LINEINPUT"POST TO #";CH$
84Ø IFCH$=""THEN27Ø
845 X=VAL(CH$):IFX<ØORX>255THENP
RINT"VALUE OUT OF RANGE":GOTO83Ø
85Ø PRINT"POSTING TO: ";BN$(X)
855 LSET H8$=CHR$(X):PUT#1,R
86Ø GOTO27Ø
865 REM- T
87Ø F$=H2$:GOSUB13Ø
875 IFMID$(F$,3,1)="1"THEN MID$(
F$,3,1)="Ø":GOTO885
88Ø MID$(F$,3,1)="1"
885 GOSUB17Ø:LSET H2$=F$
89Ø PUT#1,R:GOTO27Ø
895 '-OPEN MSGHDR/SYS-
9ØØ F$="HDR/SYS:"+DR$(PEEK(467Ø)
)
9Ø5 OPEN"D",#1,F$,11Ø
91Ø FIELD#1,5 AS H1$,1 AS H2$,3
AS H3$,2 AS H4$,8Ø AS H5$,5 AS H
6$,5 AS H7$,1 AS H8$,8 AS SP$
915 K1=LOF(1):RETURN
92Ø '-OPEN MSG/SYS-
925 F$="MSG/SYS:"+DR$(PEEK(4671)
)
93Ø OPEN"D",#2,F$,8Ø
935 FIELD#2,8Ø AS MG$
94Ø K2=LOF(2):RETURN
945 '-OPEN BOARD MENU-
95Ø FF$="MENU/SYS:"+DR$(PEEK(467
3))
955 OPEN"D",#2,FF$,25Ø
96Ø FIELD#2,1 AS M1$,1 AS M2$,1
AS M3$,1 AS M4$,16 AS M5$,23Ø AS
 M6$
965 K2=LOF(2):RETURN
```



TELECOMMUNICATIONS

16K ECB

RTTY for the Color Computer

By Marty Goodman

This article and the accompanying program provide a simple, practical means of using the Radio Shack Color Computer to send and receive RTTY information. Nonlicensed radio enthusiasts may find the article and the program of some interest, although most of the commercial international radio text transmissions are sent via microwave and satellite. Indeed, outside of RTTY encountered on the Ham bands, most RTTY transmissions in the HF bands consist of encrypted material.

What Is RTTY?

RTTY is a very old means of encoding text information for transmission over the radio. The version most commonly used encodes a 1 (mark) as a 2125-Hz tone and a 0 (space) as a 2295-Hz tone. This encoding is then used to make up characters consisting of five bits each. This five-bit code is the Baudot code, an early predecessor of the present-day, seven-bit ASCII code. Five bits allow for coding only 32 different characters, but one of those codes is a shift character, which can be used in conjunction with other characters to get a somewhat greater range of characters. Still, only uppercase letters, the numbers, and a limited number of punctuation marks are allowable.

The version of RTTY implemented here can send and receive at 45.45 Baud (60 words per minute).

This may seem slow to those accustomed to 300, 1200 or 2400 Baud modems, but it is faster than many folks can type. More important, it is sufficiently slow that it results in more reliable transmission over radio than even machine generated and received Morse code. Thus, while the RTTY protocol is quite old and slow, and while this program is a fairly limited implementation of it, it still has real practical value to radio amateurs.

Program Characteristics

The program to be presented is a simple one. Many desirable features, such as backspace in the transmit buffer, saves to disk, and transmitting of a previously prepared text file, have not be implemented. Macro 80C Source code for the program will be available in the Data Communications area on Delphi, so that assembly language programmers may enhance it as they please. The program supports only the slowest Baud rate for RTTY transmission. However, this program allows you to type at the keyboard and have RTTY tones transmitted out the gray (Aux) plug line of the cassette port on the CoCo. RTTY tones are received from a high frequency receiver into the zero crossing detector of the cassette port (black, Earplug).

When an RTTY signal is received, owners of older shortwave receivers need to adjust their variable BFO until the tuning indicator on the RTTY program's screen shows they have adjusted the tones to the right frequency. Reception in such cases is greatly enhanced if a narrow band audio pass filter is used to filter the output of your HF receiver. Such a pass filter should be constructed to pass very narrow bands centered on the two tone frequencies, 2125 and 2295 Hz. Some HF receivers come with a built-in RTTY filter centered on 2200 Hz. If such a feature is available, it should be used. Owners of newer digital receiver equipment will not have a variable BFO, but may be able to get by using an IF shift control that is often provided, combined with a 200-Hz IF filter if that is available.

Hardware Setup

The output of your receiver is fed into the cassette input of the CoCo. As noted, a narrow pass audio filter greatly improves performance. If your audio

```
911 DATA2ØØ3CCØØØFD15826Ø63D3DAC
8B2Ø138EØ41ØA68584BFA7859658D758
E686CA4Ø,3334
912 DATAE78617Ø12639ACØ1ACØ186Ø1
5CC16Ø25Ø4A1Ø12ØØ5B5FF2Ø27F25CC1
6Ø25Ø4A1,2918
913 DATAØ12ØØ5B5FF2Ø26F2398EØ4Ø5
CE12D2A6CØA78Ø8CØ4ØD25F7CC343CF7
FF21B7FF,3811
914 DATAØ186Ø2B7FF2ØØF59ØF5A9E46
9C442724CCØF61DD5186Ø5975BE68Ø9F
46D75C5F,3331
915 DATA8D54AC94CCØØØ85A26FDØ45C
8D48ØA5B26F2EC9B3DAC8BCC15ACDD51
538D3796,3719
916 DATA5326C7CC343CB7FF21F7FFØ1
8EØ4Ø5CE12B7A6CØA78Ø8CØ4ØD25F739
B6FF2Ø8A,3926
917 DATAØ2B7FF2Ø862A975Ø39B6FF2Ø
84FDB7FF2Ø8627975Ø3924Ø48DE22ØØ4
8DEB2ØØØ,3572
918 DATA8Ø1D4A26FDB6FF2Ø88FCB7FF
2ØD65Ø4FD359DD59935124Ø88DØF965Ø
8Ø232ØE2,3911
919 DATADD5939AC943DACØ139964226
7CDC4ØCØ37498AØ124Ø22ØØ3CCFEF8DD
4Ø8EØ152,3388
92Ø DATA3AB7FFØ2B6FFØØ8A8Ø1F89E8
84E48421FEA78486Ø7DD429E449C4626
C69C4826,4Ø61
921 DATAC48E12DA9F469F489F443996
42263ADC4ØCØ37498AØ124Ø22ØØ3CCFE
F8DD4Ø8E,3579
922 DATAØ1523AB7FFØ2B6FFØØ8A8Ø1F
89E884E48426BEA78496414C27Ø88B37
974121FE,3738
```

```
923 DATA2Ø8186Ø7DD4216FF7A4A9742
D641CBØ8D741Ø4432475867FB7FFØ2B6
FFØØ844Ø,36Ø7
924 DATA27Ø58E12422ØØ58E127A2ØØØ
A685E6852B289E44984AD74A846Ø26Ø9
C41FE78Ø,2968
925 DATA3D21FE2Ø12C54Ø26Ø4861B2Ø
Ø4861F2ØØØC41FED81A1Ø19F44399E44
5C26ØE96,2654
926 DATA4A844Ø974ACCØØØ4E78Ø3D2Ø
EA5C26ØBCCØ8Ø2A78ØED81A18B2ØDC5C
26Ø4Ø353,3177
927 DATA2ØØ55C273AA1843DA18B39D6
4B273ECØØ4D74B8EØ5EØ3AEC84ED88EØ
CC6Ø6ØED,384Ø
928 DATA81EC84ED88EØCC6Ø6ØED84D6
4B27Ø43DAC84398EØ5EØ9F4C86BFA784
21FE3986,4321
929 DATA6ØA79FØØ4C9F46CCØØ2ØD74B
399E489C4426Ø7CCØØØC5A26FD39A68Ø
9F489E4E,3224
93Ø DATAA6862B129E4CA78Ø8CØ6ØØ24
139F4C86BFA784AC8B394C26ØF12866Ø
A79FØØ4C,3Ø97
931 DATA862Ø974B3D3Ø84394C26Ø98E
12229F4EA1Ø12ØØB4C26Ø78E12Ø29F4E
2ØØ13D3D,2124
932 DATA12398Ø45FF416Ø5349558Ø44
524A4E46434B545A4C5748595Ø514F42
47FE4D58,2865
933 DATA56FD8Ø73FF6D6Ø5E78778Ø64
74676C617A687562697263767Ø71797F
66FE6E6F,3954
934 DATA7BFD8Ø43594E49414D5A5446
4B4F525C4C5856574A455Ø475E535D55
51258Ø8Ø,2885
```

```
935 DATA8ØFF363733212A3Ø35272638
2E3E2C233C3DFEFDFC8Ø8Ø8Ø8Ø8Ø43
594E4941,3Ø59
936 DATA4D5A54464B4F525C4C485657
4A455Ø475E535D5551258Ø8Ø8ØFF8Ø2D
3134298Ø,2819
937 DATA3A2B2F328Ø8Ø8Ø8Ø8Ø39FEFD
FC8Ø8Ø8Ø8Ø5254545960524543454 9
56456Ø6Ø,3436
938 DATA6Ø6Ø6Ø4D41524B6D6D7E7C6D
6D535Ø41434554524 14E534D4954ØØØØ
ØØØØØØØØ,2263
939 DATAX,3584
```

Graphically Speaking: The Artistic BBS

continued from page 28

```
4Ø INPUT"cOMMAND SUMMARY
          enter TO START
          COMMAND";Q$
5Ø IFQ$="C"ORC$="c"THENGOTO56Ø
6Ø CLS:PRINT"CHOOSE COLOR
                1-GREEN        2-YEL
LOW             3-BLUE         4-RED
                5-BUFF         6-CYA
N               7-MAGENTA      8-ORA
NGE"
7Ø INPUT C:IFC<1ORC>8THENGOTO7Ø
8Ø INPUT"WHAT IS THE NAME OF YOU
R PICTURE";F$
9Ø IF F$="" THENPRINT"YOU MUST C
HOOSE SOMETHING.":GOTO8Ø
1ØØ IFLEN(F$)>8THENPRINT"TOO LON
G. REDO":GOTO8Ø
11Ø CLS(Ø):X=32:Y=16
12Ø A4=INT(Y/2):A5=INT(X/2):A6=(
A4*32)+A5:CP=PEEK(1Ø24+A6)
13Ø FORT=1TO25:NEXTT:SET(X,Y,C):
FORT=1TO25:NEXTT:RESET(X,Y)
14Ø I$=INKEY$
15Ø GOSUB 54Ø
16Ø IFI$=""THENGOTO12Ø
17Ø IFI$="^"THENY=Y-1
18Ø IFI$=CHR$(12)THENCLS(Ø)
19Ø IFI$=CHR$(1Ø)THENY=Y+1
2ØØ IFI$=CHR$(9)THENX=X+1
21Ø IFI$=CHR$(8)THENX=X-1
22Ø IFI$="E"ORI$="e"THENGOSUB55Ø
23Ø IFI$="Q"ORI$="q"THEN END
24Ø IFI$=" "THENGOSUB35Ø
25Ø IFI$="S"ORI$="s"THENGOTO37Ø
26Ø IFI$="L"ORI$="l"THENGOTO43Ø
27Ø IFI$="C"ORI$="c"THENGOSUB62Ø
28Ø IFI$="T"ORI$="t"THENGOSUB66Ø
29Ø IFI$="F"ORI$="f"THENGOSUB73Ø
3ØØ IFX<ØTHENX=Ø
```

```
31Ø IFX>63THENX=63
32Ø IFY<ØTHENY=Ø
33Ø IFY>31THENY=31
34Ø K=Ø:GOTO12Ø
35Ø SET(X,Y,C):RETURN
36Ø 'SAVE ROUTINE
37Ø GOSUB54Ø:OPEN"O",#1,F$
38Ø FORQ=1Ø24TO1535
39Ø W=PEEK(Q):PRINT#1,W
4ØØ NEXTQ
41Ø CLOSE#1:GOTO12Ø
42Ø 'LOAD ROUTINE
43Ø CLS:LINEINPUT"FILE YOU WISH
TO LOAD (INCLUDE  EXTENSION) ";F
F$:IFFF$=""THENGOTO34Ø
44Ø IFINSTR(1,FF$,"/")=ØANDINSTR
(1,FF$,".")=ØTHEN46Ø
45Ø GOTO47Ø
46Ø PRINT"MUST INCLUDE EXTENSION
":FORT=1TO1ØØØ:NEXTT:GOTO43Ø
47Ø OPEN"D",#1,FF$:E=LOF(1):CLOS
E#1:IFE=ØTHENPRINT"FILE NOT FOUN
D":CLOSE#1:KILLFF$:FORT=1TO1ØØØ:
NEXTT:GOTO43Ø
48Ø OPEN"I",#1,FF$
49Ø FORQ=1Ø24TO1535
5ØØ IFEOF(1)THENGOTO52Ø
51Ø INPUT#1,W:POKE Q,W
52Ø NEXTQ:CLOSE#1
53Ø F$=FF$:X=1:Y=1:GOTO12Ø
54Ø POKE (1Ø24+A6),CP:K=Ø:RETURN
55Ø RESET(X,Y):RETURN
56Ø CLS 'COMMAND SUMMARY
57Ø PRINT"  UP ARROW - MOVE CURS
OR UP      DOWN ARROW - MOVE CURS
OR DOWN      RT. ARROW - MOVE CURS
OR RIGHT     LT. ARROW - MOVE CURS
OR LEFT              'S' - SAVE PICT
URE                  'L' - LOAD PICT
URE                  'E' - ERASE AT
```

```
CURSOR"
58Ø PRINT"                LOCATION
              <SPACE> - PUT DOT A
T CURSOR          'C' - PROMPTS C
OLOR                    CHANGE, H
IT 1-8            'T' - TEXT MODE
, STARTS                ABOVE CUR
SOR HIT                 <ENTER> T
O LEAVE"
59Ø PRINT"        'Q' - QUIT
6ØØ INPUT"PRESS <ENTER> TO START
";T$:RUN
61Ø 'CHANGE COLOR
62Ø I$=INKEY$:IFI$=""THENGOTO62Ø
63Ø D=VAL(I$):IFD<1ORD>8THENRETU
RN
64Ø C=D:GOTO12Ø
65Ø 'TEXT MODE
66Ø W=INT(Y/2):Z=INT(X/2):O=(W*3
2)+Z:O=O+1
67Ø IFO<1ORO>51ØTHENRETURN
68Ø I$=INKEY$:IFI$=""THENGOTO68Ø
69Ø IFI$=CHR$(13)THENRETURN
7ØØ IFI$=CHR$(8)THENO=O-1:PRINT@
O," ";:GOTO68Ø
71Ø PRINT@O,I$;:O=O+1
72Ø GOTO67Ø
73Ø FOR XX=1Ø24 TO 1Ø56
74Ø Z=PEEK(XX):POKE 3ØØØØ+XX,Z
75Ø NEXT XX
76Ø PRINT@Ø,"";:INPUT"FILENAME";
F$
77Ø IF LEN(F$)>8THENGOTO76Ø
78Ø FOR XX=1Ø24 TO 1Ø56
79Ø Z=PEEK(3ØØØØ+XX):POKE XX,Z
8ØØ NEXT XX
81Ø RETURN
```

Coming to 'Terms' With the CoCo 3

By Rick Adams and Dale Lear

Term 3 is a simple terminal program for the Color Computer 3. It has few features; the purpose of this program is to demonstrate the fact that reliable 1200 Baud RS-232 communication out the CoCo's "bit-banger" port may be obtained by utilizing the programmable interrupt timer included with the Color Computer 3.

Sharp-eyed, technically-oriented users will note that the interrupt routine is driven by setting the new timer at seven times the Baud rate, making the sampling rate on the bit-banger port fast enough for reliable start-bit detection. This luxury is not available on the Color Computer 2. There are only two fixed-rate interrupt clocks built in; one is too fast to use for this purpose, while the other is too slow!

Despite the simplicity of *Term 3*, it does have some things going for it: it supports true upper- and lowercase letters in the 40- or 80-column modes available on the Color Computer 3, and both input and output are fully buffered, allowing type-ahead.

With a little experimentation, Color Computer users with moderate assembly-language experience could add features to this bare-bones communication demo such as ASCII buffer uploading and downloading, use of the function keys (welcome addition) to generate user-defined text strings, and so on.

TERM3.BAS is the BASIC program that pokes a number of communications parameters into memory, loads the machine language portion of the package and starts things going. The comments regarding the parameters are fairly self-explanatory; the defaults shown will do nicely in the majority of cases. (Note that setting the left margin to '2' avoids the problem of the width 40 mode on a TV set causing the first two characters to disappear.)

The assembly language portion of *Term 3* is named *TERM3.BIN*. Users with assembly language experience may use the source listing shown as *TERM3.SRC* (and their favorite assembler) to produce this file.

Perhaps we'll see some of you on Delphi as you take your *Term 3* program out for a test drive through the telecommunications network. We hope to see you there! □

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

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

Listing 1: TERM3BAS

```
1Ø CLS
2Ø PRINT "======================
====="
3Ø PRINT "   TERM3   VERSION 1.Ø
"
4Ø PRINT "      COPYRIGHT  1986"
5Ø PRINT "DALE LEAR  AND RICK AD
AMS"
6Ø PRINT "======================
======"
7Ø '
8Ø '*** PARAMETERS ***
9Ø '
1ØØ '---DISPLAY MODE---
11Ø '
12Ø 'NUMBER OF COLUMNS (4Ø OR 8Ø
)
13Ø CMAX=4Ø
14Ø '
15Ø 'NUMBER OF LINES (24)
16Ø LMAX=24
17Ø '
18Ø 'LEFT MARGIN (FOR TV)
19Ø LFMAR=2
2ØØ '
21Ø 'FOREGROUND COLOR (WHITE)
22Ø FCOLOR=255
23Ø '
24Ø 'BACKGROUND COLOR (BLACK)
25Ø BCOLOR=Ø
26Ø '
27Ø '---BAUD RATE---
28Ø BAUD=12ØØ
29Ø '
3ØØ 'TIMER SET TO 7X BAUD RATE
31Ø T=INT((14318181/4)/(BAUD*7))
32Ø T1=INT(T/256)
33Ø T2=T-256*T1
34Ø '
35Ø '---AUTOLF---
36Ø AUTOLF=Ø
37Ø '    Ø-NO
38Ø '    1-YES
39Ø '
4ØØ '---DUPLEX---
41Ø DUPLEX=Ø
42Ø '    Ø-FULL
43Ø '    1-HALF
44Ø '
45Ø WIDTH CMAX
46Ø POKE &H2ØØ,CMAX
47Ø POKE &H2Ø1,LMAX
48Ø POKE &H2Ø2,LFMAR
49Ø POKE &H2Ø3,FCOLOR
5ØØ POKE &H2Ø4,BCOLOR
51Ø POKE &H2Ø5,T1
52Ø POKE &H2Ø6,T2
53Ø POKE &H2Ø7,AUTOL
54Ø POKE &H2Ø8,DUPLEX
55Ø LOADM "TERM3"
56Ø EXEC
```

Listing 2: TERM3

```
                   00010 *----------------------------
                   00020 *   TERM3   VERSION 1.0
                   00030 *      COPYRIGHT  1986
                   00040 *DALE LEAR  AND RICK ADAMS
                   00050 *----------------------------
                   00060 *
                   00070 * TERM3 IS A SIMPLE TEMINAL
                   00080 * PACKAGE FOR THE COCO 3
                   00090 * UTILIZING THE PROGRAMMABLE
                   00100 * INTERRUPT TIMER TO CONTROL
                   00110 * THE SERIAL PORT.
                   00120 *
                   00130
                   00140 *----------------------------
                   00150 * EQUIV DEFINITIONS
                   00160 *
         0020      00170 BLANK  EQU 32
         0003      00180 BREAK  EQU 3
         000D      00190 CR     EQU 13
         000A      00200 LF     EQU 10
         0008      00210 BS     EQU 8
         0100      00220 SZOUT  EQU $100 SIZE OF OUTPUT BUFFER
         0100      00230 SZIN   EQU $100 SIZE OF INPUT BUFFER
         0000      00240 ATTR   EQU $0
                   00250 *----------------------------
                   00260 *DIRECT PAGE COMMON
                   00270 *
0000               00280        ORG 0
                   00290 *------------------------------
                   00300 *
                   00310 *SCREEN DISPLAY COMMON
                   00320 *
0000               00330 LINCOL
0000               00340 LIN    RMB 1    CUR LINE
0001               00350 COL    RMB 1    CUR COL
                   00360 *------------------------------
                   00370 * COMMUNICATIONS COMMON
                   00380 *
0002               00390 XSLICE  RMB 1
0003               00400 XBIT    RMB 1
0004               00410 XCHAR   RMB 1
0005               00420 RSLICE  RMB 1
0006               00430 RBIT    RMB 1
0007               00440 RCHAR   RMB 1
0008               00450 BHOLD   RMB 1
0009               00460 XHOLD   RMB 2
                   00470 *------------------------------
                   00480 * BUFFERED I/O POINTERS
                   00490 *
000B               00500 GETOUT RMB 2
000D               00510 PUTOUT RMB 2
                   00520 *
000F               00530 GETIN  RMB 2
0011               00540 PUTIN  RMB 2
                   00550 *----------------------------
                   00560 * PARAMATERS POKED IN BY BASIC
                   00570 *
0200               00580        ORG $200
0200               00590 CMAX   RMB 1    MAX COL
0201               00600 LMAX   RMB 1    MAX LINE
0202               00610 LFMAR  RMB 1    LEFT MARGIN
0203               00620 FCOLOR RMB 1    FOREGROUND COLOR
0204               00630 BCOLOR RMB 1    BACKGROUND COLOR
0205               00640 BAUD   RMB 2    BAUD RATE CONSTANT
0207               00650 AUTOLF RMB 1    0-NO, 1-YES
0208               00660 DUPLEX RMB 1    0-FULL, 1-HALF
                   00670 *----------------------------
                   00680 *BUFFERS
                   00690 *
4000               00700        ORG $4000
4000               00710 VIDBUF  RMB $1800   SCREEN
5800               00720 INBUF   RMB SZIN    INPUT BUF
5900               00730 OUTBUF  RMB SZOUT   OUTPUT BUF
                   00740
                   00750 *----------------------------
                   00760 *   MAINLINE
                   00770 *----------------------------
6000               00780        ORG $6000
6000               00790 MAIN
                   00800
                   00810 *----------------------------
                   00820 *INITIALIZATION
                   00830 *
6000 7F    FF40    00840  CLR $FF40      TURN OFF DISK MOTOR
6003 7F    FFD9    00850  CLR $FFD9      SPEED UP CPU
6006 17    0138    00860  LBSR CLRSCN    CLEAR SCREEN
                   00870 *
                   00880 *               CLEAR I/O BUFFERS
6009 8E    5900    00890  LDX #OUTBUF
600C 9F    0B      00900  STX GETOUT
600E 9F    0D      00910  STX PUTOUT
6010 8E    5800    00920  LDX #INBUF
6013 9F    0F      00930  STX GETIN
6015 9F    11      00940  STX PUTIN
6017 108E 0200     00950  LDY #SZIN+SZOUT
601B C6    FF      00960  LDB #-1
601D               00970 ISET
601D E7    80      00980  STB ,X+    FILL BUFS W/-1'S
601F 31    3F      00990  LEAY -1,Y
6021 26    FA      01000  BNE ISET
                   01010 *
6023 17    016E    01020  LBSR TIMER     START TIMER
                   01030
                   01040 *----------------------------
                   01050 *BODY OF MAINLINE
                   01060 *
                   01070
6026               01080 LOOP1
6026 17    0035    01090  LBSR RECV  RS232 -> A REG
6029 81    FF      01100  CMPA #-1
602B 27    05      01110  BEQ LOOP2
602D 17    0075    01120  LBSR PUT   A REG -> SCREEN
6030 20    F4      01130  BRA LOOP1  (TIS MORE IMPORTANT TO
                   01140 *           RECIEVE THAN TO TRANSMIT)
                   01150
6032               01160 LOOP2
6032 17    0040    01170  LBSR GET   KEYBOARD -> A REG
6035 81    FF      01180  CMPA #-1
6037 27    ED      01190  BEQ LOOP1
6039 7D    0208    01200  TST DUPLEX SKIP IF FULL DUPLEX(0)
603C 27    03      01210  BEQ LOOP3
603E 17    0064    01220  LBSR PUT   A REG -> SCREEN
6041               01230 LOOP3
6041 17    0002    01240  LBSR SEND  A REG -> RS232
6044 20    E0      01250  BRA LOOP1
                   01260
                   01270 *----------------------------
                   01280 *SEND CHARACTER TO SERIAL PORT
                   01290 *  (VIA OUTBUF)
                   01300 * ENTER W/ CHAR IN A REG
                   01310 *
6046               01320 SEND
6046 34    02      01330  PSHS A
6048 9E    0D      01340  LDX PUTOUT
604A               01350 SND1
604A E6    84      01360  LDB ,X
604C C1    FF      01370  CMPB #-1
604E 26    FA      01380  BNE SND1 WAIT FOR LAST XMIT
6050 A7    80      01390  STA ,X+  PUT CHAR IN BUFFER
6052 8C    5A00    01400  CMPX #OUTBUF+SZOUT
6055 26    03      01410  BNE SND2
6057 8E    5900    01420  LDX #OUTBUF
605A               01430 SND2
605A 9F    0D      01440  STX PUTOUT
605C 35    82      01450  PULS A,PC
                   01460 *----------------------------
                   01470 *RCV CHARACTER FROM SERIAL PORT
                   01480 *  (VIA INBUF)
                   01490 * RETURN W/ CHAR IN A REG
                   01500 * -1 IF NOTHING RECEIVED
                   01510 *
605E               01520 RECV
605E 9E    0F      01530  LDX GETIN
6060 A6    84      01540  LDA ,X
6062 81    FF      01550  CMPA #-1
6064 27    0C      01560  BEQ RCV1
6066 C6    FF      01570  LDB #-1
6068 E7    80      01580  STB ,X+
606A 8C    5900    01590  CMPX #INBUF+SZIN
606D 26    03      01600  BNE RCV1
606F 8E    5800    01610  LDX #INBUF
6072               01620 RCV1
6072 9F    0F      01630  STX GETIN
6074 39            01640  RTS
                   01650 *----------------------------
                   01660 * GET CHAR FROM KEYBOARD
                   01670 * RETURN W/ CHAR IN A REG
                   01680 * -1 IF NOTHING RECEIVED
                   01690 *
6075               01700 GET
6075 17    0090    01710  LBSR XYCALC
6078 E6    01      01720  LDB 1,X
607A 34    14      01730  PSHS B,X
607C C6    C0      01740  LDB #$C0
607E EA    01      01750  ORB 1,X
6080 E7    01      01760  STB 1,X
6082 AD    9F A000 01770  JSR [$A000]  CALL BASIC KBOARD ROUTINE
6086 27    16      01780  BEQ GET8
6088 84    7F      01790  ANDA #$7F
608A C6    EF      01800  LDB #$EF     CHECK CTRL KEY
608C F7    FF02    01810  STB $FF02
608F F6    FF00    01820  LDB $FF00
6092 C4    40      01830  ANDB #$40
6094 26    0A      01840  BNE GET9
6096 81    3D      01850  CMPA #'=     BASIC KB DRIV GIVES A
6098 27    04      01860  BEQ GET8     PHONY "=" ON CTRL KEY
609A 84    1F      01870  ANDA #$1F    IF DOWN REMOVE BITS 5+6
```

```
609C 20    02        01880  BRA GET9
609E                 01890 GET8
609E 86    FF        01900  LDA #-1
60A0                 01910 GET9
60A0 35    14        01920  PULS B,X
60A2 E7    01        01930  STB 1,X
60A4 39              01940  RTS
                     01950 *-----------------------------
                     01960 * PUT CHAR TO SCREEN
                     01970 * ENTER W/ CHAR IN A REG
                     01980 *
60A5                 01990 PUT
60A5 34    12        02000  PSHS A,X
60A7 81    20        02010  CMPA #32
60A9 24    1E        02020  BHS PUT3
                     02030
                     02040 *TEST FOR CONTROL CHARACTER
60AB 81    0D        02050  CMPA #CR
60AD 27    0C        02060  BEQ PUT1
60AF 81    0A        02070  CMPA #LF
60B1 27    12        02080  BEQ PUT2
60B3 81    08        02090  CMPA #BS
60B5 26    1D        02100  BNE PUT9
                     02110
                     02120 *BACK SPACE
60B7 0A    01        02130  DEC COL
60B9 20    17        02140  BRA PUT8
                     02150
                     02160 *CARRIAGE RETURN
60BB                 02170 PUT1
60BB F6    0202      02180  LDB LFMAR
60BE D7    01        02190  STB COL
60C0 7D    0207      02200  TST AUTOLF
60C3 27    0D        02210  BEQ PUT8
                     02220
                     02230 *LINE FEED
60C5                 02240 PUT2
60C5 0C    00        02250  INC LIN
60C7 20    09        02260  BRA PUT8
                     02270
                     02280 *PRINTABLE CHARACTER
60C9                 02290 PUT3
60C9 17    003C      02300  LBSR XYCALC (CALC SCREEN LOC)
60CC A6    E4        02310  LDA ,S
60CE A7    84        02320  STA ,X       (STORE CHARACTER)
60D0 0C    01        02330  INC COL
                     02340
                     02350 *
                     02360 *FIX X-Y LOC
60D2                 02370 PUT8
60D2 8D    02        02380  BSR XYFIX
60D4                 02390 PUT9
60D4 35    92        02400  PULS A,X,PC
                     02410 *-----------------------------
                     02420 * BRING LINE/COL IN VALID RANGE
                     02430 *
60D6                 02440 XYFIX
60D6                 02450 XY1
60D6 DC    00        02460  LDD LINCOL
60D8 F1    0200      02470  CMPB CMAX
60DB 24    0C        02480  BHS HICOL
60DD F1    0202      02490  CMPB LFMAR
60E0 25    10        02500  BLO LOCOL
60E2 B1    0201      02510  CMPA LMAX
60E5 24    19        02520  BHS HILIN
60E7 20    1E        02530  BRA XY9
                     02540
                     02550 * COLUMN TOO HIGH,
                     02560 * GO TO NEXT LINE
60E9                 02570 HICOL
60E9 F6    0202      02580  LDB LFMAR
60EC D7    01        02590  STB COL
60EE 0C    00        02600  INC LIN
60F0 20    E4        02610  BRA XY1
                     02620
                     02630 * COLUMN TOO LOW,
                     02640 * GO TO PREV LINE
60F2                 02650 LOCOL
60F2 F6    0200      02660  LDB CMAX
60F5 5A              02670  DECB
60F6 D7    01        02680  STB COL
60F8 0D    00        02690  TST LIN
60FA 27    DA        02700  BEQ XY1
60FC 0A    00        02710  DEC LIN
60FE 20    D6        02720  BRA XY1
                     02730
                     02740 * LINE TOO HIGH
                     02750 * SCROLL
6100                 02760 HILIN
6100 17    0016      02770  LBSR SCROLL
6103 0A    00        02780  DEC LIN
6105 20    CF        02790  BRA XY1
                     02800
6107                 02810 XY9
6107 39              02820  RTS
```

```
                     02830 *-----------------------------
                     02840 * CALCULATE X/Y SCREEN LOC
                     02850 * ENTTRY LINE/COL
                     02860 * EXIT   X-SCREEN LOC
                     02870 *
6108                 02880 XYCALC
6108 8E    4000      02890  LDX #VIDBUF
610B 96    00        02900  LDA LIN
610D F6    0200      02910  LDB CMAX
6110 58              02920  LSLB (ACCOUNT FOR ATTRIBUTE BYTE)
6111 3D              02930  MUL
6112 30    8B        02940  LEAX D,X
6114 D6    01        02950  LDB COL
6116 58              02960  LSLB (ACCOUNT FOR ATTRIBUTE BYTE)
6117 3A              02970  ABX
6118 39              02980  RTS
                     02990 *-----------------------------
                     03000 *SCROLL SCREEN UP ONE LINE
                     03010 *
6119                 03020 SCROLL
6119 8E    4000      03030  LDX #VIDBUF
611C F6    0200      03040  LDB CMAX
611F 58              03050  LSLB (ACCOUNT FOR ATTRIBUTE BYTE)
6120 4F              03060  CLRA
6121 33    8B        03070  LEAU D,X
6123 B6    0201      03080  LDA LMAX
6126 4A              03090  DECA
6127 3D              03100  MUL
6128 1F    02        03110  TFR D,Y
612A                 03120 SCR1
612A EC    C1        03130  LDD ,U++
612C ED    81        03140  STD ,X++
612E 31    3E        03150  LEAY -2,Y
6130 26    F8        03160  BNE SCR1
6132 B6    0200      03170  LDA CMAX
6135                 03180 SCR2
6135 C6    20        03190  LDB #BLANK
6137 E7    80        03200  STB ,X+
6139 C6    00        03210  LDB #ATTR
613B E7    80        03220  STB ,X+
613D 4A              03230  DECA
613E 26    F5        03240  BNE SCR2
6140 39              03250  RTS
                     03260 *-----------------------------
                     03270 * CLEAR SCREEN
                     03280 *
6141                 03290 CLRSCN
6141 F6    0203      03300  LDB FCOLOR SET FOREGROUND COLOR
6144 F7    FFB8      03310  STB $FFB8
6147 F6    0204      03320  LDB BCOLOR SET BACKGROUND COLOR
614A F7    FFB0      03330  STB $FFB0
614D F7    FF9A      03340  STB $FF9A  AND BORDER
6150 C6    4C        03350  LDB #$4C
6152 F7    FF90      03360  STB $FF90  SET INITIALIZATION REGISTER
6155 C6    03        03370  LDB #3
6157 F7    FF98      03380  STB $FF98  SET VIDEO MODE REGISTER
615A C6    05        03390  LDB #$5
615C B6    0200      03400  LDA CMAX
615F 81    50        03410  CMPA #80
6161 26    02        03420  BNE CL1
6163 C6    15        03430  LDB #$15
6165                 03440 CL1
6165 F7    FF99      03450  STB $FF99  SET VIDEO RES REGISTER
6168 CC    4000      03460  LDD #VIDBUF
616B 44              03470  LSRA
616C 56              03480  RORB
616D 44              03490  LSRA
616E 56              03500  RORB
616F 44              03510  LSRA
6170 56              03520  RORB
6171 8A    E0        03530  ORA #$E0
6173 B7    FF9D      03540  STA $FF9D  SET VERT OFFSET REGISTERS
6176 F7    FF9E      03550  STB $FF9E
6179 7F    FF9F      03560  CLR $FF9F
                     03570
617C 8E    4000      03580  LDX #VIDBUF
617F 108E 0780       03590  LDY #80*24
6183 CC    2000      03600  LDD #BLANK*256+ATTR
6186                 03610 CL2
6186 ED    81        03620  STD ,X++   CLEAR SCREEN
6188 31    3F        03630  LEAY -1,Y
618A 26    FA        03640  BNE CL2
618C 0F    00        03650  CLR LIN
618E F6    0202      03660  LDB LFMAR
6191 D7    01        03670  STB COL
6193 39              03680  RTS
                     03690 *-----------------------------
                     03700 * SET UP PROGRAMMABLE
                     03710 * INTERRUPT TIMER
                     03720 *
6194                 03730 TIMER
                     03740 *
                     03750 *SET UP INTERRUPTS
6194 1A    50        03760  ORCC #$50      OFF FOR NOW
                     03770 *
```

```
                  Ø378Ø *TURN OFF OLD IRUPTS
6196 C6   2C      Ø379Ø  LDB #$2C
6198 F7   FFØ1    Ø38ØØ  STB $FFØ1
619B F7   FFØ3    Ø381Ø  STB $FFØ3
619E F7   FF23    Ø382Ø  STB $FF23
61A1 F7   FF23    Ø383Ø  STB $FF23
61A4 F6   FFØØ    Ø384Ø  LDB $FFØØ
61A7 F6   FFØ2    Ø385Ø  LDB $FFØ2
61AA F6   FF2Ø    Ø386Ø  LDB $FF2Ø
61AD F6   FF22    Ø387Ø  LDB $FF22
                  Ø388Ø *
61BØ 3Ø   8D ØØ29 Ø389Ø  LEAX DOFIRQ,PCR
61B4 BF   Ø11Ø    Ø39ØØ  STX $11Ø         SET FIRQ PROGRAM
                  Ø391Ø *
                  Ø392Ø * SET-UP TIMER INTERRUPT
61B7 C6   6Ø      Ø393Ø  LDB #$6Ø
61B9 F7   FF91    Ø394Ø  STB $FF91        SELECT CLOCK
61BC 7F   FF92    Ø395Ø  CLR $FF92
61BF C6   2Ø      Ø396Ø  LDB #$2Ø
61C1 F7   FF93    Ø397Ø  STB $FF93        ENABLE TIMER INTERRUPT
61C4 C6   5C      Ø398Ø  LDB #$5C
61C6 F7   FF9Ø    Ø399Ø  STB $FF9Ø        ENABLE GIME FIRQ
61C9 FC   Ø2Ø5    Ø4ØØØ  LDD BAUD
61CC F7   FF95    Ø4Ø1Ø  STB $FF95
61CF B7   FF94    Ø4Ø2Ø  STA $FF94        SET UP TIMER COUNT
                  Ø4Ø3Ø *
                  Ø4Ø4Ø * INIT XMITTER/RECVR
61D2 ØF   Ø5      Ø4Ø5Ø  CLR RSLICE
61D4 ØF   Ø2      Ø4Ø6Ø  CLR XSLICE
61D6 ØF   Ø6      Ø4Ø7Ø  CLR RBIT
61D8 ØF   Ø3      Ø4Ø8Ø  CLR XBIT
61DA 1C   AF      Ø4Ø9Ø  ANDCC #$AF       ENABLE INTERRUPTS
61DC 39           Ø41ØØ  RTS
                  Ø411Ø *-----------------------------
                  Ø412Ø * PROCESS TIMER INTERRUPT
                  Ø413Ø *
61DD              Ø414Ø DOFIRQ
61DD D7   Ø8      Ø415Ø  STB BHOLD
61DF 7F   FF93    Ø416Ø  CLR $FF93        CLEAR TIMER INTERRUPT
61E2 C6   2Ø      Ø417Ø  LDB #$2Ø
61E4 F7   FF93    Ø418Ø  STB $FF93
                  Ø419Ø
                  Ø42ØØ *
                  Ø421Ø *SERVICE TRANSMITTER
                  Ø422Ø *
61E7 ØA   Ø2      Ø423Ø  DEC XSLICE
61E9 26   3E      Ø424Ø  BNE XMI9
61EB 9F   Ø9      Ø425Ø  STX XHOLD
61ED C6   Ø7      Ø426Ø  LDB #7
61EF D7   Ø2      Ø427Ø  STB XSLICE
61F1 D6   Ø3      Ø428Ø  LDB XBIT
61F3 27   Ø4      Ø429Ø  BEQ XMI1
61F5 ØA   Ø3      Ø43ØØ  DEC XBIT
61F7 26   2Ø      Ø431Ø  BNE XMI3
61F9              Ø432Ø XMI1
61F9 9E   ØB      Ø433Ø  LDX GETOUT
61FB E6   84      Ø434Ø  LDB ,X     ON NEXT BYTE IN BUFFER
61FD C1   FF      Ø435Ø  CMPB #-1     -1
61FF 27   26      Ø436Ø  BEQ XMI8
62Ø1 D7   Ø4      Ø437Ø  STB XCHAR   ELSE
62Ø3 C6   FF      Ø438Ø  LDB #-1
62Ø5 E7   8Ø      Ø439Ø  STB ,X+
62Ø7 8C   5AØØ    Ø44ØØ  CMPX #OUTBUF+SZOUT
62ØA 26   Ø3      Ø441Ø  BNE XMI2
62ØC 8E   59ØØ    Ø442Ø  LDX #OUTBUF
62ØF              Ø443Ø XMI2
62ØF 9F   ØB      Ø444Ø  STX GETOUT ADVANCE BUFFER POINTER
6211 C6   ØA      Ø445Ø  LDB #1Ø
6213 D7   Ø3      Ø446Ø  STB XBIT   XMIT 1Ø BITS
6215 1C   FE      Ø447Ø  ANDCC #$FE START BIT->CARRY
6217 2Ø   Ø7      Ø448Ø  BRA XMI4
6219              Ø449Ø XMI3
6219 D6   Ø4      Ø45ØØ  LDB XCHAR
621B 1A   Ø1      Ø451Ø  ORCC #1     STOP BIT->CARRY
621D 56           Ø452Ø  RORB        DATA BIT->CARRY
621E D7   Ø4      Ø453Ø  STB XCHAR
622Ø              Ø454Ø XMI4
622Ø C6   ØØ      Ø455Ø  LDB #Ø
6222 59           Ø456Ø  ROLB        CARRY->RS232 OUT
6223 59           Ø457Ø  ROLB
6224 F7   FF2Ø    Ø458Ø  STB $FF2Ø
6227              Ø459Ø XMI8
6227 9E   Ø9      Ø46ØØ  LDX XHOLD
6229              Ø461Ø XMI9
                  Ø462Ø
                  Ø463Ø *
                  Ø464Ø *SERVICE RECEIVER
                  Ø465Ø *
6229 D6   Ø6      Ø466Ø  LDB RBIT         IF NOT RECEIVING
622B 27   Ø6      Ø467Ø  BEQ RCI1          GO CHECK FOR START BIT
622D ØA   Ø5      Ø468Ø  DEC RSLICE       ELSE
622F 27   18      Ø469Ø  BEQ RCI3          CONTINUE TO RECEIVE
6231 2Ø   3D      Ø47ØØ  BRA RCI9
6233              Ø471Ø RCI1
6233 F6   FF22    Ø472Ø  LDB $FF22  WATCH FOR START BIT
6236 56           Ø473Ø  RORB
6237 24   Ø4      Ø474Ø  BCC RCI2
6239 ØF   Ø5      Ø475Ø  CLR RSLICE NO START, CLEAR COUNTER
623B 2Ø   33      Ø476Ø  BRA RCI9
623D              Ø477Ø RCI2
623D ØC   Ø5      Ø478Ø  INC RSLICE POSSIBLE START, INC COUNTER
623F D6   Ø5      Ø479Ø  LDB RSLICE
6241 C1   Ø4      Ø48ØØ  CMPB #4    IF 4 X'S, START RECEIVER
6243 26   2B      Ø481Ø  BNE RCI9
6245 C6   ØA      Ø482Ø  LDB #1Ø    RECEIVE 1Ø BITS
6247 D7   Ø6      Ø483Ø  STB RBIT
6249              Ø484Ø RCI3
6249 ØA   Ø6      Ø485Ø  DEC RBIT
624B 27   ØF      Ø486Ø  BEQ RCI4   SEE IF FINISHED BYTE
624D F6   FF22    Ø487Ø  LDB $FF22
625Ø 56           Ø488Ø  RORB       DATA BIT->CARRY
6251 D6   Ø7      Ø489Ø  LDB RCHAR
6253 56           Ø49ØØ  RORB       CARRY->RECV BYTE
6254 D7   Ø7      Ø491Ø  STB RCHAR
6256 C6   Ø7      Ø492Ø  LDB #7
6258 D7   Ø5      Ø493Ø  STB RSLICE SET COUNTER FOR NXT BIT
625A 2Ø   14      Ø494Ø  BRA RCI9
625C              Ø495Ø RCI4
625C 9F   Ø9      Ø496Ø  STX XHOLD
625E 9E   11      Ø497Ø  LDX PUTIN
626Ø D6   Ø7      Ø498Ø  LDB RCHAR
6262 E7   8Ø      Ø499Ø  STB ,X+    DELIVER CHAR TO BUFFER
6264 8C   59ØØ    Ø5ØØØ  CMPX #INBUF+SZIN
6267 26   Ø3      Ø5Ø1Ø  BNE RCI5
6269 8E   58ØØ    Ø5Ø2Ø  LDX #INBUF
626C              Ø5Ø3Ø RCI5
626C 9F   11      Ø5Ø4Ø  STX PUTIN
626E 9E   Ø9      Ø5Ø5Ø  LDX XHOLD
627Ø              Ø5Ø6Ø RCI9
627Ø D6   Ø8      Ø5Ø7Ø  LDB BHOLD
6272 3B           Ø5Ø8Ø  RTI
                  Ø5Ø9Ø
          6ØØØ    Ø51ØØ  END MAIN

ØØØØØ TOTAL ERRORS
```

continued from page 13

programs conflict with the joysticks. Maybe Mr. van der Poel can solve this little quirk on subsequent releases.

Ultra Telepatch is supplied on disk only and requires 64K RAM and an unpatched version of *Telewriter-64*. It can be backed up for safekeeping, and comes with an 11-page instruction manual that is easy to follow.

I believe you will find *Ultra Telepatch* as impressive as I have. I've learned that with CoCo, anything is possible.

(Bob van der Poel Software, 17435-57 Avenue, Edmonton, Alberta, Canada T6M 1E1, $19.95 plus $2 S/H)

— Jerry Semones

continued from page 43

```
P5+2YY
2200 NEXTYY:PRINT#-2,CHR$(P5);:N
EXT:PRINT#-2,CHR$(13);:NEXT:GOTO
3000
2210 FORY5=0TO188STEP4:PRINT#-2,
CHR$(27)CHR$(83)"0512";:FORX5=0T
O255:P5=Z5
2220 IFPPOINT(X5,Y5)<1THENP5=A1
2230 IFPPOINT(X5,Y5+1)<1THENP5=P
5+A2
2240 IFPPOINT(X5,Y5+2)<1THENP5=P
5+A3
2250 IFPPOINT(X5,Y5+3)<1THENP5=P
5+A4
2260 PRINT#-2,CHR$(P5)CHR$(P5);:
NEXT:PRINT#-2,CHR$(13);:NEXT:GOT
O3000
3000 NEXTPG
```

Keycad/Keyflow: CoCocad and CoCoflow Modification

By James Ventling

This is a modification for those who don't always have a joystick or mouse handy to use with either the *CoCocad* (Oct. '85) or the *CoCoflow* (Mar. '86) programs. I wanted to use *CoCoflow* with some of my students but didn't have joysticks to go around. Instead, I changed the program to accept keyboard input. In place of the joystick, the arrow keys are used for cursor movement. The arrow keys may be held down for continuous movement or, for faster movement, hold the 'J' key (for jump) while using the arrows. The CLEAR key is used in place of the firebutton. When making a selection from the icons at the top of the screen, be sure to press the down arrow key until the cursor reappears or the option may de-select before you have a chance to use it.

While using *CoCoflow*, we found that the symbols for decision and connection were too small to place text information in. I made a further modification to increase the size of these shapes. We also dicovered that a screen-print utility could be added to *CoCoflow* due to its smaller memory requirements. In the original *CoCocad* and *CoCoflow*, to do a screen-print you had to dump all nine screens to disk and then use a separate screen print program. This used 28 grans of disk space! By adding a screen-print routine to the end of *CoCoflow*, you can print directly from memory.

Lines 20 through 30 replace the joystick input with keyboard input. PEEK is used to read the keyboard so you can tell if a key is being held down. The keyboard table is cleared in Line 20 so you can tell when a key has been released. Then the program looks to see if the 'J' key, any of the arrow keys, or the CLEAR key is being pressed.

Variables "XX" and "YY" are used to simulate a joystick input. The variables 'X' and 'Y' are not incremented directly because these variables are also used in some subroutines and could be changed when you least want it. Lines 29 and 30 check to make sure 'X' and 'Y' don't go out of bounds.

Line 121 starts the cursor at a convenient location at the top of the screen near the icon selection. You also have to keep the use of the CLEAR key from being misinterpreted as a keystroke when placing text on the screen. Changing Line 550 so as to ignore the CLEAR key takes care of this.

To change the size of the decision and connect symbols in *CoCoflow*, you must change lines 120, 910 and 930. In Line 120, array sizes are increased to accommodate the larger symbols. The new DRAW strings and GET-PUT sizes for the larger symbols are in lines 910 and 930.

To add a screen-print routine to *CoCoflow*, first eliminate the screen dump in lines 1970 and 1980. Keep the page-display routine in Line 1980 and add you own screen-print routine starting at Line 2000. I have included a simple BASIC screen-print routine for the C-ITOH Prowriter.

Many thanks to Peter Kerckhoff for creating the original *CoCocad* and to Dennis Page for the *CoCoflow* modification. Remember to give credit to *CoCocad* or *CoCoflow* if you publish any graphics created with these programs.

To make the modification for keyboard input, load Cococad or Cocoflow and type in Listing 1.

To make the modification for larger decision and connection symbols in *CoCoflow* type in Listing 2.

To add a screen-print routine, change lines 1970, 1980 and 1990. Add the screen-print routine at Line 2000. Don't forget Line 3000.

Listing 1: COCOMOD1

```
20 FORQZ=339TO344:POKEQZ,255:NEX
T:JK=PEEK(340)
21 IFPEEK(341)=247THENYY=YY-1:IF
JK=253THENYY=YY-7
22 IFPEEK(342)=247THENYY=YY+1:IF
YY<7THENYY=YY+12ELSEIFJK=253THEN
YY=YY+7
23 IFPEEK(343)=247THENXX=XX-1:IF
YY<6THENXX=XX-1ELSEIFJK=253THENX
X=XX-7
24 IFPEEK(344)=247THENXX=XX+1:IF
YY<6THENXX=XX+1ELSEIFJK=253THENX
X=XX+7
25 IFPEEK(339)=191THENP=3ELSEP=0
29 X=XX*4:IFX<3THENX=3:XX=1:ELSE
IFX>252THENX=252:XX=63
30 Y=YY*4:IFY<3THENY=3:YY=1:ELSE
IFY>188THENY=188:YY=47
121 XX=16:YY=9
550 GOSUB60:A$=INKEY$:GOSUB70:IF
A$=""THEN550ELSEIFASC(A$)=12THEN
550 ELSEPLAYB$:IFA$=CHR$(13)THEN
POKEAD(PG),255: AD(PG)=AD(PG)+1:
POKEAD(PG),0:GOSUB110: GOTO520
```

Listing 2: COCOMOD2

```
120 DIM C$(3),A(8),AD(8),C1(1),C
2(1),C3(1), L1(6),L2(6),L3(6),L4
(6),CM(45),CO(45),MD(255), MO(25
5):B$="V31L10004B":NF$="NONE"
910 DRAW"BD16M+24,-16M+24,+16M-2
4,+16M-24,-16":XW=48:YW=33:RETUR
N:'DECISION
930 DRAW"BD9U3EUE3RER3FRF3DFD3GD
G3LGL3HLH3UHU2":XW=17:YW=17:RETU
RN:'CONN
```

Listing 3: COCOMOD3

```
1970 '
1980 FOR PG=0 TO 8:PMODE4,1:SCRE
EN1,1:COLOR0,1:PCLS:GOSUB1790
1990 '
2000 'PUT YOUR SCREEN PRINT ROUT
INE HERE
2010 CLS:PRINT"GRAPHICS PRINT-OU
T FOR
PROWRITER":INPUT"READY PRINTER";
QQ
2020 PRINT:PRINT"WHAT SIZE PRINT
-OUT?":INPUT"1 OR 2",NN:IFNN<1 O
R NN>2THEN2020
2050 PRINT#-2,CHR$(27)CHR$(84)"1
6";:PMODE4,1:SCREEN1,1
2060 A1=3:A2=12:A3=48:A4=192:Z5=
0:ONNN GOTO2180,2210
2180 FORY5=0TO190STEP8:PRINT#-2,
CHR$(27)CHR$(83)"0256";:FORX5=0T
O255:P5=Z5:FORYY=0TO7
2190 IFPPOINT(X5,Y5+YY)<1THENP5=
```

continued on page 42

BARDEN'S BUFFER

Presenting a Quiz for Color Computer Assembly Language

By William Barden, Jr.
Rainbow Contributing Editor

Our local Color Computer Users Group in Orange County, Calif., is an organization with somewhat eclectic interests. Within the organization are special interest groups on BASIC, assembly language, sushi and automatic weapons. In spite of the weird aspects of the user's group, it's fun to attend the meetings. At the last meeting, the chairman of the SIG on assembly language, presented an enjoyable little assembly language quiz. (Actually, it wasn't that enjoyable. The doors were locked and we couldn't get out until we had tried the quiz.)

The quiz is reproduced in this month's column so you can test yourself and see if you really know assembly language as well as you think you do. Readers who get all answers correct will be treated to a sushi lunch and a used AK-47 assault rifle the next time they're in Orange County. The answers to all questions are at the end of this column. A score of 10 to 12 qualifies you as a master assembly language programmer, 7 to 9 indicates that you are a professional AL programmer, 4 to 6 marks you as a journeyman AL programmer, and less than 4 means you better go back and hit the books to brush up on your programming skills.

The Quiz

1) Here's an easy one to begin with. Write an assembly language program to load the A Register with decimal 230 and the B Register with decimal 15, and then find the product of the two numbers in the D Register (A and B).

2) What does this code do?

```
        LDD     OP1
        LDU     #ø
LOOP    SUBD    OP2
        BLO     OUT
        LEAU    1,U
        BRA     LOOP
OUT     JMP     OUT
OP1     FDB     XXX
OP2     FDB     XXX
```

3) Here's a relative toughie, but if you write down the results for a few test cases, you should be able to see what this code accomplishes:

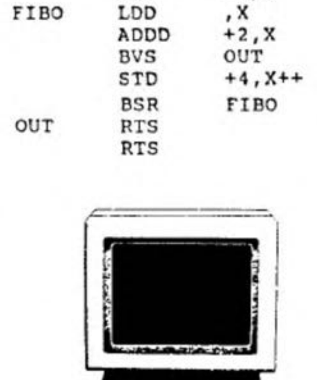
```
        LDD     #1
        STD     INT
        LDU     #ø
        LDD     OP
LOOP    SUBD    INT
        BLO     DONE
        LEAU    1,U
        LDX     INT
        LEAX    2,X
        STX     INT
        BRA     LOOP
DONE    JMP     DONE
OP      FDB     XXX
INT     RMB     2
```

4) If you're reeling from the last problem, here's one that should be easier. The B Register contains a two's complement number. Write a short piece of code to put a zero into the A Register if B is positive, or a -1 into A if B is negative. Hint: The 6809 instructions RELGN and POLITCS are *not* used in the code.

5) This one tests your addressing mode capability. Location $3E00 contains a constant. The X Register contains a value of $3FFF. The Y Register contains $3DC0. The DP register contains a value of $3E. The B Register contains $40. Write down at least four ways to load the A Register with the constant. Assume the instruction to be used is located at $3F00.

6) Here's a tricky one. What does this code accomplish?

```
        LDX     #$4øøø
        LDD     #ø
        STD     ,X
        LDD     #1
        STD     +2,X
FIBO    LDD     ,X
        ADDD    +2,X
        BVS     OUT
        STD     +4,X++
        BSR     FIBO
OUT     RTS
        RTS
```



7) The A Register contains a value of zero through 14. Write a routine to convert the values as follows:

Zero through eight become one through nine

Nine through 15 become 16 through 22

The routine must consist of fewer than 10 instructions.

8) This one shouldn't be too bad. The A Register contains a two's complement number. Divide this number by 8. The result must be valid for either positive or negative numbers. As an example, -100 divided by 8 must produce a result of -12 and +100 divided by 8 must produce a result of +12.

9) A table containing values of zero through 255 starts at BUFFER and ends at BUFEND. What does this code do?

```
LOOP1   LDY     #Ø
        LDX     #BUFFER
LOOP2   LDD     ,X+
        CMPA    ,X
        BLO     NEXT
        LDY     #1
        EXG     A,B
        STD     -1,X
NEXT    CMPX    #BUFEND
        BNE     LOOP2
        LEAY    -1,Y
        BEQ     LOOP1
```

10) This is an assembly language subroutine that's called from a BASIC program. It starts at Location $400, the beginning of the text screen. What does it accomplish? Or does it even run?

```
        LDA     #$39
        LDX     #$6ØØ
LOOP    STA     ,-X
        BRA     LOOP
```

11) At a recent Color Computer User's Group party, there were eight CoCo freaks at a corner table. The following program determines which of these probabilities?

a) The probability that there will be twice as many men as women among the users.

b) The probability that there will be an equal number of men and women among the users.

c) The probability that there will be more men then women among the users.

d) The probability that the user on the left will have an autographed copy of *The Complete Rainbow Guide to OS-9*.

```
        LDX     #Ø
        CLR     CNT
LOOP1   CLRB
        DEC     CNT
        LDA     CNT
        BEQ     OUT
LOOP2   LSLA
        BCC     NEXT1
        INCB
        TSTA
        BEQ     FIN
NEXT1   BRA     LOOP2
FIN     CMPB    #4
        BNE     NEXT2
        LEAX    1,X
NEXT2   BRA     LOOP1
OUT     RTS
CNT     RMB     1
```

12) Finally, the last problem: Which two registers in the 6809 can be added together with one instruction?

The Answers

1) This *should* have been an easy one if you remembered that the 6809 has a multiply instruction called MUL. The code is this:

```
LDA     #23Ø    load A with decimal 23Ø
LDB     #15     load B with decimal 15
MUL             find product in D
```

The result, 3450, is in D after multiplying 230 in A and 15 in B ($E6 in A and $0E in B). Remember the MUL instruction is an unsigned multiply. This means each operand in A and B can be zero through 255 and represents only positive numbers. The maximum product will be 255 times 255 or 65,025 ($FE, $01).

2) The code in the question is reproduced again with comments below:

```
        LDD     OP1     load D with operand 1
        LDU     #Ø      clear quotient
LOOP    SUBD    OP2       subtract divisor
        BLO     OUT       go if residue < Ø
        LEAU    1, U      bump quotient
        BRA     LOOP      loop 'til residue < Ø
OUT     JMP     OUT       dummy
OP1     FDB     XXX       16-bit dividend
OP2     FDB     XXX       8-bit divisor
```

This code is a divide routine that divides a 16-bit operand in D by an eight-bit operand in memory. The quotient result is in U at the end of the divide. Unfortunately, the 6809 doesn't have a divide instruction, so any division has to be accomplished in software. The division here is not a particularly effective division because it divides by repetitive subtraction. If OP1 is 65535 ($FFFF) and OP2 is one, for example, the loop is executed 65536 times! However, the code here is uncomplicated compared to a bit-by-bit divide and it's not bad to use occasionally.

3) The code in the question is reproduced again with comments below:

```
        LDD     #1      integer
        STD     INT     store for subtract
        LDU     #Ø      clear result
        LDD     OP      get square
LOOP    SUBD    INT       subtract 1, 3, 5, etc.
        BLO     DONE      go if residue < Ø
        LEAU    1,U       bump result
        LDX     INT       set next odd integer
        LEAX    2,X
        STX     INT
        BRA     LOOP      loop 'til residue < Ø
DONE    JMP     DONE      dummy
DONE    JMP     DONE      dummy
OP      FDB     XXX       number to find SQR
INT     RMB     2         holds odd integers
```

Still puzzled? This routine finds the square root to the next lowest integer of the number in OP. For example, if OP contained 41,000, the result in the U Register would be 202. The crux of the algorithm is the fact that the square root of a number is equal to the total number of odd integers in the number. The square of 100, for example, is 100 - 1 = 99 - 3 = 96 - 5 = 91 - 7 = 84 - 9 = 75 - 11 = 64 - 13 = 51 - 15 = 36 - 17 = 19 - 19 = 0. The number of odd integers is 10 — 1, 3, 5, 7, 9, 11, 13, 15, 17 and 19.

4) The huge program that solves this problem is shown below:

```
        SEX                 sign extend B into A
```

This instruction is one of the more interesting in the 6809 repertoire, but it does nothing more than "sign extend" the operand in B into the A Register. If the sign bit, Bit 7, of the B Register is zero (positive), zeroes are put into the A Register. If the sign of the B Register is one (negative), all ones are put into the A Register.

In case you're hazy about two's complement notation, remember that it's a way of expressing both positive and negative numbers. An eight-bit register can hold values of -128 through +127 in this format. Positive numbers have the sign bit set to zero and the number in bits 6 through 0 of the Register. A +100 would be 01100100, for example. Negative numbers have the sign bit set to one and the two's complement of the value in bits 6 through 0. A -100 would be 10011100.

Why SEX? Since 16-bits adds and subtracts, and other arithmetic processing is done in the D Register (A and B combined), it's a handy way to make a 16-bit signed number out of eight bits.

5) Some of the possible ways to load A with the contents of Location $3E00 are:

```
LDA  $3EØØ       extended addressing
LDA  -$1FF,X     indexed addressing
LDA  $4Ø,Y       indexed addressing
LDA  $3EØØ       direct page addressing
LDA  B,Y         accumulator offset addressing
LDA  $3EØØ,PCR program counter relative addr
```

A dark horse candidate is:

```
LDD  $3EØØ
```

which loads A, but also clobbers the contents of the B register.

The extended addressing mode specifies the memory address in the last two bytes of the three-byte instruction. The indexed X-addressing example adds the contents of the X Register, $3FFF and -$1FF to get the effective address of $3E00 before the load is done. The indexed Y addressing adds $3DC0 in Y to $40 to get the same effective address. The direct page addressing example computes the effective address by using the contents of DP as the upper eight bits of the address and the second byte of the instruction — $3E, $00 in this case. The accumulator offset adds the contents of index Register Y and the contents of B. The PCR example puts an offset of -$104 in the last two bytes of the four-byte PCR instruction. The effective address is computed by adding the current contents of the program counter $3F04 (the start of the instruction after the LDA) to an offset of -$104 to get an effective address of $3E00.

That wasn't too bad, was it?

6) This problem isn't hard to follow if you write down the results. The X Register points to an open-ended buffer area as shown in Figure 1. Each entry in the buffer is made up of two bytes. Zero is put into the first entry and one into the next to initialize the subroutine. The FIBO loop adds the nth entry to the (n + 1) entry. The result is put into the (n + 2) entry. The pointer in X is then bumped by two. A BSR then calls the FIBO code again. Why the BSR instead of a BRA? No reason other than to demonstrate a simple case of recursion. The FIBO code is called repeatedly until the result is so large that overflow results. In this case the RTS is executed to return from the subroutine. Since there are many levels of BSRs at this point, each return is made to the first RTS repeatedly, much like peeling the layers of skin on an onion.

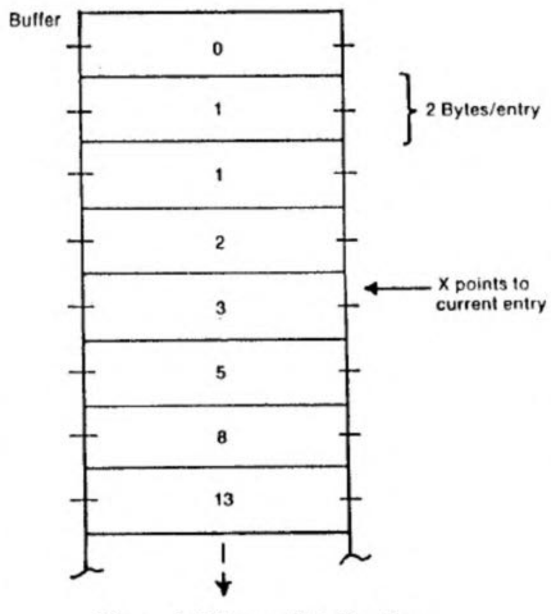


Figure 1: Fibonacci Buffer Area

The results in the buffer area look like this: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2504, 4181, 6765, 10946, 17711, 28657. Each term is computed as the sum of the two preceding it. This sequence is a famous mathematical sequence known as the Fibonacci series, with applications in diverse areas, including computer algorithms.

Overflow occurs when the 23rd term is reached with a result of 46,368. At this point, the stack is 23 levels deep and uses 46 bytes for return addresses! If you run this code, make certain you have enough room for the stack. The annotated code is:

```
FIBOS   LDX     #$BUFFER  start of buffer
        LDD     #Ø        initialize first term
        STD     ,X        in first 2 bytes
        LDD     #1        initialize 2nd term
        STD     +2,X      in second two bytes
FIBO    LDD     ,X          get nth term
        ADDD    +2,X        add nth+1
        BVS     OUT         go if too large
        STD     +4,X        store nth+2
        LEAX    +2,X        bump
        BSR     FIBO        call compute term
OUT     RTS               many happy returns
BUFFER  RMB     1ØØ
```

7) Sorry, I just couldn't resist this one. Admittedly, this application has limited use. However, the code is:

```
ADDA      #1        bump by one
DAA                 decimal adjust
```

The DAA instruction is one you may never have used. It is a "decimal adjust" that allows BCD, or binary-coded-decimal operations. In BCD, the decimal digits of zero through nine are coded in each four bits. Each four-bit chunk, called a "nibble" or "nybble," can only contain values of 00000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, or 1001, and cannot contain the values 1010, 1011, 1100, 1101, 1110, and 1111. The DAA is used after an add or subtract to adjust the binary add back to proper BCD form. If this were not done, the add:

```
       ØØØ11ØØ1  (19 in bcd)
      +ØØ11Ø1Ø1  (35 in bcd)
       --------

would result in

       ØØØ11ØØ1  (19 in bcd)
      +ØØ11Ø1Ø1  (35 in bcd)
       --------
```

```
        Ø1ØØ111Ø  (not 54 in bcd!)

    instead of

        ØØØ11ØØ1  (19 in bcd)
       +ØØ11Ø1Ø1  (35 in bcd)
        --------
        Ø1Ø1Ø1ØØ  (54 in bcd)
```

The adjustment is made by adding +6 to either or both nibbles. In the problem here, the DAA causes the adjustment of the least significant nibble if the result is 10 to 16.

8) The answer is a snap if you know your shifts:

```
ASRA              divide by 2
ASRA              divide by 4
ASRA              divide by 8
```

As you probably know, shifting right by one bit divides by two, by two bits divides by four, and so forth. Dividing by any power of two can be done by the appropriate number of right shifts. However, if the number to be shifted is a signed, two's complement number, a logical shift (LSR, LSL, etc.) won't work. The two's complement number -100 is 10011100. Shifting right one bit logical results in 01001110, or a value of +80. However, shifting right one bit arithmetic results in the correct result of 11001110, a value of -50.

If the value cannot be evenly divisible by a power of two, a negative result is sometimes rounded down by one. The number -105 in two's complement form is 10010111; shifting right arithmetic results in 11001011, or -53.

9) The code here is a bubble sort that sorts the data in buffer into ascending order. Values of 23, 56, 1, 3 and 17, for example, would be sorted into 1, 3, 17, 23 and 56. The bubble sort is a simple sort, but not very fast: Here's the annotated code:

```
LOOP1   LDY     #Ø          load "swap" flag
        LDX     #BUFFER     point to start
LOOP2   LDD     ,X+           get two entries, and bump
        CMPA    ,X            compare pair
        BLO     NEXT          go if in order
        LDY     #1            set swap flag
        EXG     A,B           exchage the two
        STD     -1,X          store swapped pair
NEXT    CMPX    #BUFEND       at end
        BNE     LOOP2         go if not at end
        LEAY    -1,Y        test for swap
        BEQ     LOOP1       go if still unsorted
```

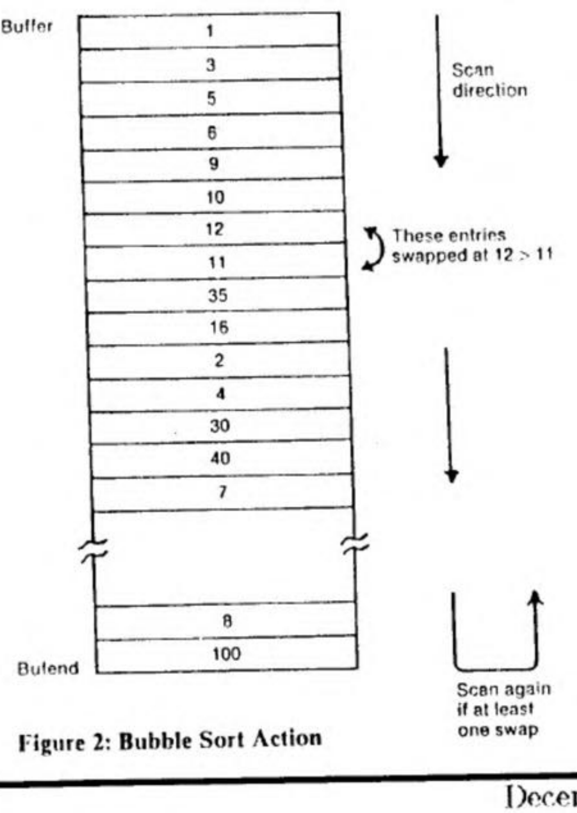

Buffer
1
3
5
6
9
10
12
11
35
16
2
4
30
40
7
8
100
Bufend
Scan direction
These entries swapped at 12 > 11
Scan again if at least one swap

Figure 2: Bubble Sort Action

The bubble sort compares two entries at a time, starting from the top of the table. (See Figure 2.) If the second entry is less than the first, the two bytes are swapped and a "swap flag" is set to one. One complete pass is made through the table and the swap flag is checked. If at least one swap occurred, another pass is made. This process continues until no swaps have been made, indicating that the data is sorted in numerical order. The LEAY -1,Y above is a tricky way to test that Y contains a one. If Y contains a one, a zero results after the LEAY and the zero condition code is set, otherwise the zero condition code is not set.

10) There's no reason why assembly language code can't be located in the text screen area. Of course, it has a tendency to be destroyed by data displayed on the screen! This subroutine can be relocated to the screen by a BASIC program as follows:

```
1ØØ DATA &H86,&H39,&H8E,&HØ6,&HØØ,&HA7,&H82,&H2Ø,&HFC
11Ø CLS
12Ø FOR I=&H4ØØ to &H4ØØ+8
13Ø READ A: POKE I,A
14Ø NEXT I
15Ø DEFUSRØ=&H4ØØ
16Ø A=USRØ(Ø)
17Ø PRINT "DONE!";
18Ø GOTO 18Ø
```

The subroutine is relocated to the first portion of the text screen. You'll see garbage characters fill up the first nine screen bytes. These garbage characters represent the machine language bytes of the assembly language program. The USR0 transfers control to the subroutine and it starts storing ASCII $39 characters to the text screen, starting from the screen end. An ASCII $39 is a "9" character, and therefore, nines start filling up the screen. When a $39 replaces the second byte of the BRA LOOP instruction, however, the branch is done to the current program counter location plus $39, location $442. This location is in the screen area and has been filled with a $39 previously. The $39 is executed as an RTS instruction which causes a return to the BASIC program. The "DONE!" message is then displayed at the screen start. All of which goes to prove that video memory is simply computer memory after all!

11) The annotated code for this problem is:

```
        LDX     #Ø          clear count of men=women
        CLR     CNT         clear 255 to Ø value
LOOP1   CLRB                  clear count of 1s (men)
        DEC     CNT           get next value
        LDA     CNT
        BEQ     OUT           go if 256 times
LOOP2   LSLA                    shift out next bit of 8
        BCC     NEXT1           go if Ø (woman)
        INCB                    bump count of men
        TSTA                    set CC
        BEQ     FIN             stop if no more 1s (men)
NEXT1   BRA     LOOP2           loop, counting men
FIN     CMPB    #4            4 men counted?
        BNE     NEXT2         go if not
        LEAX    1,X           bump count of men=women
NEXT2   BRA     LOOP1         continue for 256 permut'ns
OUT     RTS                 return
CNT     RMB     1           count of 255 to Ø
```

This code determines the probability that the number of men will equal the number of women. A probability of one means that the number of men will *always* equal the number of women. A probability of zero means that the number of men will *never* equal the number of women. The probability here is obviously somewhere in between.

In this problem there are eight users at a table. We're not told whether they are men or women. If we let each bit of a byte represent an individual user, however, we can use the assembly language subroutine to figure out the probability. Men are represented by a 1 bit while women are represented by a 0 bit. Let's try a simpler case first. Suppose that there are only four users at a table. The possible permutations are:

```
ØØØØ  4 women
ØØØ1  3 women, 1 man
ØØ1Ø  3 women, 1 man
ØØ11  2 women, 2 men
Ø1ØØ  3 women, 1 man
Ø1Ø1  2 women, 2 men
Ø11Ø  2 women, 2 men
Ø111  1 woman, 3 men
1ØØØ  3 women, 1 man
1ØØ1  2 women, 2 men
1Ø1Ø  2 women, 2 men
1Ø11  1 woman, 3 men
11ØØ  2 women, 2 men
11Ø1  1 woman, 3 men
111Ø  1 woman, 3 men
1111  4 men
```

The probability here is the number of times that men equal women divided by the total number of cases, or 6/16 + 3/8 + .375.

You can see that the number of times women equal men can be computed by generating the binary numbers from zero to 15 and then counting the number of cases where there are two ones. The same thing can be done for a group of eight users (or any size group). The previous code generates the binary numbers from 00000000 through 11111111 and then counts the cases where the number of ones is four. The result is 70/256, or a probability of .273 that the number of male CoCo users will equal the number of female CoCo users. This little program is great for those Color Computer social gatherings.

12) I'll bet you forgot about the obscure ABX instruction! This instruction takes the contents of B, treated as an unsigned number, and adds it to the X Register, with the result going into X. This is a handy way to increment the X Index Register when it is used as a pointer, which it often is.

Pi Revisited

The column on generating pi drew a lot of interest from readers. First to respond was Carey Bloodworth of Swink, Okla., who noted a more efficient way to generate pi and informed me that his program ran three times as fast as the one in the column. (At that point I had produced a program that was twice as fast as the one appearing in the column, but Carey's sounds faster). If you're interested in this problem, contact Carey at P.O. Box 17, Swink, OK 74761.

Andre Needham of Renton, Wash. sent a pi formula that converges much faster. He also noted that he has memorized pi to 42 places. Bruce Arsenault of Nova Scotia also sent a long letter detailing a faster algorithm.

Michael Frank, 4515 Oak Hill Road A-5, Chattanooga, TN 37416, sent a program that calculates 1000 digits of pi in six minutes by an efficient divide routine. Sounds like Carey and Michael should communicate.

Edward Freeman Yendall of North Fort Meyers, Fla., sent a fascinating letter describing computer processing of a special form of prime numbers called Mersenne primes. His original work (he included a printout) was done in the 1950s on a Burroughs Datatron computer! Edward has now duplicated the work on the CoCo.

If enough readers are interested in problems of this sort, I'd be happy to oblige you in future columns. Let me and RAINBOW know.

Next month, I'll be back with more CoCo assembly language topics. Till then, keep assembling.

MOTOR CONTROL USING CoCoConnection

by Geoff Fiala

16K ECB + CoCoConnection

PREPARATION FOR COCO CONFERENCE has caught up with me therefore I have not had time to continue any further work on the motor control Application. However in preparation for COCO Conference the need arose for a circuit to demonstrate the CoCo Connection controlling 240 volt AC mains appliances such as an electric fan, heater, or lighting appliance.

Such a circuit is shown in fig 1 and provides two Double Pole Double Throw (DPDT) relays. These relays have a coil voltage of 12V DC and contact ratings of 5 AMPS @ 240V AC or 5 AMPS @ 30V DC and can switch mains rated appliances up to 1200 Watts.

In keeping with our previous concept of providing electrical isolation between the Computer and the external switching circuit, opto-isolators are again used to provide isolation between the computer and the relay driver circuit which operates at 12V DC. A LED is also provided to give a visual indication when the relay is switched on.

CIRCUIT DESCRIPTION.

As both of the relay circuits are identical, operation of only one ciruit will be discussed. Operation of the relay RL1 is controlled by the relay driver circuitry consisting of transistors Q1 and Q2 and opto-isolator U1. When a logic "0" (<0.4V) is applied to input IP1, transistor Q1 will turn on and the emittor voltage of 5V. This causes current to flow through the photodiode in the optocoupler U1 via resistor R3 which causes the phototransistor in U1 to conduct and switch on coil and switch the relay contacts.

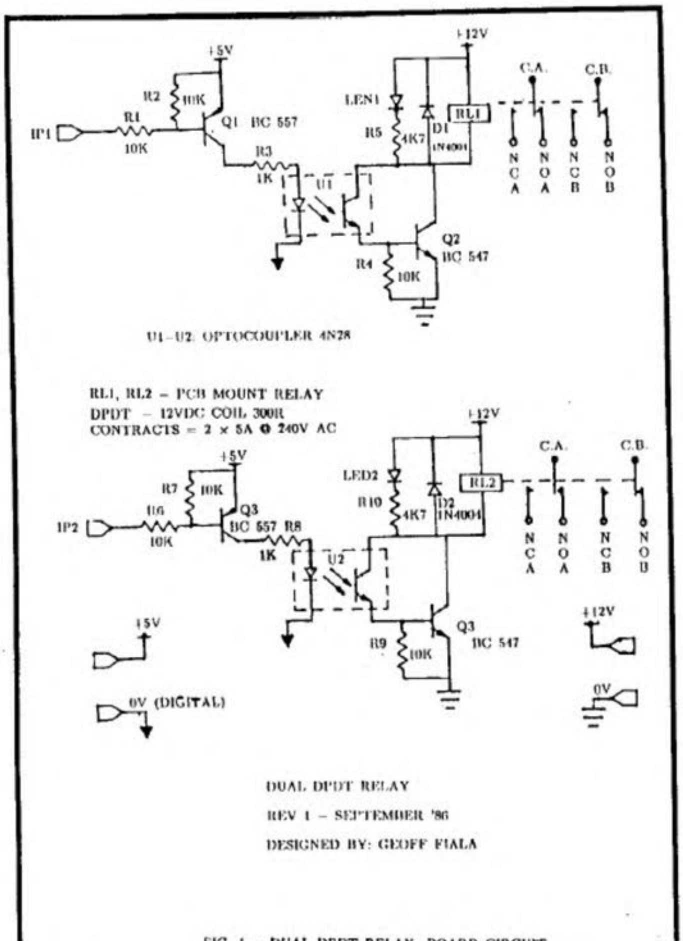
The common contacts of the relay will now be switched from the Normally Open contacts N.O.A and N.O.B to the Normally Closed contacts N.C.A and N.C.B.and LED1 will also light to indicate that the relay has been switched. The diode D1 is used to is used to clamp voltage spikes that occur when the inductiven relay coil is switched OFF, thus protecting the phototransistor in U1 and transistor Q2.

The amount of current required to switch the relays is dependent upon its coil resistance which is nominally 300 ohms and hence the current drawn from the 12V supply will be about 40 milliamps Therefore if both relays are switched on approximately 80 milliamps of current will be required from the 12V supply. A 12V DC plug pack rated at 100 milliamps would prove satisfactory if no other 12V DC source is available.

The relays I used in this circuit as well as all the other parts are available either from Dick Smith Electronics or Jaycar Electronics.

CONNECTION TO THE COCO CONNECTION.

The circuit can connect to any one of the four ports on the Coco Connection and I have chosen to connect it to Port 1A terminals T1 and T2 which correspond to port lines PA1 and PA2 as shown in fig 2.The 5V and 0V connection are taken from terminals T8 and T7 on port 3 connector.



U1-U2: OPTOCOUPLER 4N28

RL1, RL2 - PCB MOUNT RELAY
DPDT - 12VDC COIL 300R
CONTRACTS = 2 x 5A @ 240V AC

DUAL DPDT RELAY
REV 1 - SEPTEMBER '86
DESIGNED BY: GEOFF FIALA

FIG. 1 - DUAL DPDT RELAY, BOARD CIRCUIT

CONNECTION TO EXTERNAL DEVICES.

Fig 3a shows the connections for connecting the unit to switch mains rated appliances. Plugs PL1 and PL2 are 240V AC surface sockets and should be mounted on the case that encloses the relay circuit board. A 5 Amp fuse has also been included to protect the relay contacts from switching excessive currents.

Fig 3b shows the connection for driving a DC motor. When the relay contacts are in the N.O. position the motor direction will be forward while in the N.C. position the motor direction will be reversed. Note that the 0V return for the 12V DC supply must note be connected to the 0V return for the 5V DC supply. These lines must not be connected together at any point in the circuit if the electrical isolation offered by the optocoupler is

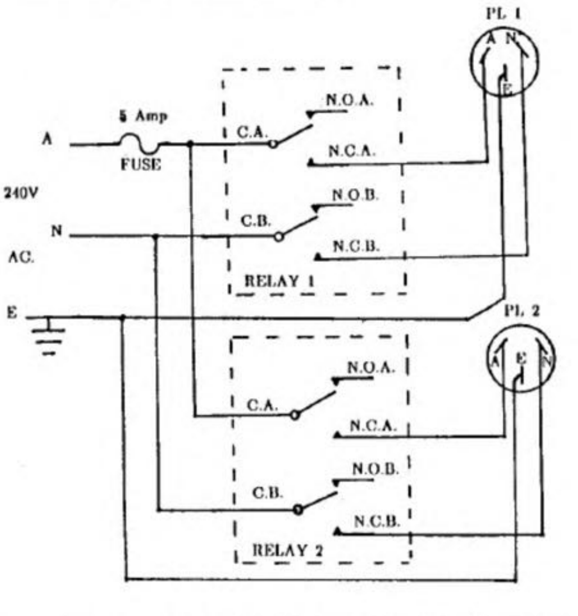


FIG. 3A – CONNECTION FOR SWITCHING MAIN APPLIANCES

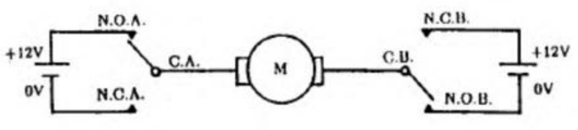


FIG. 3B – DRIVING D.C. MOTOR

to be maintained.

CONTROL SOFTWARE.

The software required to operate the relay circuit involves the following steps:

1) Initialization of Port 1A lines PA0 to PA7 as outputs.
2) Turn the relevant relay on by programming PA0/PA1 to a logic low (writing a "0").
3) Delay for a period of time (if necessary).
4) Turn relevant relay off.

As was discussed earlier the relays are controlled by programming a logic low or "0" condition on port lines PA0/PA1 to turn them on and programming a logic high or "1" on these port lines to turn them off. This equates to poking the correct decimal byte value corresponding to the required logic condition to the address location of port 1A. ie

```
POKE ADDR,VALUE    where ADDR =Port 1A ADDRESS
                         VALUE = Decimal Byte Value.
```

Referring back to the previous article on the motor control circuit, it was mentioned that the decimal byte value to be POKED can be worked out using the following. Each bit position in the byte value is numbered 0 to 7 and is assigned a weighing equal to a power of 2 as shown below:

Bit Position	Weight
D7 = PA7 = 2W	= 128
D6 = PA6 = 2V	= 64
D5 = PA5 = 2U	= 32
D4 = PA4 = 2T	= 16
D3 = PA3 = 2S	= 8
D2 = PA2 = 2R	= 4
D1 = PA1 = 2Q	= 2
D0 = PA0 = 2P	= 1

If the bit position is set to a logic 1 then its weight is to be included and if its bit position is set to a logic 0 then a weight of zero is included for that position.

In our example , after initialization all lines PA0-PA7 will be at a logic 1 and to turn on relay1 PA0 must be set to a logic 0 and the byte value would be as follows:

```
PORT 1A LINES      PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
logic value         1   1   1   1   1   1   1   0
weight value       128+ 64+ 32+ 16+ 8 + 4 + 2 + 0
decimal byte value = 254
```

and the command would be POKE ADDR, 254

The attached sample programme follows the steps outlined above by initializing Port 1A as outputs, and then switches Relay1 on for a specified delay afterwhich Relay2 is switched on for a specified delay. At the end of this delay both relays are switched off.

Although in this example we have only used two relays, duplicate circuits could be constructed to provide another 6 Relays. These could be connected to the remaining lines PA2-PA7 on Port 1A enabling 8 devices to be controlled simultaneously providing control for a lightshow. In this case the 12V DC supply must be able to supply at least 400 milliamps of current to drive all eight relays if they are to be on at once.

Thats about all for this month, next month we will look at a circuit for monitoring input conditions, enabling us to control output settings in response to certain input conditions.

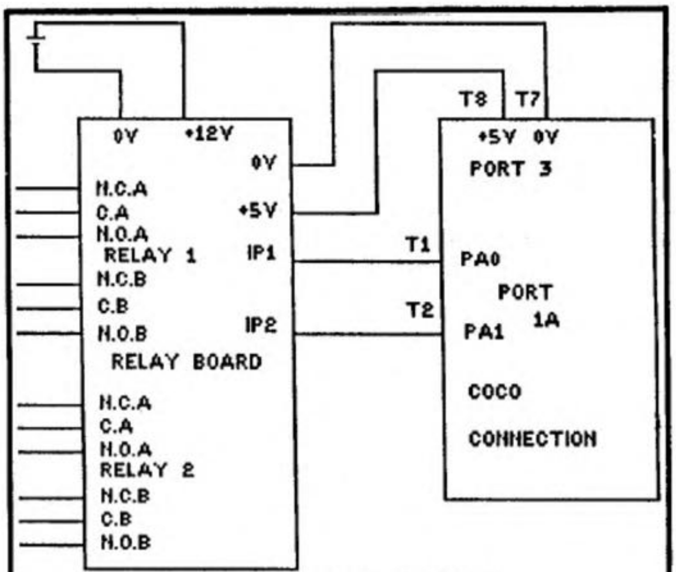


FIG. 2 INTERCONNECTION BETWEEN COCO CONNECTION AND RELAY BOARD.

The Listing:

```
0 GOTO10
1 '**** APPLICATIONS FOR COCO **
2 '***** GEOFF FIALA ***********
3 SAVE"88:3":END
10 REM DRIVER PROGRAMME FOR RELA
Y CONTROL BOARD
20 REM WRITTEN BY GEOFF FIALA -A
UG'86
30 REM INITIALIZE PORT 1A AS OUT
PUTS
40 A=&HFF80:REM PORT 1A  BASE AD
DR
50 POKE A+1,4:POKE A,255:POKE A+
1,0:POKE A,255:POKE A+1,52
60 POKE A,254:REM SWITCH RELAY1
ON
70 FOR I=1 TO 2000:NEXT I:REM DE
LAY
80 POKE A,252:REM SWITCH RELAY2
ON AS WELL
90 FOR I=1 TO 2000:NEXT I:REM AN
OTHER DELAY
100 POKE A,255:REM SWITCH BOTH R
ELAYS OFF.
110 STOP
```

FRICKERS FOLLIES

by Jack Fricker

WELL I FINALLY got my New CoCo 3 and I love it. It finally arrived last friday and since then I have spent quite some time since then playing around with it.

There some things and tips that I would like to pass on to you. One of the most impressive things about it is the fact that it runs TWICE as fast as before and this makes a big diference to things like paint and draw.

Another of the improvements and one of the major ones is to the graphics. The new improvements are in the resoloution department 640x192 although I think the one that will be used the most is the 320x192 16 colour mode because the 640 mode has 4 colours. You can choose these colour from the 64 available.

This also applies to the text screens. There are 3 text screens, one of these is the 32 column one that we are all familiar with, the second is the first of the high density screens. In this mode you can have either 40 or 80 column displays. The next one is the graphics screen itself, this is one of the major changes in the CoCo3.

To do this on the old CoCo you had to do some complicated drawing which was very S-L-O-W. To do this on the 3 there is a command called HSCREEN which is very easy to use.

The use of the two new commands ATTR and PALETTE take a bit of getting used to, but once you get used to them they aren't too bad. One of the things that makes it confusing when you first try to use them is the fact that TANDY (well the designers anyway) decided that you should have 8 foreground and 8 background colours.

The way you use these is there are 16 registers 0-7 for the background and 8-15 for the foreground. To use these registers there is the PALETTE command. The syntax "PALETTE 0,0" will set the value 0 into the background register (0), this will cause the background to be black. If you used "PALETTE 8,63" you would set the value 63 into the first of the foreground registers which is one of the white shades.

Since there are 8 fore and 8 background registers (TANDY call them SLOTS) and the way that you use them is with the ATTR (attributes) Command. This lets you select the registers that you wish to use for the moment. You can only use 2 registers at one time but as soon as you use them you can change to another set.

You may only want to print 1 character or draw one line and then change to another set and then another. You can only use 16 palettes on any one screen but you can change the attributes of any character you wish to use which means that you can set the background to one colour and print 8 characters with the 8 foreground colours then change the background colour and print another 8 characters using the same 8 foreground colours but will look different from the first set of 8 characters.

For those of us who like to know the assembly language locations and the pokes that go with them, here are the ones that I have found out about so far that control the various screens.

To use these registers from BASIC just poke the values that correspond to the colours that you want into the register than you want. These registes are the same ones that are used by the BASIC PALETTE command.

For instance when you use the command PALETTE0,0 what you are doing is poke the value 0 into the register &HFFB0, when you palette 1,63 you are pokeing 63 into the register &HFFB1. If you poke a number larger than 63 into the location then the computer will start again from 0 which means that 0 and 64 are the same colour.

The registers &HFFB0 to &HFFB7 are the background registers, &HFFB8 to &HFFC0 are the foreground registers. These are for the 40 and 80 column displays.

The 32 column text screen that we all know only too well as being black on green and virtually inflexable on the CoCo1 and CoCo2 is changeable on the 3. The registers for this screen on startup are set to the green background and black foreground. These registers are located at &HFFBC for the foreground and &HFFBD for the background.

These correspond to the palette registers 12 and 13.

The cls command has also been changed alond with the other things. The CLS command has also been changed along with the other things. The CLS command now uses the same registers that are used for the background colours &HFFB0-&HFFB7 (cls1-cls8).

These can be changed with the PALETTE command, for instance PALETTE 0,0 will set the CLS1 register to black (0). Remember to add 1 to the palette register to get the cls register.

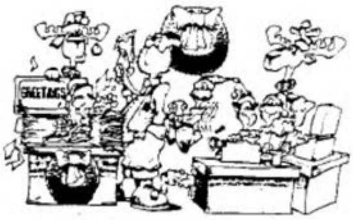
With the 32 column screen you are limited to any foreground and any background, but you are limited to one foreground and one background which is different from the 40 and 80 where you have 8 fore and 8 background registers.

One of the other things that have been changed is the colours of the block graphics in the 32 col mode. These are the same shape and numbers as they were on the earlier systems. The fore and background colours of these blocks can be set to any of the 64 available. These registers are also used to control the cursor colours. The cursor cycles through the 8 background registers so that if you change all the background registers to one colour the cursor will seem to disappear as it wil be the same as the background.

In case you thought that the same old pmode3 and pmode 4 games will be of no more use anymore you can take heart. You CAN in fact change the colours

continued on page 19

KISSable OS-9



Looking At Blue Sky For OS-9 Level II

By Dale L. Puckett
Rainbow Contributing Editor

It's been a long wait — but well worth it. The new Color Computer 3 is simply outstanding. The graphics knock your socks off. With the new windowing capability that is now a part of OS-9 Level II and the promised *Multi-View* visual shell, we should see some really super software soon.

It's Time for Frank to UnFLEX His Bias

As I stared at the outstanding resolution, bright colors and amazing animation on the new Color Computer 3's screen, I couldn't help but remember the debate we had with Frank Hogg in the May 1984 issue of THE RAINBOW. Here's a sample from Frank's article titled, "On OS-9 Matters, Frank FLEXes His Bias."

"First, Tandy did not do a pure OS-9. Close, but not pure. The disk driver will only support single-sided drives and at the maximum, only 40 tracks. To put BASIC09 on the system disk, you have to delete files; otherwise it won't fit. A single-drive user is plumb out of luck. You can't change the stepping speed of the drives either. So, if you have drives better than Tandy's, you will not be able to utilize the extra speed.

"Second, Tandy does not provide a Hi-Res screen with OS-9. You are left to work with the pathetic 16 by 32, uppercase only screen . . . ."

I debated Frank then because we were already publishing articles that told how to work around the limited disk size and upgrade the performance of OS-9 on the CoCo. Level I OS-9 on the original Color Computer 2 was and is a bargain. Level II OS-9 on the Color Computer 3 is a steal.

It's time for Frank to unflex his bias. Tandy has listened to us and eliminated many of our complaints. For example, the disk drivers in OS-9 Level II read information about the physical configuration of your disk drives from the device descriptors like they're supposed to. There is now a Hi-Res screen. In fact, with the new windowing capability of OS-9 Level II, we can view our text on 24 80-character lines and create several windows onscreen containing high resolution graphics or additional text. Since OS-9 is multi-tasking we can have the output from a different task going to each of these windows. The promised *Multi-View* shell makes the windowing features so easy to use that anyone can use them.

Volkswriter Deluxe for OS-9

On the OS-9 68000 scene, MicroTRENDS has announced that *Volkswriter Deluxe* is being ported to OS-9. The program was named the best of 1984 by the editors of *PC Magazine*. It is fast, reliable and easy to use. All commands are logical and concise, requiring the fewest keystrokes per function of any word processor.

Volkswriter Deluxe OS-9 features text merge, note pad, horizontal scrolling, multi fonts, expanded document size, unlimited moves, onscreen tutorial and help keys, special characters and printer installation. It runs on the MicroTRENDS 68000 Jonathan card that plugs into the Apple II.

If this program is written in a high level language maybe the folks at MicroTRENDS will convince Lifetree Software to move it on to the new Color Computer 3. We can only hope.

During his address to the OS-9 Users Group Community Buffet at RAINBOWfest Palo Alto, Computerware's Paul Searby called on software developers "to set as a priority the task of making OS-9 on the CoCo more user friendly." At the time he praised Tandy for releasing products like *Deskmate, Micro Illustrator, OS-9 Profile* and *Robot Odyssey*. He also put his money where his mouth is by supporting Mike Bailey's *The Last Word*.

The great news is that the new Color Computer 3 is going to make it a whole lot easier for programmers. With OS-9 Level II the limited memory problems all but disappear and the new windowing environment makes programming point-and-click software much easier.

So, what would we like to see in our software Christmas stocking next Christmas? Let's dream.

Since OS-9 Level II is made for multitasking and multi-view makes windowing a snap, maybe someone will come up with a management tool like this for OS-9. Think about the possiblities: In one window you can access your rolodex with a click of the mouse. Select a name, click the mouse again and dial the phone. While you are on the phone you can study your "to do" list in another window. Or, open a document window under one of your headlines and take notes from the call. If you want to keep a record of your calls for billing purposes, you can use the program's date and time stamp to put the time at the beginning and end of the notes from the call. When you hang up, click on another window and you can look at your calendar. Select the amount of time in each segment of your day and push another button and *More* creates a subheadline for each period of the day. At this point you only need to add your own assignments or meeting notes.

What else can you do? Select a section of your outline and pick an option from a pull-down menu and it instantly prints a bullet chart you can print directly on an overhead transparency. If you would rather have things printed in neat little boxes, make another menu selection and a neatly formatted organization chart pops onto your screen.

What about the items in your "to do" list that require you to write a letter or memo? Do you need to quit the program and start up a word processing program? No! You simply click twice on a marker in front of your headline and a text window opens and you can create a document of any length. Of course, you can also put a document containing graphics under a headline. Finally, all or any part of your outline can be exported to any other word processing program.

This is advanced idea processing combined with desktop publishing. I can't wait to run it on the new CoCo 3.

UNIX Comments From Kevin Kuehl

Kevin Kuehl of Valparaiso, Indiana, has contributed many excellent programs to these pages. Recently, after we published a description of Brian Lantz's *KShell*, Kuehl wrote to let us know that the quote expansion feature Brian added was not pure UNIX. In fact, he quoted from a paper and book by Steven Bourne, author of the famous *Bourne Shell.*

"There are three quoting systems used on UNIX systems, the single quote, the double quote and the escape character," Bourne wrote. "The single quote transfers whatever is inside the pair verbatim to the program. The double quote transfers everything but single quotes and escape characters. The escape character transfers the next single character to the program."

Kuehl, calling for correct information about UNIX in RAINBOW also gave examples from a book named *The UNIX System*:

```
echo \?yields?
echo \\yields\
echo xx'***'xx
  yieldsxx***xx
echo The date is 'date'
.yieldsThe date is date.
```

Complete Rainbow Guide to OS-9 Brings Hackers Together

Kevin Darling has sent us a great story from North Carolina. It seems that Steve Croom was having problems with his hard disk drive. When Darling found out that Croom, a Navy man stationed on the West Coast, was from North Carolina, too, and was planning to come home on leave soon, he suggested that Croom bring the drive and he would try to fix it.

"As I prepared to leave for the airport I realized we had no idea of what each other looked like," Darling said. Trying to think of an identifying object, I grabbed *The Complete Rainbow Guide to OS-9* on the way out. When I reached the arrival gate, I simply sat down and laid the book on the table next to me. Sure enough, Steve spotted the book.

"We still can't believe that we never met before, except on the OS-9 SIG, and he gave me a $700 hard disk drive on faith. The only ID he ever saw from me at the airport was *The Complete Rainbow Guide to OS-9*."

We also received a short thank you note from Eric W. Tilenius. We mentioned several months ago that he was looking for some talented programmers.

"Thanks to you, I've been in touch with some very talented and interesting people," Tilenius said. He promised us one of the very first copies of *Print Shop* when it comes out. Should be fun!

Databases Are Gold Mines

Since we're talking about telecommunicating, here's a tip. The databases managed by the many Special Interest Groups (SIGs) on most of the commercial services are a gold mine. For example, the OS-9 section on RAINBOW's CoCo SIG on Delphi is really beginning to take shape.

Check out Steve Bjork's fantastic bouncing ball demo program. You'll have a fantastic demo to show off your Color Computer if you download the binary code. And, if you download the source code, you'll see how Steve makes magic with graphics and animation. If you need a good screen-oriented editor, download the *Dolphin Technology Text Processing System* from the CoCo SIG's OS-9 database. It's written in C and has many machine language routines to speed up crucial sections of the program. If you contribute more than $20 to the author you'll receive Version 2.00 of this editor. Version 2.00 has been expanded and includes merge, block duplicate, printer output and display memory, as well as search and replace features.

OS-9 Level II Program Development System Has Screen Editor

When you buy OS-9 Level II for $79.95, you get BASIC09 with it. Now that's a bargain for high level language programmers.

There is also a program development system available that features a screen editor, assembler and linker.

The screen editor is the SCRED editor that has been available for other OS-9 Level II and 68000 systems for several years.

SCRED is straightforward and easy to use. It can be used to create or modify any text file. It lets you do either line or character oriented editing.

SCRED has three modes, Command, Edit and Insert. A set of commands is available in both the Command and Edit modes. The top line of the terminal displays the line number your cursor is on, the column number, the name of the file being edited, the amount of space left in your buffer and the mode you are working in. If your file is larger than the memory buffer used by SCRED, simply issue the (More) command when you want to write out the section of text you are working on and read in another. SCRED displays 80 characters on the screen, although it can accept lines up to 256 characters long.

> *". . . pick an option from a pull-down menu and it instantly prints a bullet chart you can print directly on an overhead transparency."*

So far I have only heard of two drawbacks to the Color Computer 3 as far as OS-9 Level II is concerned. The first is the fact that the Tandy Sound/Speech Cartridge does not work with it because OS-9 always runs at 1.79 MHz and the Sound/Speech Cartridge can only deal with .89 MHz. The good news is you can add a switch to bypass the XOR gate in the Sound/Speech Cartridge and it will work at the higher clock speed. The other drawback is that a modification must be made to the expansion interface to allow it to run. You need to take it to a computer center to have it modified.

Larson's *SysGo* Revisited

In May of this year we published an alternate *SysGo* module for OS-9 Level I, Version 1.00 or Version 1.01. Also that month, we discussed various techniques you could use to force OS-9 to start up in a RAM disk. David Curtis, of Heath, Ohio, put the tips into action and sent us the completed product. It is listed here. Curtis also submitted a simple utility that takes the place of the Microsoft BASIC CLS command. Interestingly enough, we received another CLS program from John Bowden of the U.S. Navy's COMTHIRDFLLT N-2 in Pearl Harbor, Hawaii. Bowden's code shows how to fork a new process since it clears the screen by calling the OS-9 display command. If you have both display and CLS loaded in memory, this method is fast, too.

Version 2.00

Along with his program listings, Curtis sent a question. "Why bother with OS-9 Level I, Version 2.00?" he asked. "The 80-column driver won't run my *WordPak I*. The disk driver won't handle double-sided drives. The hard drive Tandy supports is out of sight price-wise. The config program is nice and permits easy creation of a custom system disk. That's about it."

I guess the answer to that question revolves around what you are going to do with your computer. If you plan on running all the new programs coming out for OS-9, you'll most likely need the latest version of the operating system. Some programs operate across a number of operating systems; many won't. Almost all of the new Tandy software products require OS-9 Level I, Version 2.00 to operate.

Starting BASIC09

Ray Preston of Rarotonga in the Cook Islands was one of a group of recent writers wanting to know how to get BASIC09 up and running.

Here's the problem. Tandy did not put BASIC09 in the CMDS directory on the production disk. They put it in the root directory. If you have two disk drives, leave your system disk in Drive /d0 and plug the BASIC09 disk into Drive /d1, then type:

```
OS9: chx /d1
OS9: Basic09
```

If you have a single-drive system, take out the system disk and plug the BASIC09 disk in Drive /d0, then type:

```
chx /d0
basic09
```

This should put you on the air once and for all.

OS-9 Software Sourcebook

The OS-9 Software Sourcebook written by Phyllis Casel can help you find that software you're looking for. It's available from Microware Systems Corporation, 1866 N.W. 114th Street, Des Moines, IA 50322, (515) 224-1929. Give them a call.

If all goes well, by the time I write the next installment, we will be settled in here in New York City and will have had our hands on the new CoCo 3 for several weeks. Now if we can just get our hands on Level II OS-9! □

Listing 1: sysgo

```
                                    ifp1
                                    endc

ØØØD                    c.cr        equ    $d
ØØØØ 87CDØØ9A                       mod    eom,name,$C1,$81,start,$ØØC8
ØØØD 53797347           name        fcs    /SysGo/
ØØ12 Ø6                             fcb    6
ØØ13 2F523Ø             newdir      fcc    "/RØ"
ØØ16 ØD                             fcb    c.cr
ØØ17 2F523Ø2F           newexe      fcc    "/RØ/"
ØØ1B 434D4453           cmds        fcc    /CMDS/
ØØ1F ØD                             fcb    c.cr
ØØ2Ø 5368656C           shell       fcc    /Shell/
ØØ25 ØD                             fcb    c.cr
ØØ26 53746172           startup     fcc    /Startup -p/
ØØ3Ø ØD                             fcb    c.cr

ØØ31 55ØØ7412           initdat     fcb    $55,Ø,$74,$12
ØØ35 7FFFØ3B7                       fcb    $7F,$FF,$Ø3,$B7
ØØ39 FFDF7EFØ                       fcb    $FF,$DF,$7E,$FØ
ØØ3D ØC                             fcb    $ØC

ØØØD                    idatlen     equ    *-initdat

ØØ3E 3Ø8C55             Start       leax   <rti,pcr
```

```
ØØØ27    ØØ41 1Ø3FØ9               os9    f$icpt
ØØØ28    ØØ44 3Ø8CEA               leax   <initdat,pcr
ØØØ29    ØØ47 CEØØ71               ldu    #$ØØ71
ØØØ3Ø    ØØ4A C6ØD                 ldb    #idatlen
ØØØ31    ØØ4C A68Ø       movidat   lda,x+,pcrt,$FØ,pcr
ØØØ32    ØØ4E A7CØ                 sta    ,u+
ØØØ33    ØØ5Ø 5A                   decb
ØØØ34    ØØ51 26F9                 bne    movidat
ØØØ35    ØØ53 3Ø8CC5               leax   <cmds,pcr
ØØØ36    ØØ56 86Ø4                 lda    #4
ØØØ37    ØØ58 1Ø3F86               os9    i$chgdir
ØØØ38    ØØ5B 3Ø8CC2               leax   <shell,pcr
ØØØ39    ØØ5E 338CC5               leau   <startup,pcr
ØØØ4Ø    ØØ61 CCØ1ØØ               ldd    #$Ø1ØØ
ØØØ41    ØØ64 1Ø8EØØ15             ldy    #21
ØØØ42    ØØ68 1Ø3FØ3               os9    f$fork
ØØØ43    ØØ6B 2527                 bcs    infloop
ØØØ44    ØØ6D 1Ø3FØ4               os9    f$wait
ØØØ45    ØØ7Ø 3Ø8CAØ               leax   <newdir,pcr
ØØØ46    ØØ73 86Ø3                 lda    #3
ØØØ47    ØØ75 1Ø3F86               os9    i$chgdir
ØØØ48    ØØ78 3Ø8C9C               leax   <newexe,pcr
ØØØ49    ØØ7B 86Ø4                 lda    #4
ØØØ5Ø    ØØ7D 1Ø3F86               os9    i$chgdir
ØØØ51    ØØ8Ø 3Ø8C9D     restart   leax   <shell,pcr
ØØØ52    ØØ83 CCØ1ØØ               ldd    #$Ø1ØØ
ØØØ53    ØØ86 1Ø8EØØØØ             ldy    #$ØØØØ
ØØØ54    ØØ8A 1Ø3FØ3               os9    f$fork
ØØØ55    ØØ8D 25Ø5                 bcs    infloop
ØØØ56    ØØ8F 1Ø3FØ4               os9    f$wait
ØØØ57    ØØ92 24EC                 bcc    restart
ØØØ58    ØØ94 2ØFE       infloop   bra    infloop
ØØØ59    ØØ96 3B         rti       rti
ØØØ6Ø    ØØ97 E1ED78               emod
ØØØ61    ØØ9A            eom       equ    *
ØØØØØ error(s)
ØØØØØ warning(s)
$ØØ9A ØØ154 program bytes generated
$ØØØØ ØØØØØ data bytes allocated
$1948 Ø6472 bytes used for symbols
```

Listing 2: cls

```
ØØØØ1          * Clear Screen Utility
ØØØØ2          * by David Curtis / Heath, OH
ØØØØ3          * To use cls load cls; link cls  and
ØØØØ4          * type cls anytime you want to clear the screen
ØØØØ5
ØØØØ6
ØØØØ7                                 nam    cls
ØØØØ8                                 ifpl
ØØØ1Ø                                 endc
ØØØ11
ØØØ12    ØØØØ 87CDØØ28                mod    endpgm,name,type,revs,start,size
ØØØ13
ØØØ14    ØØØD 636CF3       name       fcs    /cls/
ØØØ15    ØØ11              type       set    prgrm+objct
ØØØ16    ØØ81              revs       set    reent+1
ØØØ17 D ØØØØ               char       rmb    1
```

```
ØØØ18   ØØ1Ø              size     equ    *
ØØØ19
ØØØ2Ø   ØØ1Ø 5F           start    clrb
ØØØ21   ØØ11 86ØC                  lda    #$ØC
ØØØ22   ØØ13 3Ø8DFFE9              leax   char,pcr
ØØØ23   ØØ17 A784                  sta    Ø,x
ØØØ24   ØØ19 1Ø8EØØØ1              ldy    #$Ø1
ØØØ25   ØØ1D 86Ø1                  lda    #$Ø1
ØØØ26   ØØ1F 1Ø3F8A                os9    i$write
ØØØ27   ØØ22 1Ø3FØ6                os9    f$exit
ØØØ28   ØØ25 C33547                emod
ØØØ29   ØØ28              endpgm   equ    *
ØØØ3Ø                              end

ØØØØØ error(s)
ØØØØØ warning(s)
$ØØ28 ØØØ4Ø program bytes generated
$ØØØ1 ØØØØ1 data bytes allocated
$18DØ Ø6352 bytes used for symbols
```

Listing 3: alternat.cls

```
ØØØØ1           * Alternative version of CLS that forks a process
ØØØØ2           * to use the DISPLAY utility to clear your screen.
ØØØØ3
ØØØØ4                               nam    cls
ØØØØ5                               ifp1
ØØØØ7                               endc
ØØØØ8
ØØØØ9     ØØØØ 87CDØØ3E             mod    clsend,name,type,revs,start,size
ØØØ1Ø     ØØØD 636CF3     name      fcs    /cls/
ØØØ11     ØØ1Ø Ø2         edition   fcb    2
ØØØ12     ØØ11            type      set    prgrm+objct
ØØØ13     ØØ81            revs      set    reent+1
ØØØ14 D  ØØØØ                       rmb    3ØØ
ØØØ15 D  Ø12C             size      equ    .
ØØØ16
ØØØ17     ØØ11 7368656C   shlstr    fcs    /shell/
ØØØ18     ØØ16 6469737Ø   cmdstr    fcc    /display ØC/
ØØØ19     ØØ2Ø ØD                   fcb    $ØD
ØØØ2Ø
ØØØ21     ØØ21 3Ø8DFFEC   Start     leax   shlstr,pcr
ØØØ22     ØØ25 338DFFED             leau   cmdstr,pcr
ØØØ23     ØØ29 1Ø8EØØØA             ldy    #$ØA
ØØØ24     ØØ2D 86Ø1                 lda    #1
ØØØ25     ØØ2F 5F                   clrb
ØØØ26     ØØ3Ø 1Ø3FØ3               os9    f$fork
ØØØ27     ØØ33 25Ø3                 bcs    error
ØØØ28     ØØ35 1Ø3FØ4               os9    f$wait
ØØØ29     ØØ38 1Ø3FØ6     error     os9    f$exit
ØØØ3Ø     ØØ3B EA7E5C               emod
ØØØ31     ØØ3E            clsend    equ    *
ØØØ32                               end

ØØØØØ error(s)
ØØØØØ warning(s)
$ØØ3E ØØØ62 program bytes generated
$Ø12C ØØ3ØØ data bytes allocated
$18FD Ø6397 bytes used for symbols
```

The Third One's the Charm

By Mark Siegel

Here we are at the start of a new era in the saga of the Color Computer. The Color Computers 1 and 2 have been great machines. The proof of this is their longevity and popularity. With the introduction of the Tandy Color Computer 3, a new age dawns for the home computer. This new computer can produce startling graphics and run many programs at the same time, and allows for a better human-to-computer interface. Of course, the best part is that it's priced so everyone can afford to buy one.

Let's get down to the facts and figures. First, the Color Computer 3 comes with 128K of RAM and can be expanded to 512K. The graphics *capabilities* are 640 by 225, although only a maximum of 640 by 192 is supported. Up to 16 colors can be displayed on the screen at the same time, and there are 64 different colors to choose from. Both 40-by-24 and 80-by-24 character screens are supported. In addition, these screens have attributes, eight foreground and eight background colors, underlining and blinking. The hardware is capable of displaying more lines of text. Keep in mind, however, most TV sets cannot display these extra lines. The Color Computer 3 can run at .89 MHz, like its predecessors. A new 1.79 MHz clock rate is provided. This additional speed allows the Color Computer 3 to outrun most of the PC compatibles, and all of its competition in this price range.

The 6809 CPU has a 16-bit program counter, which means it can only address 64K at any one time. Yet, the Color Computer 3 can have 512K of RAM in it, and the 6809 can execute a program from all the RAM. This is done with a device called an MMU (Memory Management Unit). The MMU is also referred to as a DAT (Dynamic Address Translator). Sounds pretty fancy. Well, it's really quite simple. There are two sets of eight-DAT registers, one for a system mode and one for a user mode.

A memory address has a 16-bit binary value. Each bit, starting with the most significant bit, selects either the upper or lower section of memory. For example, if the highest bit in the 16-bit address is on, the processor will only select memory in the upper 32K of address space. The three most significant bits break memory into 8K blocks throughout the map. These three bits produce eight combinations. This set of combinations point to the eight DATs. Each DAT register can be made to point to an 8K block in the 512K memory map. By changing these DAT registers, the 6809 can address any location in the ½-meg address space. Having two sets of DATs allows the 6809 to switch memory maps very quickly. An operating system like OS-9 Level II changes the user's memory configuration during an interrupt, and allows for many programs and/or programs longer than 64K to run within the system.

Now that we have provided the 6809 CPU with a way to address more memory, we can look at how the superb graphics use it. First, let's look at all the graphics modes.

Mark Siegel is the product manager of Color Computer and home entertainment products for Tandy Corporation.

Figure 1

ATTR	Displays character attribute, foreground, background, blink and underline
HBUFF	Allocates space outside of BASIC's program area for Hi-Res GET/PUT buffers
HCIRCLE	Draws a circle on Hi-Res screen
HCLS	Clears Hi-Res screen
HCOLOR	Sets foreground and background on Hi-Res screen
HDRAW	Draws an object on Hi-Res screen described by a string
HGET	Copies an area on Hi-Res screen to a buffer
HLINE	Draws a line on Hi-Res screen
HPAINT	Fills an area on Hi-Res screen
HPRINT	Displays text on Hi-Res graphics screen
HPUT	Displays a block stored in a buffer on Hi-Res screen
HRESET	Resets a point on Hi-Res screen
HSET	Sets a point on Hi-Res screen
HSCREEN	Selects Hi-Res mode for display
HSTAT	Returns character location, character and attribute
LOCATE	Positions cursor on a screen
LPOKE	Pokes a byte into any location in the 512K map
ON BRK GOTO	Causes the BREAK key to be trapped
ON ERR GOTO	Causes an error to be trapped
PALETTE	Changes color registers
WIDTH	Selects 32-, 40- or 80-column display
BUTTON	Returns status of firebutton
ERLIN	Returns the line number in which an error occurred
ERNO	Returns number of the error
HPOINT	Returns a point on Hi-Res screen
LPEEK	Peeks a location in the 512K map

Compatible Modes

64 by 32	8 color
128 by 100	4 color
128 by 192	4 color
256 by 192	2 color

New Modes

160 by 192	16 color
320 by 192	4 color
320 by 192	16 color
640 by 192	2 color
640 by 192	4 color

Most of the games written for the Color Computer use the 128-by-192 four-color mode; this mode takes up 6K of memory. And, of those games, most use two graphics screens for a total of 12K. The 320-by-192 16-color mode uses 32K of memory for just one screen. A game that requires two screens of video uses 64K. That is the maximum amount of memory that you could have in the old Color Computer. You can write some really fine looking programs with this kind of color and resolution. However, to do a program like *Zaxxon* in this kind of resolution would take a lot of CPU time to move such a large section of memory. The new computer has been equipped with both vertical and horizontal smooth scrolling. What this does is allow the video screen to act like a window on top of a larger screen. Thus, we get the effect of moving large amounts of memory without using very much CPU time. It is also important to note that all graphics modes use contiguous memory. This makes address calculations simpler and faster.

The 16-color registers can be set to any of 64 colors. The primary set of colors consists of red, green and blue. Each color has up to three shades. By mixing the color and shades together you produce the effect of shading and contouring of objects. This allows for anti-aliasing (non-stair-step lines), and many other state of the art display techniques. You can set as many of the color registers to the same color as you want. This allows you to hide objects on the screen and have them appear by just changing their palette color. Even more dramatic effects can be produced by changing the palettes continuously, as in producing a flickering fire on the screen.

Another area addressed by the new computer is the interrupt system. Special interrupt control registers have been incorporated to allow the processor to spend far less time in the interrupt service routine. This hardware allows interrupts to be generated by the keyboard, joystick button, serial port, cartridge port, V-blank and a programmable interrupt generator. These interrupts can be vectored to either the IRQ or FIRQ. The programmable timer interrupt has a 12-bit counter and can use either the 15,000 Hz or a 70-ns clock. The programmable interrupt timer can be used to aid the processor in producing sound through the six-bit D/A converter or to provide a clocking system for the "bit-banger" serial port.

Those of you who like good, crisp hardware-generated text are going to love the CoCo 3. As stated earlier, we have 40- and 80-column text with attributes. In addition, there are 32 international characters in the character set. The programmable timer generates the blink rate for the blinking attribute, color registers 0 to 7 produce foreground colors and registers 8 to 15 produce the background colors. Add in the underline and control of border colors and you can produce some pretty attractive screens. However, you will want a CM-8 color monitor. The CM-8 is not another PC-compatible RGBI monitor, but rather an analog Hi-Res RGB monitor.

Figure 2

Select font	You may use different font styles
Point	Plot a point
Line	Draw a line
Circle	Draw a circle
Get block	Copy a block into a system buffer
Put block	Copy a block from system to screen
Fill	Paint the screen
Use logic	And, or, xor, no logic
Use pattern	Apply pattern to command
Ellipse	Draw an ellipse
Arc	Draw an arc
Create a window	
Use overlay window	
Proportional	Proportionally-spaced text
Bold text	
Invert text	
Underline text	
Download font	
Download buffer	
Scale on/off	
Protect on/off	

Each joystick port now has two firebuttons. The resolution of the joystick is still 64 positions internally. However, with the Hi-Res joystick adapter and OS-9 Level II, you can get 640 true analog positions.

With all these features added to this new machine, it still maintains compatibility with its predecessor. The exception is software that uses the VDG/SAM semi-graphics modes or undocumented BASIC ROM calls. Most programs should work if they worked on the Color Computer 1 and 2. Third-party products that follow these rules should work:

1. Use only documented ROM calls
2. Do not write to an address above $FE00
3. Make sure the map is selected for 16K internal and 16K external.

To top off the Color Computer 3, more power has been added to the BASIC ROM in the computer. These range from support of the new graphics to error handling. Figure 1 is a summary of the newly added BASIC commands.

These new features work with the Disk system, giving the user a new Disk Extended BASIC. For compatibility, BASIC still runs at .89 MHz. You will find the 26 new commands both useful and fun.

OS-9 Level II From Microware Systems

The new OS-9 comes with a windowing system. This system allows you to have a multi-user system at one display and keyboard. Until now the only way you could have more than one program requiring keyboard input and display output was to attach a terminal to the Color Computer. Now, you can tell OS-9 to open another terminal on the *same screen* or a different screen. The windowing system allows for multiple screen and multiple resolutions, and all active at the same time. To my knowledge there is no other system at any price that has this capability. In graphics modes, the system allows the features

continued on page 61

BITS AND BYTES OF BASIC

BASIC09 on the CoCo 3

By Richard A. White

The new Color Computer 3 is here and it's what many of us had been waiting for in a new CoCo. As expected, Level II OS-9 is provided which can fully utilize a 512K machine. The Level II OS-9 package includes BASIC09 rather than an assembler and sells for a modest $79.95.

While all software that runs on a CoCo 2 will run on the new machine (provided undocumented ROM calls are not used), these programs run in the CoCo 2 mode and do not use the enhancements in the CoCo 3. Current BASIC09 provides some graphics support for CoCo 2 modes. Level II BASIC09 is expected tŏ support the new graphics modes. This, coupled with the fact that BASIC09 comes with Level II OS-9, should drastically increase its popularity. Up to now, there has been little incentive for the more casual user to buy BASIC09. The only available software for BASIC09 comes from the OS-9 Users Group. Because of the small group of owners, there has been no commercial BASIC09 software. This may change.

BASIC09 has always had major advantages over Extended Color BASIC. Two of these are speed and programming ease. Provided adequate graphics commands are available in the new version, it will be possible to write game programs that otherwise would need to be written in assembly language or C. This is not to say BASIC09 rivals machine language in speed; it doesn't. But it is much faster than Extended Color BASIC or, for that matter, GW-BASIC running on MS-DOS machines. Couple this with the 1.7 MHz microprocessor speed, and all sorts of programming doors open.

In most respects, BASIC09 is a programmer's dream. First, it is very modular. Separate procedures may be saved separately and loaded as needed from the disk. When the procedure has been used and is no longer needed, it can be killed, freeing memory for other procedures.

Though the current BASIC09 editor is a line editor, it does do syntax checking as each line is entered. I will put up with a line editor just to get this feature. Further, other checks are made when you leave the edit mode. Forget a NEXT, for example, and you are told.

The Debug mode is another highly appreciated feature. The syntax and other program details may be correct and the darned thing still won't work. With Debug, you can single step through the program and really see what is happening.

In the February RAINBOW (Page 231), I talked about how to get set up to use BASIC09. Those instructions may not necessarily apply to the Level II version. Still, if you are just getting started with BASIC09, you may want to study that column. In March, I discussed what happens when you first get BASIC09 up and running (Page 226). I'm going to summarize some of that material, but you may want to read that article, too.

The BASIC09 distribution disk comes with four files. At minimum, two of these, basic09 and runb must be copied to the CMDS directory of your system disk. We'll worry about the other files later. With the CMDS as your execution directory, type `EX BASIC09 #10K`. BASIC09 loads and you are in its system mode. The #10K provides 10K bytes of workspace. If you don't do this, BASIC09 defaults to a 4K byte workspace of which a little over 1K is allocated for BASIC09's own use. You can change the workspace size from system mode. Type `mem 10000` to get 10K bytes. Type `mem` and available workspace memory is displayed. Available memory for the workspace depends on which procedures are loaded when you boot OS-9. I can use as much as 14K and still have some memory left outside BASIC09 for loading and using disk-resident OS-9 utilities.

You can do a number of things in the system mode. Type `e` or `edit`, and a procedure name, and you enter the edit mode. This is the line editor which permits you to write a program module or edit one whose source code was loaded while you were in the system mode. Once you have entered and edited your program, you will want to run it to see if it works. While in edit mode, type `q` and press ENTER to return to system mode. Now you can type `run` and the procedure name to run the program. Note that in BASIC09, programs or program modules are called procedures.

Despite the syntax checking the editor does as you enter program lines and the checking done when you quit the editor, there may still be problems in your program. Some of these BASIC09 will find as the program runs. In this case, it puts you into Debug mode and displays the offending line along with an error message. If you have printerr in your boot and the file of error messages on your system disk, you get an error number and message. Otherwise, you get only an error number which you can look up in the BASIC09 manual. At this point, make sure you understand which line has offended BASIC09 and the type of error, and press ENTER to return to the system mode.

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

Whenever you are in system mode you can press ENTER and a directory of BASIC09 procedures in your workspace is displayed. An asterisk (*) appears to the left of the last active procedure. This directory also lists the size of each procedure in the workspace, its data space requirements and available workspace memory. Since none of the procedures is running at this time, no data space is allocated. The situation arises when the data space needed by a procedure is larger than available workspace memory. BASIC09 flags this by printing a question mark after the data space requirement for the procedure.

You cannot run a procedure when there is insufficient data space. It is important that you be able to run the source code version of your program from BASIC09 system mode because Debug mode is available. There are a number of strategies available to make this happen. One is to enlarge the workspace to use all available memory. A second is to keep procedures small and load them only as needed. A third is to limit data memory requirements until the procedure is totally debugged. This third option depends on how you dimension variables. We'll discuss that in a later column.

Now it's time to write a short program and get some hands-on experience. With the new CoCo 3, it is going to be fun to measure just how much faster it is at the 1.7 MHz clock rate. There are lots of possible benchmark measures, but one that is generously documented ,is the *Sieve of Eratosthenes* program to calculate prime numbers. Versions of the program in various languages including C and PASCAL, along with execution times on various microcomputers were published in "Eratosthenes Revisited: Once More Through the Sieve," by Jim Gilbreath and Gary Gilbreath, Page 283, *Byte Magazine*, January 1983.

In BASIC09 system mode, type `e sieve`. This puts you into edit mode, ready to type in the program. You know you are in the editor because the B: prompt of system mode is replaced with an E: prompt. The cursor sits in the space after the colon. Chapter 4 of the BASIC09 manual gives a good description of how to use the editor.

The first character entered after the E: prompt is the command character. BASIC09 source code may be line numbered or not. The ability to eliminate line numbers is one of the language's major strengths. A space typed at the control character position permits entry of any characters that follow as a string. When the ENTER key is pressed, BASIC09 attempts to compile the preceding string to a condensed form known as I code. If it can, all is well and the E: prompt returns for entry of the next line or a control character. If the line cannot be compiled, it is reprinted on the screen with an arrow pointing to the suspected error point along with an error message. At minimum, the message may look like this example from the manual: `01FC ERR #43`.

The 01FC is the number of bytes from the beginning of the procedure to the error that was interpreted to be #43.

To illustrate, let's do a step-by-step example. BASIC09 is loaded and we are at the B: prompt:

```
Basic09
ready
B:
```

Let's get into edit mode and call our procedure sieve:

```
B:e sieve
PROCEDURE SIEVE
*
E:
```

A common error I make is to forget the ending quote when I print a string. Here is what happens when I make that mistake:

```
E: print "Missing quote
print "Missing quote
    ^
Error #041
- No Ending Quote
*0000 ERR print "Missing quote
E:
```

When a syntax error like this is detected, the cursor is positioned just before the offending line in the procedure as indicated by the '*'. To correct the error, type a `c` in the control character position. Follow it with a delimiter character which can be a slash or any punctuation character. Next comes the character(s) to change. Here we need to add a character, so enter an `e` to position where the added character is to go, following with a delimiter matching the first one and then `e"`. Here is how it looks and the result:

```
E:c/e/e"
  print "Missing quote"
E:
```

The cursor is now just past the line in the program. If you go back to look at the line again by typing a dash as a control character, it looks different:

```
E:-
*0000   PRINT "Missing quote"
```

Once the line is right, BASIC09 compiles it. On going back, the line was decompiled and the keyword `PRINT` was capitalized. It is good practice to enter programs in lowercase. Then, when you go back over the code or list it to the printer, only the keywords are capitalized. The program will be easier to follow. Now issue a control character 'd' to delete the line:

```
E:d
```

Now enter the following *sieve* program.

```
PROCEDURE sieve
DIM sizeof:INTEGER
sizeof:=819Ø
BASE Ø
DIM flags(8195):BOOLEAN
DIM i,prime,k,count,iter:INTEGER
PRINT "1Ø iterations"
SHELL "date t"
FOR iter:=1 TO 1Ø
count:=Ø
FOR i:=Ø TO sizeof
flags(i):=TRUE
NEXT i
PRINT "initialized"
FOR i:=Ø TO sizeof
IF flags(i) THEN
prime:=i+i+3
(* print prime *)
k:=i+prime
WHILE k<=sizeof DO
flags(k):=FALSE
k:=k+prime
ENDWHILE
count:=count+1
ENDIF
NEXT i
NEXT iter
SHELL "date t"
PRINT count; " primes"
END
```

Now we can start looking at some of the parts that will be in most BASIC09 procedures. Like PASCAL, BASIC09 lacks the dynamic memory management in conventional BASICs. Therefore, variables must be dimensioned to inform BASIC09 how to arrange data memory.

```
DIM sizeof:INTEGER
sizeof:=8190
BASE 0
DIM flags(8195):BOOLEAN
DIM i,prime,k,count,iter:
INTEGER
```

There are a variety of variable types in BASIC09, but only Boolean and Integer appear in our example. The variable `flags (8195)` is an array starting with a 0th member (base 0) with 8195 members. A Boolean variable uses only one byte, so the array `flags (8195)` uses 8196 bytes with its 0 member. Integer variables use two bytes each — more bytes are used to dimension them than their data uses.

Variables are not automatically initialized when the program is run. A variable is assigned memory space. That space may contain any sort of garbage. The following code makes 10 passes through the program and initializes the variable count and array flag (8195) at the beginning of each pass:

```
FOR iter:=1 TO 10
count:=0
FOR i:=0 TO sizeof
flags(i):=TRUE
NEXT i
PRINT "initialized"
```

Following the initialization is the program code that does the real work. BASIC09 custom calls for assignments to be made with ":=" rather than just '='. This follows PASCAL practice. For example, the line count:=0. However, if we wanted to know if count were a zero in an IF statement, the':' is not used and will give an error. The right way is IF count=0 THEN.

OS-9 modules can be called from a running BASIC09 program. Shell "date t" is an example. The sieve program prints the date and time when the program starts and, when it has finished, it serves as a timer except you need to subtract the start time from the finish time to get elasped time.

With the program properly entered, type the control character q and press ENTER to leave edit mode. Now BASIC09 checks that variables have all been declared and that all control structure keywords match up properly. If you get error messages, from system mode type e and press ENTER which puts you back in edit with your procedure to make corrections. Many times a bunch of error messages show up. One missing NEXT or ENDIF near the front of the precedure confuses BASIC09 and it produces an error message for each succeeding control structure. When this happens, I list the program from system mode to the printer with the command list myprogram >/p. All error messages produced on leaving edit are printed at the end of the listing.

Let's assume you escaped edit mode without incident. Type save sieve to save the program to your current data directory. Finally type run sieve.

OK, how fast is fast? The C version compiled with the Microware C compiler under OS-9 on my CoCo executed in 24 seconds. Not bad for a machine running at .9 MHz. For comparison, a 22 second time was reported for a C compiled program on an IBM PC at 4.77 MHz clock. Because of lack of integers and memory, the sieve cannot be run under CoCo BASIC. BASICA on an IBM PC was reported with a 1,990 second benchmark running integer variables.

Fanfare please! The BASIC09 sieve took 450 seconds on my CoCo. There was no difference between running source code in the compiler and packed code. More about packing in a later column. I expect doubling the clock rate on the CoCo 3 will halve the run time. Now you know one reason I have not moved to a Tandy 1000 or something similar.

The Third One's the Charm

continued from page 58

described in Figure 2.

In graphics, all windows are scaled to 640 by 192. This allows for programs to be written for one size screen without having to worry about what portion of some other screen the application will run on in the future. To change from window to window, the user presses the CLEAR key to move forward to the next window or SHIFT-CLEAR for the previous windows. The window system acts like a super terminal, so you do not use up program memory space for video display.

OS-9 Level II provides many other valuable system functions. Among these is record locking. This allows more than one program to access the same information file at the same time without conflict. Because of the MMU, Level II does not permit memory fragmentation. A full disk driver is included in the system so larger drives can be added in the future.

Developing software under this new system will be a challenge in many ways. First, it is possible to run one OS-9 Level I program under the window system. What this means is that under the Color Computer Level I system, video memory is mapped into the real address space. This Level I video emulation has some additional functionality. Under this system you can have up to two VDG video screens of 6K each or a 16K, 160-by-192 16-color screen, the capability of changing the color palettes and more.

When running any I/O-oriented task, it is the programmer's responsibility to not waste system time or permit his task to endanger I/O.

Here's an example. You have a program that uses the mouse/joystick pointer device on one window and, on another window, you are playing an arcade game. You switch from a friendly user shell on Window 1 to the arcade game on Window 2. You start moving the joystick around to shoot down the flying saucers. Well, by moving the joystick to shoot at the saucer you pull down the disk utility menu back on our friendly menu. If a programmer is not careful, conceivably, when you push the button to fire at the saucer, the button could be misread by the menu which thinks you have selected to format the disk drive. You finish the game, go to the menu and, because the program did not play by the rules, you have lost all the programs and data on the disk. But take heart; OS-9 provides the information so this need not ever happen.

There are some things that both users and programmers should be aware of. First, if you have more than one task running that does disk file I/O, and one of the programs tells you to swap disks in the disk drive, be careful. By swapping the disk, you may deprive the other program of its data. Here again, the programmer should have taken precautions against this by using good error trapping.

With some good forethought by the companies that produce and sell software, the Color Computer 3 could be a new industry standard.

This new machine will challenge the programming community with new possibilities. It will spark our collective imaginations into producing software unlike any other. It will open new doors, cross new boundaries and set Color Computer owners apart from the crowd. Those who intend to write software for Tandy must use OS-9, but they will find that OS-9 will make their lives a lot simpler.

This new software will allow the Color Computer 3 to grow and mature with new, exciting concepts of what can be done on a home computer. Both Radio Shack and the third-party world can produce new, innovative software, and expand our concepts of how we interact with and use a computer.

GOLDSOFT

Hardware & Software for your TANDY computer.

HARDWARE

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

SOFTWARE

Magazines: Australian Rainbow Magazine — THE magazine for advanced CoCo users! Australian CoCo Magazine — THE magazine for the new user of a Tandy computer. Also suits owners of CoCos, MC 10s, Tandy 1000s, 100s, 200s & 2000s. **Back Issues:** Australian Rainbow Magazine. (Dec '81 to now.) **Please Note:** Some months out of stock. Australian CoCo Magazine. (Aug '84 to now.) **Please Note:** Some months out of stock. CoCoBug Magazine. For CoCo — usually 8 programs in each magazine. (Sep '84 to Oct '85) Australian MiCo Magazine. For Tandy MC 10 computers. (Dec '83 to Jul '84)	Australian Rainbow 1986 1982 — 1985 Australian CoCo 1986 Sept 1984 — 1985 each each	$4.95 $2.50 $3.75 $3.00 $1.00 $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! Back issues of CoCoOz are always available	Each Tape Subscription, 6 months 12 months Each DISK Subscription on disk, 12 months	$9.50 $42.00 $75.00 $10.95 $102.50
Rainbow on Tape, or Disk: Australian. The programs you see listed in Australian Rainbow Magazine are available on tape. A boon if you don't understand the language! American. We also supply the programs found in American Rainbow on tape. Please specify either Australian or American.	Each Tape Subscription, 12 months **NEW** for 1986 ONLY Each DISK Subscription on disk, 12 months	$15.00 $144.00 $15.00 $172.00
MiCoOz: The programs in the MiCo section of Australian CoCo Magazine. (For MC 10 computers only) Back issues of CoCoOz and MiCoOz are always available	Each Tape	$9.50
GOLDDISK 1000 — programs from 'softgold' for your Tandy 1000 on disk, and,		$10.95
Goldlink Goldlink is our very special service on Viatel 642# which you can access with a 1200/75 Baud modem and the appropriate software. Goldlink may be accessed at no charge, but access to our BBS on Goldlink costs 15/30c each time or $36.00 annually. Later we will also provide software for you to download, and members will be able to obtain this at no further charge or at reduced charges.	Subscription 12 months	$36.00
Books: HELP: A ouick reference guide for CoCo users. BYTE: Guide for new CoCo users. MiCo HELP: A quick reference for owners of MC 10 computers.		$9.95 $4.00 $9.95
Say the Wordz: by Oz Wiz & Pixel Software Two curriculum based speller programs for your Tandy Speech/Sound Pack.	Tape 32K ECB	$29.95
Bric a Brac: Blank tapes . . . 12 for $18.00 or $1.70 each. Cassette cases . . . . 12 tor $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 (Disk Directory manager)	ALEX. HARTMANN
BANKSTAT (Statement annal & store)	BARRY HATTAM
INSURE (Analyse home contents)	ROY VANDERSTEEN
SPOOL64K (Printer spooler req 64K)	WARREN WARNE
2BC (Hold 2 sep progs in mem)	WARREN WARNE
DATABASE (THE tape database)	PAUL HUMPHREYS
RESTACC (Tape restaurant accounts)	DUNG LY
PRSPDSHT (Disk print out SPDSHEET)	GRAHAM MORPHETT
PERSMAN (Personal finance management)	PAUL HUMPHREYS
CC5 (Sales Invoicing-tape sys)	GRAHAM MORPHETT
COCOFILE (Tape data base)	BRIAN DOUGAN
DPMS (Disk Program Management Sys)	PAUL HUMPHREYS
40KGREY (40K Basic for grey 64K CoCo)	RAY GAUVREAU
TAXATION (Calc tax payable)	?
SPDSHEET (Disk 22 column spreadsheet)	GRAHAM MORPHETT
ACS3 (Multi disk data base)	GREG WILSON

Best of CoCoOz #5. ADVENTURES.

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

Best of CoCoOz #6. PRESCHOOL.

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

Best of CoCoOz #7. GRAFIX.

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

Best of CoCoOz #8. GAMES.

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

Best of CoCoOz #9. 32K GAMES.

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

Best of CoCoOz #10. EDUCATION2.

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

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

GOLDSOFT

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

ORDER FORM

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

Or charge my credit card monthly
TAPE/DISK ONLY

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

Additional Requirements:

....

....

Sub No: □□□□□□ **or □ New Subscription**

Name:

Address:

P.C.

Phone No.:

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

Goldsoft Gift Certificate

NOW AVAILABLE !

For Christmas, Birthdays etc

Give a subscription to:—

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

— IN FACT anything we sell !! —

SO ORDER NOW

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	ROB 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 MIGHELL	07 269 3846
SCARBOROUGH	PETER MAY	07 203 6723
WOODRIDGE	BOB DEVRIES	07 375 3161
AIRLIE BEACH	GLEN EVANS	079 46 1264
BIGGENDEN	ALAN MENHAM	071 27 1272
BOWEN	TERRY COTTON C/O	077 86 2220
BUNDABERG	RON SIMPKIN	071 71 5301
CAIRNS	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	02 645 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