Registered by Australian Post - Publication No QBG 4009

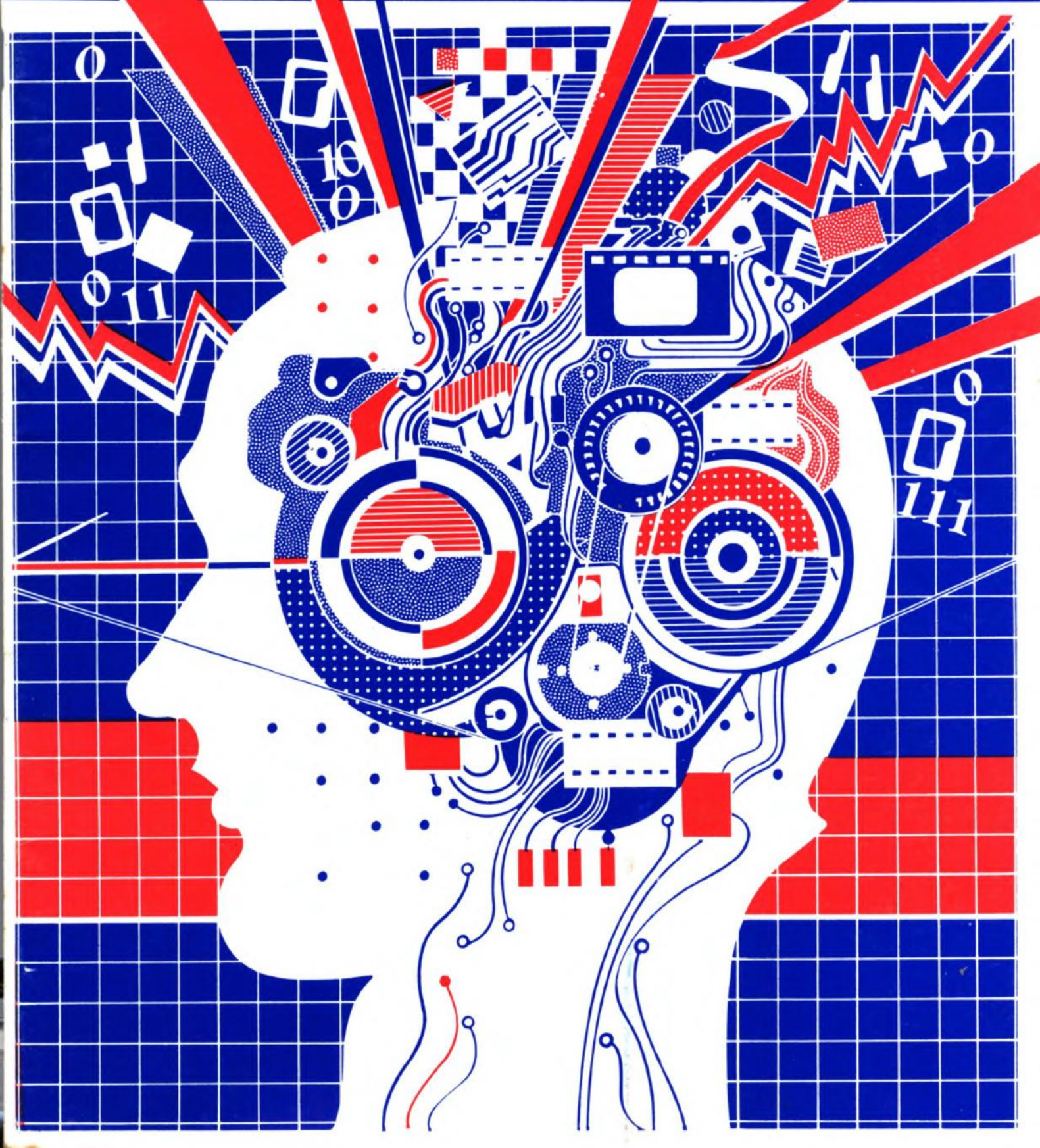
PNG K5.95
NZ $3.95
$4.95

For your TANDY Color Computer

AUSTRALIAN RAINBOW

October, 1986

No.64



ADVENTURE GAMES

What's on
The Best of CoCoOz #5

ADV 32K/16K ..... S.RAYNER
(Restore a nuclear power plant)
ORBQUEST ..... TONY PARFITT
(Save your people from evil!)
LABYRINT ..... JAMES RAYMOND
(Try to get out of this one!)
ADVENTURE + ... SEAN HOWE
(Find wealth!)
QUEST ..... C&K SPRINGETT
(Seek out the 3 crystals.)
PRISON ..... TIM ALTON
(You gotta get out, see..)
OPALTON ..... IAN CLARKE
(A mining adventure.)
WIZSOC ..... DARRELL BERRY
(Get free before time runs out!)
TREASURE ..... C DAVISON ET AL
(Find the Warlock's possessions)

PRICE

Tape	$10.00
Disk	$16.00

NOTE: Most of these games are 32K or 64K.

GOLDSOFT. P.O. BOX 1742,
SOUTHPORT, 4215.

Computer Hut Software

Music Box ..... Tape only $34.95
Learn to Read Music ..... Tape or Disk $33.95

CoCoTex ..... Tape or Disk $79.95
Use your CoCo to contact Goldlink on Viatel.

Special!!
64K upgrades for SOME CoCo's $40.00
Phone or message us on Viatel for details.

T1000 Games:
Sea Search $49.95
Shenanigans $49.95

See us on Goldlink this month!

We accept :-
* BANKCARD
* MASTERCARD
* VISACARD

VIATEL NUMBER
*64262#

Computer Hut Software
21, Williams Street.
Bowen. Qld. 4805.
Phone (077)86-2220

ALL ORDERS SHIPPED SAME DAY

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.

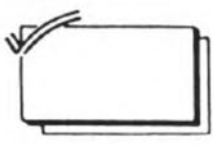
What's A CoCo

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=7985THENB0ELSEPRINT
   "DATA ERROR":STOP
80 EXEC X:END
90 DATA 182, 1, 106, 167, 140, 60, 134
100 DATA 126, 183, 1, 106, 190, 1, 107
110 DATA 175, 140, 50, 48, 140, 4, 191
120 DATA 1, 107, 57, 129, 10, 38, 38
130 DATA 52, 22, 79, 158, 25, 230, 129
140 DATA 39, 12, 171, 128, 171, 128
150 DATA 230, 132, 38, 250, 48, 1, 32
160 DATA 240, 183, 2, 222, 48, 140, 14
170 DATA 159, 166, 166, 132, 28, 254
180 DATA 189, 173, 198, 53, 22, 126, 0
190 DATA 0, 135, 255, 134, 40, 55
200 DATA 51, 52, 41, 0
```

Using Machine Language

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

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

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

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

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

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

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

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

The Crew

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

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

AUSTRALIAN

RAINBOW CONTENTS

PRINT #—2

ON WEDNESDAY (27/7) we were to get the new Color Computer direct from Tandy. It never arrived. I was fairly disappointed. I wanted to toy with it so I could get a 'real' impression of it, unlike the impression everyone else was giving me.

Instead it arrived on the Friday night prior to CoCoConf. I didn't get much done that night, mainly because I was, as Graham puts it, drooling over the new computer.

You all know about the hardware features the new CoCo has; we've told you about them either over the phone, on Viatel, in the magazine and word of mouth. So I won't bore you with that. Instead I'll tell you about the 'enhanced' commands the new CoCo has.

The Basic in the new machine is Microsoft Extended Basic with enhancements by Microware. When you turn it on, you're immediately in the all-RAM mode which means that the entire contents of ROM (including the disk controller) is copied into RAM. (No need to run a 64K program for this one.) Anyway, through the night I discovered some of these neat-o commands in one way or another.

The new hi-res screen and graphics (320 x 192 and 640 x 192) don't use any more memory in the BASIC memory area than they do now. They are located elsewhere in memory. In this way, all software is still 'totally compatable'.

The new commands include HBUFF, HCIRCLE, HCLS, HCOLOR, HDRAW, HPOINT, HGET, HLINE, HPUT, HRESET, HSCREEN and HSET. You can also print on the graphics screen using the following commands: HPRINT, HSTAT and LOCATE. You have three screens to work with: 32, 40 or 80 colunms using the syntax WIDTH 'x' where 'x' is 32, 40 or 80.

There are 64 colours to choose from, but you can only use any 16 colours in one of the screen modes. The commands for these are ATTR, PALETTE and CLS.

New error trapping as well as BREAK trapping commands are ON BRK GOTO, ON ERR GOTO, ERLIN and ERNO.

What we know as the "64K" is located in locations $70000 to &7FFFF. So to access other memory locations, you have to use LPEEK and LPOKE. This means that doing a PRINT PEEK(0) returns the same memory value as does LPEEK($70000). LPEEK and LPOKE allow us to access memory locations outside the normal 64K address space.

For those with the deluxe joysticks (with two buttons) the new CoCo serves a purpose as well. The BUTTON command can be used to read each of four fire buttons; two on the left and two on the right.

While I was there, I tried a few DOSes that were lying around the place. RS-DOS 1.0 & 1.1 work perfectly, even though the power-up message doesn't register 1.0 or 1.1 DOS. The Rainbow Bits DOS 1.3 or 1.4 won't work. Apparently when they designed the 1.3 & 1.4 they used the tokens that the CoCo wasn't using then.

In the OS-9 field it supports functions like windows and a whole lot more. See the article in this magazine on windowing.

But the CoCo 3 has new commands and therefore must have new tokens. So when you load a program with 1.4 DOS using the CoCo 3, you get commands like DOS 1,4 (when it should normally read ATTR 1,4).

All in all, it's a nice machine to work with and well worth the money. Normally machines with such functions cost around about $1500 to $2500. This new CoCo is merely $449!

CoCoConf

CoCoConf was held on the weekend August 30-31st. It was a great success. About 130 people attended and all of us had a great time. Compared to last years Conference, we had twice as many people attending with half as many problems. This was due to the fact that we had a lot more people helping out. Most of them were volunteered, and others volunteered. (There is DEFINETLY a difference in the last phrase.)

The people we would like to thank for helping out are:

* Maurice Phillips: Played security guy Fri & Sat. nights with me setting up the tables & chairs to later get wrapped all night in the graphics abilities of the Atari 520ST.
* Andrew White: Was the games player, waiter for those who stayed up Sat. night & sort-of made a fool of himself.
* Andrew Simpson: stayed back Sat. night to see what's new as well as play games who then afterwards left his jacket behind.
* Rainer Horn: Guy with the camera standing from a distance taking photos of embarressing situations as well as bringing along the model train to display the usefulness of the CoCoConnection.
* Michael Horn: Another one who stayed back part of Fri night to see 'what's new' just to find out that he was getting us dinner at 3 in the morning. Another you can blame for taking photos. Also, he gets a mention for best stunt scene. You see, he nearly totaled his car to get to CoCoConf on Saturday Night. (Talk about getting involved! Gee!)
* Annette Morphett: Sales woman who tried hard to sell the stuff we've been selling you lot for the past year.
* Julie Vidler: The good-looking one that sat behind a table most of Saturday giving out name tags.
* (One who remains anonymous): Helped in bringing computers back from the conference to end up getting a ticket for crossing a red light. (Hiya Rob!)
* Tony Evans: Guy from Computer Hut Software who has the largest range of software for the CoCo. He's another one who got booked, I hear, for doing 145 KPH going home from the conference.
* Lots of other people that can't be named here otherwise we would have a whole magazine of thanks to those who helped.
* Seagulls: They fed us and had to put up with us bunch for the whole weekend along with those people who watched their football team lose. (By the way, the score was 0 for Seagulls, 24 for the opposition, whoever they are.)
* Graham: (I'm going to mention him even though

he doesn't want me to...) He's the one you should blame if there was anything you didn't like. You see, he organized most of it.

My basic role there was to stay up all Friday and Saturday night and watch over the computers with Maurice. Over the 2 - 3 days we got about 5 hours sleep. That's why the few who saw us at CoCoConf, saw us staggering around like Zombies.

Either way, it was a lot of fun!

Programming Competition

There were so many games & utilities that we (or rather I) spent 3 full days going through all of the programs to see how they measured up. For those which did, I had to then pick one of three. Graham then chose the best one.

To see who won and why they won, see the article about the contest in the following pages.

The Future

Seeing that I edit this magazine, I feel that there should be more Australian material put into the Rainbow. I also feel that the Australians can do a lot better than some of the stuff that the Americans are putting out.

So the bottom line is that we are going to put a lot more Australian material into the Rainbow than previously. We find that there is a lot of technical information and material in the office that, really is inappropriate for Australian CoCo.

Australian CoCo is a magazine for the beginner and novice. Australian Rainbow is a magazine for semi-/experienced programmers & hackers seeking advice on what other things they can do with their CoCo.

In the Rainbow, we have material like hardware modifications, other computer languages, advice, utilities, etc.

I would like you computer experts to get that high-level stuff in here! Let's show them Yankees something! Let's see who's more capable!

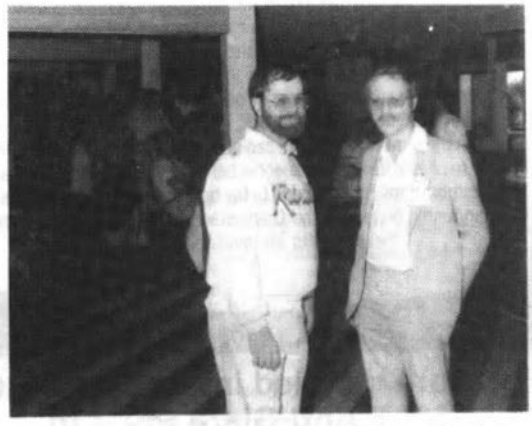
Co CON

Brian Dougan explains the fine parts of MS-DOS.

Wilfred from Tandy examines part of the CoCoConnection system which controls this model railway whilst Richard Pankhurst and Rob Hillard look on.

Graeme Nicholls discusses OS9.

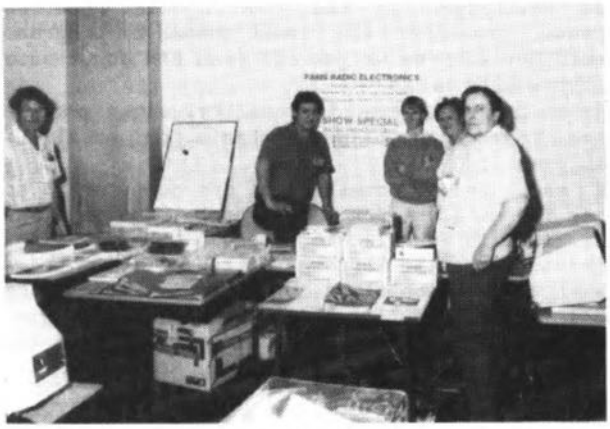
Alex discusses some Advanced BASIC concepts.



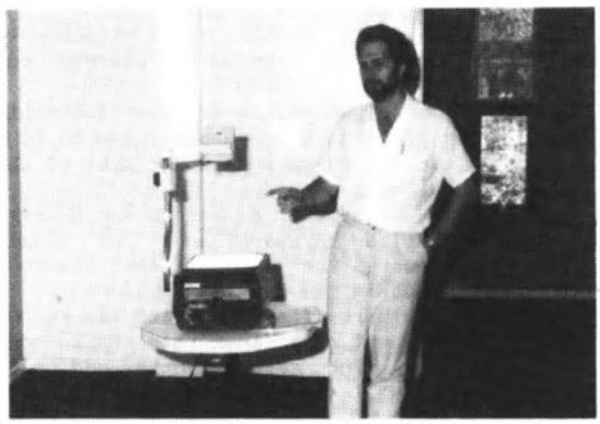
Graham Morphette with Ken Allen (Tandy).

Co F'86

Brian Dougan again(!)



Jackie Cockinos (second left) displays Paris Radio's wares.



Ross Eldridge discusses Education & computers.

Mike Turk discusses BASIC.

Bob Delbourgo shows how a CoCo can be used for research.

CONTEST WINNERS

At CoCoConf, the prizes and awards for the year were presented.

This year a number of the actual recipients were present, so it was nice to be able to present the awards personally.

The Greg Wilson Award for Services to the Tandy Computer Community went to Brain Dougan & Family.

Brain was one of the founding members of the CoCo community in Australia & is probably best known to many CoCo users as the Author of the definitive CoCo utilities Omnikey, & Diskfile.

He is known to locals as a tireless worker, willing to give of his time to the extreme to help someone in need.

Brain has helped people all over Australia with often vexing problems & is rarely known to be beaten in the solving of these problems - most of the time with no thanks or recompense.

Many is the night that he burned the midnight oil doing upgrades & hardware mods for people. He virtually pioneered the use of the double sided double drive set in one box.

He was also part of a team which was responsible for the development of a first rate CoCo modem, & was responsible for several other add on devices which rated as well or better than the imported products.

Brain has devoted himself for over three years to helping other people, and the award recognises this devotion.

However this award also recognises Brain's Family, & the part they have played in support, help with projects, and a continual commitment to playing host / hostess to a never ending stream of visitors.

The Greg Wilson Award was returned by Bob Delbourgo, whose Family won it last year, and, after Bob had spoken on the importance of the award, and reminded us of the very important place Greg Wilson still has in our hearts & in our history, Ken Allen from Tandy then made the presentation.

It came as a complete shock to Brain & family, who provided one of the conference's memorable moments with their look of surprise!

To the Dougans we say "Thank You" for everything.

They receive the Greg Wilson Award trophy which stays with them for this year. They also receive a keepsake wall plaque.

Other awards included a number of Special Mentions. The Special Mentions are not awarded in any particular order or number. However this year there were 4 Special Mentions.

Tom Lehane, like the other three recipients, received his award for his continued support.

In Tom's case, he has supplied programs virtually every month since the MiCo magazine began in 1983.

When he purchased a CoCo sometime later, he did not forget his MC 10 friends, & continued to supply MC 10 programs, plus an increasing number of CoCo programs. His programs are oftem definitive, usually simple, and always easy for the new users to understand.

Tom received a Probe from Goldsoft and a quantity of software from Computer Hut Software.

Max Bettridge, of the four Special Mention recipients, produces the least number of programs per year. But even so we get one from him about once every two months or so.

Max's programs are top quality, often machine language, programs with an emphasis on excellent quality graphics.

He shows us the potential of our computer, & so often is the catalyst for a program by someone else.

Max lives in Bowral, and programs on his own. He has had to learn the hard way, & he is living proof that with a CoCo, isolation is no barrier to gaining a high level of computing skill.

Max received a Probe from Goldsoft, a Koala Pad from Tandy & some software from Computer Hut Software.

Steve Youngberry is also in a small town... much smaller in fact than Bowral.

Steve lives in Tara, Qld, and again until recently has worked alone.

Steve produces consistent quantities of quality medium to hard level Basic programs, usually with a graphics bent.

I can't speak too highly of this man, who like the other recipients is a very humble man who "is just doing what he enjoys".

Steve received a Probe from Goldsoft and some software from Computer Hut Software and Tandy.

Soon after we took over the publication of the magazines in 1984, the first article I read was from a mother of 10 in Forbes.

I read the article again & again for several days. It had a quality about it which has shown up time and again in Johanna Vagg's work... that of real down to earth sincerity and concern for others.

Since that day, she has continued to present on a monthly basis, even in hospital when she should have been giving birth to Katie, articles of enduring value to us all.

Johanna, like Steve & Max, was able to be at CoCoConf, and she received an Ears package from Bayne & Trembath.

It was hardly a secret that Bob Horne would be

continued on Page 16

REVIEWS OF PRODUCTS JUST RELEASED IN THE USA

Software Review — Software Review

Innovative *Plateau of the Past* is a Swashbuckler

After many years and many Adventures, it would take something uncommon to really stir my interest. Not just another dragon to slay, another cave to explore, another 300K monstrosity, but . . . well, something different that could really offer an interesting challenge. Something new.

Enter *Plateau of the Past* — just the thing to whet your whip-cracking appetite.

Almost as soon as you begin *Plateau*, you will be reminded of Indiana Jones. Rather than a classic text or a modern graphics Adventure alone, this Adventure combines a little of both. A dash of animation and a separate fight screen help add to the challenge and the difference of *Plateau*. This is not an easy Adventure to solve! One nice feature of this program is that authors Chuck Jager and Jim O'Keefe have waxed philosophical, and the text and gravestones have interesting sayings. Death, my dear fellow Adventurers, is no longer boring!

The program is composed of three major components plus start-up routines and so forth. They are the text window, generally the lower half of the screen, and the map window, a Hi-Res 2-inch by 2-incl area in the upper center of the screen. The map window reads like a map from above, but displays the moves of the characters in the Adventure, as well as your progress, major events and confronted meanies. The third screen is the fight screen. It pits you against the enemies, displays your and their attributes and describes the fights with blow-by-blow descriptions. Fights are resolved when one side's stamina has dropped to zero.

Several things make the game smoother and more enjoyable. These include the compass that charts your progress; being able to check your ability, stamina and luck; selecting and reviewing your weapons; and even the opportunity to eat. The documentation is well-written and clear and tells you all you need to know. The rest is up to you and fate, because luck plays a large part in *Plateau of the Past*.

Good as it is, I found several drawbacks to *Plateau*. First, there is no game-save feature. It could be argued that one is not necessary, however, a save routine would have been a nice extra. Second, you must restart the entire Adventure each time you die. This is tedious, and a shortcut (save routine) would have come in handy. Third, the fight routines, while being creative and well-represented, are somewhat reminiscent of the sound effects portrayed on-screen in the *Batman* television series. Substituting graphics here instead of words would make worlds of difference. A slightly larger map window would have been better, too.

All of these are minor points, however, when considering this extremely well-written, well-presented, innovative Adventure program. *Plateau of the Past* is a good, rollicking swashbuckler of an Adventure to keep even Indiana Jones proteges happy for a long time!

(Zytek, Ltd., P.O. Box 701, Blue Island, IL 60406, 32K disk only $26.95 plus $2 S/H)

— Jeffrey S. Parker

Get a Kick from *Karate*

Karate is a game for either one or two players requiring a 64K ECB Color Computer and joysticks. The software is available on either disk or tape, and is copy protected.

After loading the program, you are greeted with a color test screen. Here you are given the opportunity to correct the color by resetting the computer. The main title screen follows and has spaces to list the top 10 scores. After selecting either one or two players, you are ready for action.

The object of *Karate* is to knock down your opponent using various karate moves, kicks and punches to score points. Your points are displayed at the top left of the screen. In a two-player game, the first player to win two matches is the winner. The number of matches won for each player is displayed at the bottom of the screen. In the one-player matches the opponent is the computer.

Action is controlled by the joystick(s) and the following karate moves are possible: low kick (trip), forward kick to body, backward kick to body, front punch to head, front punch to body, front kick to head, backward kick to head and drop kick. In addition to these moves, your man can do either forward or backward flips. The moves and punches appear authentic. Obviously the programmer, Dave Dies, knows something about this popular Oriental sport.

The graphics are really outstanding, and unique sounds are used with the score screen. One man has a blue belt and hair, while the other has a red belt and hair.

My only complaint, after learning that the software is copy protected, is that no mention is made in either the documentation or the RAINBOW ads about a warranty. Certainly the author has a right to protect his software, but he should also tell the customer what can be done in the event a problem develops with the program.

If you are into karate, you will get a kick out of this program. But even if you're not, you will find it an interesting game and a pleasant diversion from typical computer fare.

(Diecom Products, 8715 Fifth Line, Milton, Ontario, Canada L9T 2X8, tape or disk $28.95 U.S., $38.95 Can. plus $2 S/H)

— Jerry Semones

One-Liner Contest Winner . . .

According to Francois, rain is rare in Australia so he wrote this program to remind his children of what it looks and sounds like.

The listing:

```
Ø PCLEAR8:FORS=1TO8:PMODEØ,S:PCL
S:FORR=1TO1ØSTEP2:CIRCLE(126,94)
,R*S:NEXTR,S:FORK=1TO1ØØØØ:FORI=
1TO4:POKE14Ø,1ØØ+RND(15Ø):EXEC43
345:PMODE4,I:SCREEN1,1:FORJ=1TOR
ND(5Ø):NEXTJ,I,K
```

Francois Bolle
Australia

Software Review

OS-9 Utilities Give Added Power to OS-9

One of the nicest things about OS-9 is you are not limited to the utilities that come with the package. The Other Guy's Software is offering a package with some interesting and useful utilities you just might want to try.

I received *OS-9 Utilities* on a disk setup in the popular style having the programs located in their own subdirectory called /cmds. The utilities supplied in the package are: *Bmode*, changes parameters on random block file managers (i.e., disk drives); *Calc*, a scientific calculator with memory; *Cat*, lists files to the standard output; *Cptree*, copies all of one directory (including any subdirectories) to another directory; *CP*, copies one file to another file; *Crypt*, encrypts/decrypts files; *FRS*, reads a Radio Shack DOS text file under OS-9; *Grep*, a patterned search routine; *Hcalc*, a hexadecimal calculator; *Help*, provides help on a user-specified topic; *Locate*, locates all occurrences of a specified filename in a specified directory (includes all of its subdirectories); *Lower*, changes all uppercase letters in a specified file to lowercase letters; *Mkdir*, makes directories; *MV*, moves files from one place to another, deleting the old file as it writes the new one; *RM*, deletes a specified file or files; *Split*, splits one file into several files of specified lengths; *TF*, formats text for display or printing; *TRS*, writes an OS-9 file on a Radio Shack DOS disk; *Upper*, changes all occurrences of lowercase letters to uppercase letters in a specified file; *Rsdir*, reads the directory of a Radio Shack DOS disk under OS-9.

In addition to these twenty utilities, the package also contains three devices. They are MEM, a RAM disk; NUL, a null device; QUE, a device for ordering command processing

I am not familiar with OS-9 packaged for applications other than the Color Computer. Some of the utilities included in the package duplicate utilities that come with CoCo OS-9 packages. These are *RM*, which seems to be the same as the OS-9 DEL command, *CAT*, which seems to be the same as the OS-9 LIST command, and MKDIR, which does the same job as the OS-9 *MAKDIR* command.

All of the utilities ran without a hitch, except *Bmode*, which the documentation warns will probably not work with the standard CoCo OS-9 drivers.

It would be hard to pick one utility over the others. I liked several of them. The ones I would use most often would be *Cptree* to copy all the files in a /cmds directory, and *MV*, which allows you to move a file from one place to another, deleting the file from its former position as it copies it to its new position.

The only minus mark for this package is for the documentation. The package comes with almost no documentation and the user is expected to get instructions from the Help files. The package does come with a three-page photocopy of the installation instructions. I solved the problem of documentation by loading *List* into memory and then listing each of the Help files to the printer. This gave me a pretty respectable set of instructions.

An unfortunate fact of CoCo OS-9 life is a shortage of both disk and memory space. For this reason I would suggest that a new purchaser of this package load each of these utilities into memory one at a time and then execute them to evaluate whether they deserve a spot in the /cmds directory or not. I believe you will find a spot for at least one or two.

Of the four qualities I look for in a utility — ease of use, usefulness, relative freedom from bugs and value for the price — this package qualifies on all four. I have already purchased two of these utilities separately for much more than the price of the entire package from The Other Guy's Software.

I recommend *OS-9 Utilities* without reservation to anyone who uses OS-9.

(The Other Guy's Software, P.O. Box H, Logan, UT 84321, $19.95 plus $2.50 S/H)

— Larry Goldwasser

Software Review

Memory Minder: Insurance Against Costly Repairs

By Dale Shell

J & M has supported our CoCo in the past and is continuing with its new and improved disk controller, a nicely packaged disk drive and *Memory Minder*, a disk drive test program. All three products interlink and are very good.

The new disk controller is well made, with J & M's standard metal case and gold contacts on the circuit board. It is compatible with both the original CoCo and the new CoCo 2s. The new features include a parallel port which supports either a Centronics compatible printer or J & M's new hard drive. Another new feature is an external ROM switch, which allows switching between two ROM (DOS) chips. The controller comes with the new version of JDOS and the associated manual. The new JDOS comes on a 27128, 16K EPROM. This is a 28-pin chip. This socket also accepts a 2764, 28-pin EPROM. The other socket inside the controller accepts the standard 24-pin chip. This can be a Radio Shack Disk BASIC ROM, a 68766 or 68764 EPROM.

Therefore, you can have two DOSs installed and can switch between the two. While the new JDOS is much more compatible with existing software, this switch allows the use of a Radio Shack DOS in the second socket, thereby eliminating all compatibility problems. If you want to, you can replace the JDOS chip with a DOS burned into a 27128 or 2764 EPROM. With both sockets available and the ability to use either 2764s or 27128s in one of the sockets, this really leaves open a host of options. The new JDOS includes all the older JDOS commands plus the ability to boot OS-9 from either a floppy disk or a hard drive. It can also boot *Memory Minder*, the disk drive analysis program, but more on that option later.

The parallel port can be used for either a printer or hard drive. At this time, J & M has the five meg and 10 meg drives available, with the 20 meg promised by the time this review is published. The hard drives work with OS-9. The hard drive memory can be broken up into as many as seven partitions, but none can be larger than five meg.

The printer option allows the use of the serial port for other purposes, but it does have a few drawbacks. First, it does not come with a printer cable. Secondly, if you have a printer that requires a pulse width of six ms or larger, you will have to use a pulse extender. This can be purchased from

J & M for $25. You need to check your printer manual and/or call J&M to find out. Some of the printers that need the extender include the Epson RX-80, Riteman Plus, Oxidata Microline 83, and all Olivetti printers.

You can get the new JDOS controller alone or with a drive system. The drive system includes the new JFD-CP disk controller, complete with the new version of JDOS operating system, and a drive with case and power supply. The options include either a single- or double-sided drive. I really like the horizontal mounting that J & M uses, and the case seems to be very sturdy.

With JDOS, using double-sided drives, the smallest program takes up the equivalent of two granules. There is really very little that can be said about disk drives. J & M's have well-built cases and they are reasonably quiet.

The last part of this trio is the *Memory Minder*. With this you are presented with a menu for clamping test, spindle speed test, index hole timing test, alignment sensitivity test, head alignment test, directional seek test, head rotation test, a special quick test and options for an analog alignment aid or change test parameters.

The test provides a quick summary indication of your disk drives for all the tests listed. Once this test is started, it runs to completion. On each of the seven tests you will get either a pass, marginal or fail indication for that test. If you get a marginal or fail, you should test that section more closely, or have it tested by a qualified technician.

A word of warning, however: If you find an error and you are not technically familiar with the hardware, I suggest you get expert advice before adjusting anything, since you may make it worse. Some test conditions are affected by other factors and not just the one that is being tested. As in the clamping test, the quality of diskette clamping is normally a function of how well the diskette is rotating in a circle, but if the drive is out of alignment, the clamping test results may be meaningless. Make sure you read all the notes in the DDA manual. If you do get a fail on the clamping test and the diskette is not the problem, then the problem should be referred to the manufacturer.

The spindle speed test measures the rotational speed of the disk drive. The speed should be between 294-306 RPMs. If you have a disk drive manual, this should be easy to adjust. Just remember, if you have not done this before, be careful. The next test, the index hole timing test, measures the time difference between the leading edge of the index hole and the beginning of the sector ID mark. Ideally, the time will be 200 microseconds, but can be in the range of 100-300 ms. This is a fairly large margin, but if the timing is out, the DDA manual gives the procedure for adjustment.

Another feature of the test allows you to determine the skew of the head. This is to determine if the head is perpendicular to the disk surface. To do this test, just compare the difference of hole timing from the inside and outside tracks. This is not something you can adjust, so let us hope it is OK. If the head skew is out, you will have to return the drive for refurbishment.

The alignment sensitivity and head alignment tests are used during head alignment. The alignment sensitivity test relates to how tolerant the drives will be to a slight head misalignment.

The directional seek test gives a measurement of how precisely the drives can repeatedly position the head over any track on the disk. The test measures the radial alignment when the head is moved in to a test track and then when it is moved out of the test track. The two measurements are compared; any discrepancies are probably due to hysteresis, and if excessive, the drive should be refurbished.

The last test is the head rotation test. This test determines how close to the center line of the tracks the head moves. The optimum alignment will have the head center line parallel to the track tangent line.

The analog alignment aid is not a test, but is very useful if you are testing drives in the traditional manner using an oscilloscope and an analog alignment disk. This section allows you to start and stop the drive motor, and position the drive head. There is very little that is automatic. This is useful to the more experienced.

Overall, I think the DDA is a good product to have around. A technician can use it in his day-to-day work and the casual user can periodically check the different parameters of the drives to see if a technician is needed. Early detection of a problem can reduce or eliminate costly repairs.

The price varies according to what controller you have, but $59 for single-sided and $75 for double-sided drives can be thought of as insurance against repairs in the future.

(J & M Systems Ltd., 15100-A Central SE, Albuquerque, NM 87123, Drive 0 (SS) $279, controller $139, *Memory Minder* (SS) $59, (SS or DS) $75)

Software Review

Casper CoCo Quick Assembler: Ideal for Beginners

Casper CoCo Quick Assembler is a fast combination of monitor, editor, assembler and debugger. To get this extra speed, the editor looks up the symbols and opcodes and saves indexes into tables instead of characters. The program comes with both a 32K version and a 64K version. The 64K version places the machine language portion of the program and your source code file in high RAM leaving most of low RAM for your machine language program. I like this program but there is good news and bad news.

First the good news. This is a delightful program. It is easy to learn and easy to use, ideal for a beginner. The BASIC loader boots the system and sets certain parameters. From this portion of the program you can save and load source code files and machine language files. The program can create its own backup copy onto a blank disk. From this part of the program simply press 'M' to go to the monitor.

The monitor-debugger is also the command level. It is here you perform most operations. You can examine and change memory, assemble source code, print out source code or object code, set up to eight breakpoints, search up or down through source code for a target string, execute machine code and view the registers as you go. You can scroll up or down through the source file line by line. Numbers can be displayed in hexadecimal or decimal.

The screen used by the program is different from the text screen. You can view the usual text screen at any time. Block move, block copy, block delete and block print can be performed on the source file. You can ask for the number of the line you are in, and you can jump to any line in the source file by entering the appropriate line number. When executing machine code, you can walk through the program one line at a time, a number of lines at a time, or execute

until you reach a breakpoint or a certain number of breakpoints.

The editor is where you type in the source code. You can edit, insert and delete lines of code or simply scroll up or down one line at a time. If you enter an improper opcode or an obvious syntax error, the program alerts you with a beep.

The assembler is activated from the monitor. If an error is encountered, the monitor jumps to the problem line for debugging.

Now for the bad news. I had a lot of trouble booting this program. On my old (gray) 'D' board 64K CoCo I get an error message. Typing RUN over again solves the problem for this old CoCo. On the newer (white) 64K CoCo 2 the computer hangs up completely. I managed to get the 32K version to run on the 64K CoCo 2 by resetting the computer and then typing RUN again. I suspect that this is a minor problem with the BASIC loader and may be corrected in future versions.

Another problem in the BASIC loader is the printer Baud rate. The manual suggests that you can set the Baud rate in Line 50 of the loader. After having some trouble with the printer, I discovered that the Baud rate had already been set at 9600, (easy enough to fix once found).

The manual is not clear enough in explaining how to save machine code once it has been assembled. Also unexplained is the Load command. This command lets you load machine language into memory. However, there seems to be no way to access it. The program deals only with source code and machine code that has just been assembled from source code, not machine language files. There is no disassembler in this program. This means that you cannot examine machine language portions of memory (such as the basic ROM) in terms of assembly language. Advanced users may find this a drawback.

Except for the bugs I have mentioned, however, this is a good program and has a reasonable price.

(Earl W. Casper, 6012 S. 14 Place, Phoenix, AZ 85040, $20)

— James Ventling

Software Review

Eagle Lander is True to Life

What is an Adventurer to do? After managing thousands of nuclear power plants, averting numerous air disasters and conquering many kingdoms, what is there left to do? Conquer the moon? That's exactly the task you are given as the owner of Saguaro Software's lunar lander Simulation entitled *Eagle*.

The idea of a computer-simulated lunar landing is certainly not a new one. These simulations first appeared on computer networks that predate the CoCo by many years. The makers of *Eagle*, however, have taken the lunar lander idea quite a bit forward. *Eagle* is controlled with joysticks on the high resolution (PMODE 4) screen. The bottom portion of the screen plots the position of the craft, providing a graphics representation of the craft in flight. Meanwhile, the top portion of the screen provides the pilot with a host of instrumentation which reports such important data as relative position, altitude, horizontal and vertical airspeed, acceleration, percentage of rocket thrust, fuel remaining and time elapsed. All these indicators help the would-be pilot appreciate the difficulties involved in managing such a complex operation as a moonshot.

The complexity increases as the craft approaches one of the two possible landing sites. Both horizontal and vertical airspeed must be brought down to near zero precisely when the craft is located in the proper position. This is an extremely difficult task, which requires many trials to achieve reasonable adeptness. To make life more dangerous, the user can "play damage control" by choosing the advanced difficulty level. It introduces such emergencies as fuel leaks, maneuvering problems and computer screen failures. These malfunctions provide extra challenges and require greater mastery of the craft's abilities to perform a successful landing.

Other operational commands allow the user to adjust the attitude window for fine or coarse control of direction (a very important feature when trying to land), perform a fuel dump and predict future position based on present course and speed values. Using the latter command, the user may jump ahead in time and accept the predicted values and position just as if he had maintained the course for the prescribed time. All these functions add greatly to the playability of the Simulation (not to mention saving the keyboard from frustrated thrashings).

Despite *Eagle*'s complex nature, the user manual provides excellent insight and interpretation of the physics involved in an actual lunar landing and how the user can apply these to his advantage. The manual also accurately describes the command structure and flow of the game. The documentation is complete in every respect, including a guarantee to replace defective disks unloadable by the original owner.

The author of *Eagle*, Art Martin, has done an excellent job in producing a realistic Simulation of a lunar landing. The Simulation is so true to life, in fact, a certain sense of "harsh reality" is incorporated into the program. This is not an easy-to-master Simulation, rather, it's a reflection of a technologically advanced procedure and requires considerable practice to perform a successful landing. Therefore, this program is excellent for hardened game players who are looking for a considerable software challenge which won't sit on the shelf after the first few successful landings.

(Saguaro Software, P.O. Box 1864, Telluride, CO 81435, $24.95 tape, $29.95 disk, requires 32K and two joysticks)

— Eric Oberle

Interactive *Wishbringer* — A Stimulating Challenge

On a scale of one to 10, I'd give *Wishbringer* 9.95+ points. *Wishbringer* is an interactive fantasy — you interact with the program to create a story with many possible solutions. Unlike many Adventure games, which accept only two-word phrases (typically a verb and object), *Wishbringer* encourages you to talk to it in complete — even compound — sentences. Every aspect of this package has been carefully designed and implemented.

To enjoy this fantasy, you need a CoCo 2 with 64K of memory and a disk drive. Optional items include a second disk drive and a printer.

The object of this fantasy is to free the seaside village of Festeron of trolls, vultures and fortress-like towers. At the fantasy's beginning, you are a postal clerk, with the task of delivering a strange-looking letter to the local Magick Shoppe. When you succeed in reaching the shop, you learn that the proprietor's cat has been kidnapped by the Evil One. She asks you to rescue her cat, and in return she will give you a stone with magical powers — Wishbringer. Upon leaving the Magick Shoppe, you discover that the once quiet and peaceful village has become a battleground between the forces of good and evil. Only you, with the magical assistance of Wishbringer, can rid the town of its evil inhabitants.

The documentation supplied with this program is superb. The 25-page, color instruction manual contains the legend of Wishbringer and complete operating instructions. Suggestions are presented for the novice, and even a sample script is included to illustrate how the fantasy develops. Instructions are given to save a position in the fantasy and to restore to that point. Additional instructions allow you to send the actual script you create to the printer. Appendices describe system commands, list some recognized verbs, explain error messages (called Wishbringer Complaints), and present copyright and warranty information. Appendix G gives a brief biography of the fantasy's originator, Brian Moriarty.

But the documentation doesn't stop with the superb manual. Also included is a poster-sized postal map (also in color) of Festeron, showing the roads and identifying many of the buildings (but not the Magick Shoppe). A reference card summarizes much of the information presented in the manual. It also details a diagnostic procedure which will check if the story data are correct (complete and undamaged). (This allows you to determine whether a problem is hardware- or software-related.)

Still another piece of documentation is the sealed letter you are to deliver to the Magick Shoppe. You open this only when instructed by the fantasy to do so.

The packaging is excellent. In addition to the documentation, Infocom has included a warranty registration card (submitting this card gets you a complimentary subscription to the New Zork Times newsletter) and several pieces of advertising literature. One of these announces the availability of hints and maps for purchase.

Although I am past the age that is so fascinated with games, I really enjoyed doing this review. I was constantly challenged and frequently surprised as the fantasy evolved. The documentation left nothing to my imagination (except, of course, the fantasy itself). Response was quick, and often revealed the author's sense of humor.

(Infocom, available in Radio Shack stores nationwide, disk $34.95)

— Jerry Oefelein

Check out *Computer* and *Business Bankbook*

By Neil Parks

Sunrise Software has two new programs available that not only serve as check registers, but also print checks. If you pay the same creditors every month, this could prove to be a real work saver.

Computer Bankbook is designed for home users. *Business Bankbook* — available in two versions — is a similar program designed for business users. Both programs are 100 percent BASIC and come on disks that can be backed up. Both come with a "code plug." This bit of hardware plugs into the right joystick port, and without it the program is unable to run.

Since *Computer Bankbook* is a BASIC program with no machine code, it should be compatible with all operating systems, but it isn't. The disk I tested appeared to run properly under JDOS, but certain data fields, which I wanted to leave empty (as permitted by the program), were filled with random garbage. Attempts to edit the data via the program's edit routine resulted in various crashes and/or loss of entries.

I had no problems with DECB 1.0 or the disk version of ADOS, so I presume that the problem with JDOS is caused by differences in direct-access disk formatting, but I don't know. Copying the program to a JDOS-formatted disk did not help.

The program ends by cold-starting the computer (`POKE 113,0:EXEC 40999`). With the disk version of ADOS, there is garbage on the screen, but pressing the Reset button restores the built-in DOS. *Computer Bankbook* is a natural for ADOS's RSV high-resolution text screen — or would be, but for some superfluous semicolons which should be removed from the menu-printing routines. The Edit screens actually work much better with RSV than they do with the regular 32-by-16 display.

The eight-page manual is well-written and easy to understand, though a couple of points were left out that probably should have been mentioned.

The first step in setting up the check register is to enter a beginning balance. Although there are only seven items on the main menu, the beginning balance is selected by pressing '8'. That was probably done because, after entering the balance, you won't use that routine again for a year.

There are two ways to enter checks. If you are going to have the computer print your checks, you use the Check Printer routine. When each check is printed, it is automatically entered into the register. For creditors you pay on a regular basis, enter their names and addresses into a file. Then every time you want to send a check to one of those creditors, you call up that file. Enter the last check number used, and the check or checks printed are properly numbered.

There is also the option of special checks, for which the

name and address of the payee is not stored in the file.

The second method of entering checks in the register is also the method for entering deposits — the Add Checks/Deposits Manually selection. The manual suggests that you may wish to use the Check Printer routine in preference over the Add Manually option even if you don't plan to have the computer print your checks. Personally, I disagree. I found the Add Manually option much easier to use.

(The manual says that if you wish to use the Check Printer routine without actually printing checks, you must change one line from a GOSUB to a REM. What they forgot to mention is that this change affects only the regular checks, not the special ones.)

Although the manual doesn't make it clear, all checks and deposits must be entered with a two-digit month. For example, January 25 has to be 01/25, not 1/25. If you enter the month with only one digit, the entry is accepted, and the check is printed properly. But the routine that prints out the transactions for a particular month keys on those two digits, so any entry with a one-digit month is omitted from the listing.

The day can go either way — 12/4 and 12/04 are equally acceptable. The year is optional. Although the *Computer Bankbook* manual doesn't expressly say so, it becomes obvious that one year is the maximum length for the file, because the printout for one month doesn't key on the year.

All entries for a given month must be consecutive. Otherwise, the balance column in the printout for that month will look strange. This may cause a slight problem if you write a lot of checks early in the month, before you get your statement and find out how much interest you earned for the preceding month. But you can always date the interest deposit as of the first of the new month to get around that. Again, this is a point that should have been mentioned in the manual but wasn't.

Each check is considered to be outstanding until you indicate, either on entry or editing, that it should be canceled (cleared). When you get your monthly statement from the bank, just print out a list of outstanding checks. A total will be printed with it. Add that total to the present balance and, if it agrees with the bank's balance, you've made balancing your checkbook quick and easy. (Deposits are considered canceled on entry, but you can always edit one to outstanding if it occurred too recently to be on the bank statement.)

Each check and deposit may be assigned an account number. Here's where the program really proves its value. You can print out a list of transactions for each account number, for any one month or for the entire file, and the list will be automatically totaled. For example, let's say all your checks for charitable organizations are assigned to account 18. Tax time comes and you want a total of charitable contributions — it's as easy as printing out account 18 for the year. For this feature alone, *Computer Bankbook* is well worth twenty bucks.

There are a few quibbles, albeit minor ones. Author James Goldsberry did not make allowances for writing a check larger than $999.99. The program can handle it, but the printout looks strange because it exceeds the "print using" format. (Four-figure deposits and balances look OK.)

There is a slight inconsistency in those routines that require a Y/N response. In some of them, the ENTER key is interpreted as Yes, in others as No, and in yet others as no response. Obviously the third situation is the most desirable and should have applied in all cases.

When you enter the beginning balance (via menu option 8), I suggest you enter the amount as zero. Then, as your first transaction, post a deposit in the amount of the balance. Use the edit function to change the name from "deposit" to "beginning balance" or "balance forward" or whatever. That way, if you ever need to recall the file for a prior year, you don't have to make any adjustment to the beginning balance. This method lets you use the same disk for more than one checking account at the same time — your personal account, your spouse's account, a joint account, etc. When you finish working with each account, just copy the CHECKS.INF file to a library disk under the name of your choice, and recall the one wanted by killing CHECKS.INF on the system disk and copying the appropriate file in its place. You can have several years for several accounts stored on one library disk. That makes more sense to me than creating a new system disk for each account file, as the standard procedure would require.

Business Bankbook is essentially the same program, with a few added features. The main difference is that instead of assigning each transaction to one account, you can assign up to four accounts. So if an invoice includes merchandise cost, freight and tax, for example, each of these components can be posted to its own account. Enter the amount for each account, and the total is calculated automatically.

Business Bankbook comes in two different versions. System I (also known as Bank 7.1 or 7.3) uses one disk drive. System II (alias Bank 9.1 or 9.3) assumes two drives — one for the system disk and one for data storage. The beginning balance entry appears on the menu in *Business Bankbook*, but works the same as in *Computer Bankbook*. (With System II, my alternative method of the zero beginning balance becomes even more useful: You don't even have to kill and copy files to go from one to another. Just rename the appropriate file on the data disk to or from CHECKS.INF.)

The 10-page manual for *Business Bankbook* was apparently written some time after the other one, because it does mention that each file should contain one fiscal year, and it does not suggest using the check printer routine to enter checks without printing them.

I did not test *Business Bankbook* under the non-Tandy operating systems, but since the two programs use similar direct-access disk I/O routines, I would expect the results of such a test to be the same as they were for *Computer Bankbook*.

Business Bankbook allows you to enter a table of Active Accounts, consisting of any account numbers you regularly use, and a name for each account. This chart can be printed out at any time, and will appear in numerical order even if you didn't enter them in order. Also, the account names in the table appear on the check vouchers.

Unfortunately, there are two bugs in this routine. In the Edit mode, you are asked if you want to change the account number or account name. But any change in the account name erroneously produces a random change in the number, and any attempt to change the number fails. Also, if you use an account number that is not in the table, the check printing routine puts an inappropriate name in instead of a blank space. There is one line on the System I program that says RUN BANK.10. However, BANK.10 is a file which occurs only in System II. Somebody didn't debug as thoroughly as he should have.

On the plus side, by the time *Business Bankbook* was written, Mr. Goldsberry corrected the inconsistency in his Y/N routines, so that only a 'Y' or an 'N' would be accepted as a response.

Business Bankbook also has a routine called Convert that allows you to convert your *Computer Bankbook* files to the

Business Bankbook format if you want to upgrade. The *Business Bankbook* manual says that a *Computer Bankbook* file of about 300 records takes seven granules of disk space. After conversion to the *Business Bankbook* format, the same data occupies eleven grans.

The manuals for both *Computer Bankbook* and *Business Bankbook* state that the check printing routines are expressly designed for use with Nebs Computer Forms tractor feed checks #9025-1, and that Line Printer VII, DMP-100, and possibly other inexpensive printers can't handle such heavy paper.

With each program comes a checklist to fill out and send in if you have occasion to write to Sunrise Software. "Please outline the problem you encountered in as much detail as possible," says Sunrise. "We will do all we can to help you solve it." I like that.

(Sunrise Software, 8906 NW 26 St., Sunrise, FL 33322, *Computer Bankbook*, 32K, one disk, $19.95; *Business Bankbook*, specify System I for one drive or System II for two drives, $49.95; $2 S/H)

Software Review

Take a Chance — Play *Skance*

Do you like games of chance such as Yahtzee or dice? Do you like to go one-on-one with your computer and sometimes win? If so, you're a likely candidate for *Skance*.

The program is written entirely in BASIC and requires a 16K CoCo with Extended Color BASIC.

When I ran the program, the instructions were simple and straightforward. Besides the instruction screen, there are four other screens you will use: The Player Screen lists all of the players, the Score Screen shows the current scores, the Dice Screen shows the dice being rolled and the Winner Screen gives the winner's name and score.

After the instructions, you are asked how many people will be playing. Up to four people can play or you can play against the computer.

The object of the game is to score as many points as possible. To score, you need to stop rolling the dice before your point number comes up a second time. Your point number is the first number you rolled.

The game consists of seven rounds of play. After the seventh round, the winner's screen comes up and displays the winner's name and score. Although the computer keeps track of all players' scores and who is playing, once a player begins rolling the dice, there is nothing displayed on the screen to tell the players who is rolling. We found this to be a problem.

After you decide how many people are going to play, the screen indicates whose turn it is. There are two options: press the 'A' to roll the dice or press the 'S' to see the current scores. If the 'A' is pressed, a screen comes up showing two dice. After each roll, you are given three choices: roll again, see score, or next player. These are chosen with the 'A', 'S' and 'N' keys. If any other key is pressed, your turn ends and the next player is up.

This game requires no skill to play, but if you like games of chance, this may be for you.

(Bob's Software, P.O. Box 391, Cleveland, OH 44107, send formatted disk and return postage.)

— John H. Appel

Hardware Review

No Frills U-Buff is an Excellent Value

As much as we hate to admit it, the present-day world puts much emphasis on time. Hurry this and hurry that; it is as if we were in a race to the finish! Well, the U-Buff printer buffer from Digital Devices fits in nicely.

I have given this little gem a thorough run for its money. Over the last month and a half, I have sent it files from Delphi, files from my word processor, data from my BASIC programs and graphics. It hasn't let me down yet. It has always printed each character reliably.

For those who don't know, a printer buffer is nothing more than a go-between. It goes between your computer and printer. A printer is limited in speed. The print head can only move so fast across the page. However, a computer can send data much faster. So the buffer allows the computer to finish sending its data more quickly. Therefore, you don't have to wait all day for the printer to finish before going ahead with your work.

As an example, when I told my system to print a 40.5K file, the prompt was back on the screen in 83.5 seconds. I was already involved in my favorite game when my printer finished printing out the 30 double-spaced pages 15 minutes and 53 seconds later.

The U-Buff is available in two configurations: 16K memory and 64K memory. A nice added feature is the average user can upgrade the unit from 16K to 64K by himself. The manual gives information on what chips to buy and how to install them properly. The only drawback is that the one-year warranty only applies to the configuration purchased from Digital Devices. Still, I believe this is very reasonable.

The buffer is designed to accept input from any computer having a Centronics interface. Since my serial/parallel converter terminates in a Centronics plug, I had no problems. The unit also terminates in a Centronics plug for the printer's parallel port.

The U-Buff comes with a 14-page, easy to read pamphlet. The instructions cover installation, testing, troubleshooting, upgrades and a whole slew of technical information plus a glossary.

Despite its usefulness, keep in mind a printer buffer isn't for everybody. But anyone with a business or who does a lot of printing will surely welcome anything to help them out.

The U-Buff is an excellent, no-frills printer buffer. For the reasonable price it is a good work horse with a good deal of backing from the manufacturer. This one gets four stars.

(Digital Devices Corporation, 430 Tenth St., Suite N205, Atlanta, GA 30318, 16K $119.95, 64K $149.95)

— Cray Augsburg

Wico Command Control Trackball Takes the Punishment

There must be an easier way to get a better score without causing serious harm to your trusty CoCo. We all tend to get carried away during the heat of battle, and this excitement is often transferred into "white knuckle" treatment of some relatively fragile plastic parts.

It's only human nature to get excited during competition. Have you ever noticed that many commercial arcade games use a trackball control instead of a joystick? The action at these arcades often got so intense, the manufacturers had to find a more reliable alternative. Well, that alternative is available for the CoCo in the form of the Wico Command Control Trackball.

This device replaces your joystick and is very rugged. It measures 5 by 6 by 1⅞ inches and weighs a little over one pound. It uses, of all things, a billiard ball as the primary user interface. A firebutton is located at the upper left corner on top so it's easy to get to with your left thumb. If you want to put it on your computer table, it won't slide around easily because it has flat rubber feet on the bottom. It does require a separate power source, which is included in the form of a plug-in wall transformer. The other cable terminates in a standard five-pin DIN plug to fit the CoCo joystick connector.

Operation is simple. You get full 360 degree movement by just moving the ball with fingers or palm. It's easy to use and allows controlled movement by simply moving the ball fast or slow. It's a unique feeling and one you have to experience to fully appreciate. I tried it on several games and was able to get higher scores than I could with my joystick. I also used it with *CoCo Max* and liked the smooth feel and easy positioning of the cursor. The only drawback I noticed on *CoCo Max* was it took longer to go from the workspace to the menu due to the vernier action of the device.

The device boasts the use of a microprocessor control circuit using photosensors and a built-in analog-to-digital convertor with an eight-bit output. Since the CoCo only has a six-bit joystick port, that probably explains some slight positioning errors with programs like *CoCo Max*. Wico is aware of this problem and even discusses it in the instruction booklet.

This device is best suited for CoCo games, although it will most certainly provide excellent service in other programs normally requiring a joystick.

Wico guarantees the Command Control for one year which attests to its durability. This device can take all the punishment you can give it — so bring on the Klingons!

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

— Jerry Semones

CONTEST WINNERS continued from Page 8

receiving the Education Progamming Award.

Bob has produced a most consistent line of Educationally sound programs for use on CoCos this year.

In fact most of his programs have broken new ground in the use of the CoCo in schools.

Tandy gave Bob a 16K ECB computer, for which Computer Hut Software provided a 64K upgrade, plus a number of Education programs for use at St Edmund's Catholic School Ipswich, Bob's place of employ.

The OS 9 Programming Award went to Ian Lobely of Tandy in Wagga.

His program will be published in Australian Rainbow next month. It demonstrates the very successful use which OS 9 can be put to in business, and opens the way for some very exciting stuff on the CoCo3 with the Level 2 OS 9.

Ian receives a 68000 Class Computer (CoCo add on) from Blaxland Computer Services for his trouble.

George McLintock is the recipient of a Goldsoft CoCoConnection for his two Utilities competition entries, "Compress" & "Remove".

These programs are just two of a vast number we received this year for this competition. In fact, due to its popularity, this competition will be expanded in the coming year.

Finally to the games.

The games we received this year were the best we have ever seen. Not only that, but there were a lot of games at this same excellent level.

In the Adventure Section, Andrew White was the winner with his "Goldsoft Adventure", which, by the way, almost became a reality for someone!

His game shows mastery of the disk system & of graphics handling techniques and what's more, is fun to play!

Andrew receives a Video Amp from Geoff Fiala, and a Koala Pad from Tandy for this fine game.

Andrew McLintock wrote a Simulation called U Boat & this fought off some very stiff competition to win Andrew the CC Bus from Paris Radio for the best Simulation submitted this year.

Finally, Craig Stewart submitted two excellent Arcade games for CoCo, "Tank" & "Shoot Out". Both these games show a working knowledge of the graphics screen in Machine Language.

In particular, "Shoot Out" is a fun game to play with the added advantage that it is played against a human adversary.

Craig has won a Tandy Disk Drive and we congratulate him on an excellent job.

New Competitions.

The competitions for the new year now get under way. The first competition for the year, with a closing date of 7th February is the Music Competition.

This competition, if sufficient entries ensue, will be broken into a number of categories dependant on the basis of the program, ie whether the program uses the PLAY command, the SOUND command, Machine language, or one of the many add on units available today.

The competition is open to owners of all computers, so we are looking forward to seeing what the owners of the IBM PC's especially, can turn out! Will they make better music than CoCo owners? Can our Commodore readers do any better?

Well we'll see as the competition unfolds.

Samples of 320 by 192 mode from demo program for new CoCo 3. Sixteen colors available. Highest resolution available: 640 by 192.

Tandy Color Computer 3 Does Windows, and a Whole Lot More

By Greg L. Zumwalt

I recently received my sixth Color Computer. As an applications programmer, I tend to wear them out long before their time. I use Color Computers in developing software and hardware for the CoCo as well as other equipment, and give the poor machines quite a beating in the process.

I removed Color Computer five from active duty and plugged number six into the expansion interface. After installing the OS-9 boot disk in Drive 0, I typed DOS, and a message appeared on the screen:

```
Welcome to OS-9 Level II for the
Color Computer 3
```

Level II? Color Computer 3?

Hardware Preview

The Tandy Color Computer 3 is a true next generation CoCo. The heart of the machine is a two-megahertz 6809 teamed with the Tandy designed GIME (Graphics Interrupt Memory Enhancement) chip. The GIME contains the Color Computer 3 enhancements, as well as the standard Color Computer VDG logic. Tandy retained the older Color Computer logic to remain compatible with most of the existing CoCo software. The enhancements, however, are impressive.

How about 512K RAM capacity with an MMU (Memory Management Unit) to assist in managing it? Or 320 by 192 16-color graphics? Or 640 by 192 four-color graphics? What about composite, RGB or standard RF video output? I like an 80-column display. The Color Computer 3 contains all of these features, plus more.

Software

Tandy provides various software enhancements that utilize the Color Computer 3 hardware. BASIC is compatible with previous 64K Extended BASIC, but allows setting screen widths (32, 40 or 80 columns), screen colors (foreground, background *and* the border), graphics modes (now up to 640 by 192, four colors), and more.

However, the real power of the new Color Computer becomes apparent with Microware's OS-9 Level II Operating System. If you are familiar with OS-9, I would suggest that now there is a real reason to develop an interest in it. For those of you familiar with Level I OS-9, you will really enjoy Level II. Level II is designed to exploit the expanded RAM capacities of the Color Computer 3, like Level II on other systems. But there is a difference.

Level II OS-9 for CoCo 3 supports windowing, a feature that, after having used it, I don't ever want to give up.

Windows

A window is an area appearing on the screen that acts just like a regular display, the main difference being you can assign more than one window to the same screen. For example, as an applications programmer, I am constantly switching between a text editor, an assembler, the application and a debugger. Anyone who has written assembly language programs knows that this cycle often repeats, sometimes many times, resulting in lots of typing and typing errors.

With windowing, I simply place a text editor window in the upper left-hand corner of the screen, an assembler window in the upper right-hand corner of the screen, a debug window in the lower left-hand corner of the screen, and finally, an application window in the lower right-hand corner of the screen. Each of these windows now acts as though I have four separate Color Computers. While the application is running, I can edit, assemble and debug all at the same time. Output from any of the programs appears instantly on the associated window. To direct input to a particular program, all I have to do is press the CLEAR key on the keyboard until the cursor appears on that window, then type. OS-9 allows me to define the size and location of each window. I can even define a window as the entire screen, in which case OS-9 switches screens when I want to work on another.

How does this work? OS-9 is what is known as a multi-tasking multi-user operating system. This means OS-9 is designed to allow more than one program (called a task) and more than one user at a time to share the same computing resources. Level II OS-9 for the CoCo 3 takes this concept one step further by providing more than one screen (the window) to accomplish this. Let's go through the process of creating a window using the OS-9 Build utility.

Building Your First Window

The Build utility allows us to create an OS-9 procedure file. A procedure file is simply a disk file containing a series of OS-9 commands that OS-9 executes in order. To create the procedure file, at the OS-9 prompt, type the following:

```
build window
```

The Build utility responds with a question mark followed by a space. Now type the following:

```
iniz w1
merge sys/stdfonts >/w1
display 1b 20 07 00 00 20 0b 00
  07 07>/w1
shell i=/w1&
```

Press ENTER twice after this last line.

After the last ENTER, the Build utility returns to OS-9. There is now a procedure file on the disk called Window1. To run the procedure, at the OS-9: prompt type:

```
window1
```

OS-9 responds with the '&' symbol followed by a number. This verifies that the Window1 procedure accomplished its task. To see what happened, press the CLEAR key on the keyboard. A new screen appears with a window in the upper left-hand corner containing the OS-9 prompt. This window, with its shell, allows you to execute OS-9 commands just like the original screen. In effect, you have two computers. Press the CLEAR key again, and the original screen reappears. Let's examine the procedure file we created and see how it works.

Under OS-9 Level II, a window is treated as simply another OS-9 device. The device name for a window contains the window descriptor (the 'w') and the window number (in this case 1). Therefore, to talk to window one the descriptor w1 is used.

The first command, Iniz, initializes w1. This is similar to an Iniz of any other OS-9 device.

The second command, Merge, tells OS-9 which character set is to be used with w1 (a variety of character fonts are now possible with the Color Computer 3).

The third command, Display, sends the command sequence to OS-9 that creates the window. The command sequence contains 10 bytes describing to OS-9 the command, type, location, size and color of the window to create. The first two bytes following Display are the create a window command. The third byte defines the window type; in our example, a 640 by 192 four-color graphics window. The fourth and fifth bytes decribe the 'X' and 'Y' character location of the upper left-hand corner of the window. The sixth and seventh bytes define the 'X' and 'Y' character size of the window. Finally, the eighth, ninth and 10th bytes describe the foreground, background and border colors for the window.

The fourth command, Shell, creates a shell for w1. The shell is what allows you to execute any of the OS-9 commands or applications.

Building Your Second Window: A BASIC Approach

To further illustrate, let's create a second window. However, this time let's create it using BASIC09 (which, by the way, is included with the OS-9 Level II operating system), and place it on the same screen that w1 occupies, but in the upper right-hand corner. Press the CLEAR key to display w1, then at the OS-9: prompt, type:

```
Basic09
```

The BASIC09 prompt appears, followed by B:. From here, type in the following:

```
e window2
  shell "iniz w2"
  dim command,path:byte
  dim count:integer
  data 27,32,0,40,0,32,11,7,0,7
  open #path,"/w2":write
  for count=1 to 10
  read command
  put #path,command
  next count
  shell "shell i=w2&"
  end
g
run
```

A second window appears just to the right of the first, whose foreground background colors are opposite those of w1. Now press the CLEAR key. Notice the cursor on w1 disappears, while a cursor appears on w2. Each time the CLEAR key is pressed, the next window is selected for keyboard input in round robin fashion.

The BASIC approach appears somewhat different from the procedure file approach; however, they both accomplish the task in a similar manner. The first line of the BASIC approach calls the BASIC09 editor, allowing you to type in the program. The second line tells BASIC09 to send the Iniz w2 command to OS-9 the same as Iniz w1 of the procedure file approach. The third and fourth lines define the variable types to be used in the program (a requirement of BASIC09). The fifth line is a DATA statement containing the command sequence that creates the window. The sixth line opens a path allowing BASIC to communicate with window two. The seventh, eighth, ninth and 10th lines are a FOR/NEXT loop, reading data from the DATA statement and sending it through the path to window two. The 11th line creates a shell on window two. The 12th line ends the program. The 13th and 14th lines exit the BASIC09 editor and run the program.

Putting Windows to Work

It is important to realize that even though a window is not selected for keyboard input, the program on it is still running. To illustrate, let's put the two windows we have created to work. We will make window one (w1) a clock that displays date and time continuously. Then, on window two (w2) we'll illus-

trate another windowing concept.

Press the CLEAR key on the keyboard until the cursor appears on w1. Then from the BASIC09 B: prompt, type:

```
kill window2
e clock
  dim time:string
  print chr$(12)
  10 if date$=time then goto 10
  else
  time=date$
  print chr$(1)
  print time
  endif
  goto 10
q
run
```

A BASIC09 program called *Clock* is now running on w1, displaying the time. Now press the CLEAR key. Notice that the cursor moved to w2, but the clock program on w1 is still running. This is an example of how multi-tasking multi-user and, of course, windowing really shine. W2 is now available to write another program, run a game, or whatever, while w1 continues to run the clock! Another plus — from the OS-9: prompt on w2, type:

```
Basic09
```

Now you have both w1 and w2 available for BASIC programs, but the real plus is both w1 and w2 share the same copy of BASIC09. No need for a second copy. The design of OS-9 allows programs to be re-entrant, essentially meaning that the same program can be used by more than one person at the same time, providing big savings in memory.

The Overlay Window

The windowing system provides a second type of window, the overlay window, which does what the name suggests. An overlay window overlays an existing device window (windows w1 and w2 that we previously created are device windows). Overlay windows are useful when an application needs to prompt the user for input, display menu selections, announce messages and so forth, because they can be designed to really attract the user's attention.

Creating an overlay window is relatively simple. So while our clock is running on w1, we will use BASIC09 on w2 to create our first overlay window.

From the BASIC09 B: prompt on w2, type:

```
e overlay
  dim count:integer
  dim command,path:byte
  data 27,34,1,2,2,12,4,0,2
  open #path,"/w2":update
  for count=1 to 9
  read command
  put #path,command
  next count
  print " this is an"
  print " overlay"
  print " window"
  10 goto 10
  end
q
run
```

The program creates an overlay window on window two with the message this is an overlay window. The program is similar to the window two program we wrote earlier. However, the data in the DATA sequence contains nine bytes. The first two bytes are the create an overlay window command. The third byte tells OS-9 to save the area under the overlay window, and restore it after we are finished with it (we will see this in a moment). The fourth and fifth bytes are the 'X' and 'Y' location of the upper left-hand corner of the overlay window in relation to the device window (w2). The sixth and seventh bytes are the 'X' and 'Y' size of the overlay window. The eighth and ninth bytes are the foreground and background colors respectively.

Notice in the BASIC program, line 10 calls itself. This was to stop the program before the END statement was encountered. Now press the BREAK key. The program is stopped, but all output to window two is still appearing within the overlay window. For example, let's list the overlay window program using the BASIC09 list function. From the B: prompt type:

```
list overlay
```

The overlay window program is listed in the overlay window. When you create an overlay window, all subsequent output to the device window containing the overlay window will appear in the overlay window. The overlay window must be turned off with the turn off overlay window command. To send the turn off command from BASIC09 at the B: prompt, type:

```
$display 1b 23
```

Notice that OS-9 restored the original image that was under the overlay window. This is because in the create an overlay window command we told OS-9 to save the area under the overlay window by setting the third byte of the command sequence to a value of one. If you set this value to zero, OS-9 will not save the area under the overlay window, and thus will not restore the area when the overlay window is turned off.

Implications

Imagine using your Color Computer 3 as a home security system, and at the same time typing a letter to a friend or business associate. Meanwhile a large C language program is compiling, and a telecom program is downloading the latest stock quotes from an online information service, all at the same time, all on the same screen, and all without ever having to disable the home security system.

Sound impossible? The Tandy Color Computer 3 teamed with Microware's Level II OS-9 and the Windowing System is designed for easy implementation of just such an environment. As we have illustrated, it is not just a fancy display package. It is a powerful tool for both user and programmer, providing the real benefits of a multi-tasking multi-user operating system in a manner that doesn't require an assembly language applications programmer to implement.

COMMENTARY

THE COLOR COMPUTER 3: A PROGRAMMER'S DREAM

by Steve Bjork

While I was writing this article about the new Color Computer 3, I realized that the CoCo has been the same basic computer for over six years! In this day of ever-changing computers, that's a long time. True, Tandy repackaged our lovable little CoCo (called the Color Computer 2), but with no real hardware or software changes.

Programmers have learned how to push the Color Computer to its outer limits. But with the old CoCo hardware, we just can't break any new ground. That's why Tandy is introducing the Color Computer 3.

The new Color Computer 3 is on the cutting edge of technology. The power of Color Computer 3 is gained by using a new custom chip called the GIME (Graphics Interrupt Memory Enhancement) in place of the older SAM and VDG chips. This new chip allows up to 512K of memory controlled by a Memory Management Unit (MMU for short), and a 640 by 225 super Hi-Res graphics screen at twice the old Color Computer speed. Using the analog RGB monitor, 225 scan lines are the maximum supported; however, new software must be written to take advantage of it. Radio Shack only supports 192 scan lines for use with TV viewing devices. Not bad for one chip!

But how can these hardware changes help the programmer? Take a look at graphics game called *Marble Maze*. Before, it was just a little too slow. But with twice the speed on the new CoCo, it is at the right pace and I am having a grand old time playing it! Soon we will see programs on this new computer that would have been too slow before.

The new text display modes are going to be a real help to programmers. Before, the text screen had only 16 lines of 32 characters each and no true lowercase. On the CoCo 3 the text display mode has 24 lines and up to 80 characters per line. Not only has lowercase been added, but also underlining and blinking attributes for each character on the screen. Because this is a *color* computer, the attributes also include eight foreground colors and eight background colors for each character.

With these additions, it is very easy to write a word processor that not only shows more of the text, but also shows any spelling errors by highlighting them in red lettering.

For a graphics programmer, the new graphic modes of the Color Computer 3 are a dream come true. Not only do we have about four times the dot resolution, but up to 16 colors on the screen at one time. On the old Color Computer there was a maximum of four colors with the colors never changing in that mode. The Color Computer 3 has up to 16 colors per screen with 64 colors to choose from. When a dot is placed on the screen only the color register (or palette) is selected. The color register holds the true color to be displayed. If a programmer wants to change all the red on the screen to yellow, all he needs to do is change that palette from the red to the yellow color code.

Let's use a picture of the New York sky line at day for example. By changing a few color registers this daytime scene slowly changes to one of dusk, then to a nightime scene with twinkling stars. All of this done by a BASIC program using the palette command!

One problem with a higher resolution display mode is it can take up to 32K of memory to display one screen. In just using two graphics screens all 64K of memory the 6809 can address would be used up. This is where the Memory Management Unit comes into play. Only one graphics screen needs to be addressed by the 6809 at one time, so the MMU is programmed to select one screen and place it in the 6809 address range.

The new BASIC ROM in the Color Computer 3 takes this one step further. The new higher resolution display modes do not use any of the BASIC program space as did the older Color Computer's graphics modes. This gives the BASIC programmer about 27K of memory (after doing a PCLEAR 1) for programs and variables, while still using a maximum size graphics screen of 32K!

A Programmer's Dream: OS-9 on the Color Computer 3

Tandy did not stop at hardware when they were improving the Color Computer; it takes software to run any computer. A more powerful and upward compatible version of OS-9 called Level II is the real key to the power of the Color Computer 3. This OS-9 Level II is able to give each program up to 63.5K of memory to work in. That is a lot workspace on a 512K system! On the old Color Computer there was only 35K to 40K to work with.

With OS-9 being a multi-tasking/multi-user operating system it would be kind of nice to use one screen for compiling a program while using another for editing a different program. By adding the Multi-View windowing system, it is as easy as pressing the CLEAR key to move from window to window. In one window you could be editing a program, then in the next window be running a test version of the program, and have all debug data in yet another window.

What's an Analog RGB Monitor Anyway?

By Ed Ellers

The Color Computer has had an RF output ever since its inception way back in 1980. This means the CoCo can be used with a standard TV set instead of a special monitor, and also that the display quality is affected by the limitations of TV (which were never intended for computer displays anyway). Many CoCo buyers have added video output jacks to their computers in order to use monitors.

Now, we have the Color Computer 3 and it has the same TV jack on the back panel along with the Channel 3|4 switch, so it still works with home TV sets. Tandy has also added video and audio output jacks to allow a factory-approved connection to a monitor (or to a VCR). It sounds like the answer to a bleary-eyed hacker's dream — but wait! On the bottom there's a strange seven-pin socket. What for? It turns out that Tandy has introduced a new 13-inch analog RGB monitor — the CM-8 — that displays the CoCo 3's text and graphics much more clearly than a composite monitor can.

RF Output

The RF output is the TO TV output that feeds into your TV set on Channel 3 or 4; the signal is much the same as what you would receive from a broadcast station or cable. The problem with this isn't so much that the RF modulator inside the computer is distorting the signal, but that the tuner and especially the IF amplifier in most TV sets do terrible things to the incoming signal in the process of changing it back into raw video. If you were to use an oscilloscope to look at the signal going into the video amplifier stages of your TV set and compare this with what goes into the modulator of your CoCo, you'd be surprised that the picture you see looks as good as it does.

Many newer TV sets have improved circuitry that gives a flatter frequency response and causes less signal distortion, but using an RF hookup is still a roundabout way to do it and far from ideal. That's why the more expensive color TV sets nowadays have video input jacks for better pictures from VCRs, videodisc players and satellite receivers; more to the point, it's also why many CoCo users have ignored the RF output in search of something better.

Composite Video

Composite video is the output from a TV camera or from the VIDEO OUT jack on a VCR; it's also what you get from a monitor adapter board for a CoCo, and from the video output on the CoCo 3. Composite means the signal includes horizontal and vertical synchronizing pulses and blanking pulses as well as video; it's the same as the signal that goes into the RF modulator in the CoCo (or into the TV station's transmitter). In black and white, a composite signal carries a lot more detail than an RF signal does, and with a good monochrome monitor (which can be black and white, but is more often green or amber) you get very clear text and graphics.

However, a composite color signal has many of the same problems that the RF method had. To squeeze three colors, which logically would take up as much space as three black and white signals, into the space of one TV channel and still have a useful black and white picture at the same time, color TV systems create a black and white signal from the three colors and then create a subcarrier to carry the added color information. The subcarrier can't handle as much detail as is present in the monochrome "luminance" signal; this works out reasonably well for regular TV viewing, but definitely doesn't when you're looking at Hi-Res graphics at close range.

There are also various imperfections in the process of separating the subcarrier from the luminance signal (we'll get to one of them later). With the Color Computer 3 there's a better way.

Analog RGB

RGB simply stands for red, green and blue, the three primary colors of color TV and computer displays. This method involves sending each color signal to the monitor over its own wire in the monitor cable. The signals are never mixed together, so there's no problem in separating them, and since the signals are only traveling a few feet there's no problem with band width.

This is the ideal way to display color graphics, but you do need the red, green and blue signals coming from the computer and a monitor that can handle them. The output is taken care of by the new GIME chip in the CoCo 3; the monitor is Tandy's new CM-8, which, at $299, is quite a bit less expensive than the analog RGB monitors sold for use with Commodore's Amiga and the Atari ST series computers.

"Analog" means the voltage fed to the monitor varies with the signal level for each color. In a digital RGB monitor, such as the ones used with the Tandy 1000 and the IBM PC, the red, green and blue signals simply go on and off. A digital RGB monitor generally only shows sixteen colors with four signal lines; the colors available with analog RGB monitors are limited only by the computer.

The CoCo 3 and the CM-8 monitor provide the best picture quality the CoCo has ever had — and *this benefit is realized even with existing CoCo programs* running on the CoCo 3. Comparing the CM-8 with what you get on a normal TV is like comparing night to day. However, there are some minor idiosyncracies in the CoCo 3's RGB output. One is the lack of artifact colors in PMODE 4 graphics intended to run on previous CoCos. These red and blue colors were caused by the dots in the display being misinterpreted by the TV or monitor's color circuits; this confusion just doesn't happen with RGB displays, so the dot patterns are now seen as dots instead of colors.

Also, for some reason the colors that the CoCo 3 provides are seen differently on the RGB monitor than they are on TV sets and composite monitors. Because of this, Extended BASIC includes the PALETTE RGB and PALETTE CMP (composite) commands that can be used to switch from one mode to the other. (The pictures of CoCo 3 graphics shown in this issue were taken from the CM-8 monitor, but the program was configured for composite monitors.)

The CM-8 monitor shows all the high-quality text and graphics the Color Computer 3 is capable of providing. However, those who want to use color composite monitors or TV sets with the new computer can do so, just as they did with previous CoCos. The CM-8 monitor can be added at any time.

Each window can be in text or graphics modes and can take up all or only part of each screen. You can even put several windows on the same screen as long as they don't overlap.

If that is not enough to get a programmer drooling, there is a full point-and-click interface with drop-down menus built into the Multi-View package. A full set of graphics drawing commands like LINE, BOX, FILL, CIRCLE, ELIPSE and GET/PUT buffers are included to round out the package.

The Multi-View package holds most of what a programmer needs to do programs like the *CoCo Max*. As a matter of fact, this package already has about 80 percent of the code that Tim used in that landmark program for the older Color Computers. It will not be long before we see many top of the line drawing programs under OS-9.

This is one programmer's view of the Color Computer 3. It has a lot of what we've been asking Tandy for in the way of power. So, I better get back to work; I have a few programs to develop that will knock your socks off!

Two-Liner Contest Winner . . .

And they're off! Pick a horse to win out of a field of seven. If you press ENTER when they are at the line, the horses will run. You and your friends can then pick the horse you want.

The listing:

```
Ø POKE65495,Ø:DIMH(2Ø,2Ø):PMODE3
,1:PCLS:DRAW"BM1ØØ,99U8R1ØU6R8D4
L2D1ØL4U3L8D3L4":GET(9Ø,83)-(121
,1Ø1),H,G:PCLS:SCREEN1,Ø:FORX=1T
O191STEP27:PUT(Ø,X)-(31,X+18),H,
PSET:U=U+1:P(U)=X:NEXTX:DRAW"C3"
:LINE(24Ø,Ø)-(255,191),PSET,BF:E
XEC44539:PLAY"O3L18CL4F"
1 FORX=1TO7:M(X)=M(X)+RND(1Ø):PU
T(M(X),P(X))-(M(X)+31,P(X)+18),H
,PSET:IFPPOINT(M(X)+33,P(X)+9)<>
1THENPLAY"O3L115CDDEFFGAAB":FORT
=1TO1Ø:PUT(M(X),P(X))-(M(X)+31,P
(X)+18),H,PRESET:PUT(M(X),P(X))-
(M(X)+31,P(X)+18),H,PSET:NEXTT:E
XEC44539:RUNELSENEXTX:GOTO1
```

Mike Cooney
Mansfield, OH

Rational Thinking

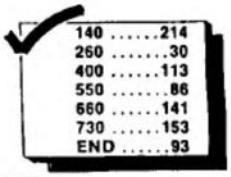
Ratios is a hands-on exercise in converting fractions to ratios and percentages; the sort of exercise which enables a pupil to increase familiarity and skill by means of unlimited repetition.

If you enter the numerator and the denominator of a fraction, you have control over the level of difficulty. If you let CoCo enter random numerators and denominators, the level of difficulty is controlled by the magnitude of the numerator you choose (Line 370). In either case, the numerator is restricted to two digits and the denominator to three, which keeps the program within the ability limits of a disadvantaged pupil. Increase the level of difficulty by changing the values of LEN(A$) and LEN(B$) in lines 130 and 140 and VAL(CR$) in Line 370. After five attempts, you are given the choice of entering your numbers or random numbers again.

In entering answers to questions, you are restricted to seven characters including a decimal. Illegal entries are erased and a new entry can be made. The instructions ask you not to type the percent (%) sign because this is added to your answer.

After an answer has been entered, the screen shows either RIGHT or WRONG and the correct answer in both cases. The entries in lines 50 and 60 ensure that CoCo's correct answer (P$) is the same length as your correct answer (E$). If you enter 33.3 as the percentage equivalent of 1/3, then you get ANSWER WAS 33.3%. If you enter 33.3333, then CoCo's correct answer will be 33.3333%. If your answer is wrong, then CoCo's correct answer may vary in length. This is followed by an X RIGHT OUT OF Y message and a score expressed as a ratio and a percentage in keeping with the theme of the exercise.

All the subroutines are at the beginning of the listing in lines 20 through 220. This should make it easier if you want to use them in your own programs. Change Line 20 to read X=A/B and Line 190 to read PRINT@162,STRING$(L,"-")"=";:RETURN and you have a basis for using the fraction for division exercises beginning at Line 240. Note that lines 180 and 190 ensure the fraction line has the length of whichever of the two, numerator or denominator, is longer.



The listing: RATIOS

```
10 CLS:GOTO230
20 X=B/A:GOTO40
30 X=(A/B)*100
40 X$=STR$(X)
50 E$=STR$(E):EX=LEN(E$)-LEN(STR
$(INT(E)))
60 IFE=INT(E)ANDX<>INT(X)THENP$=
LEFT$(X$,5)ELSEP$=LEFT$(X$,LEN(S
TR$(INT(X)))+EX)
70 RETURN
80 IFE$=P$THENR=R+1:PRINT@268,"R
IGHT!":A$="ANSWER WAS"+P$:PRINT@
303-INT(LEN(A$)/2),A$;:IFQ=1THEN
PRINT"%":GOTO100ELSE100
90 IFE$<>P$THENWR=WR+1:PRINT@268
,"WRONG!":A$="ANSWER WAS"+P$:PRI
NT@303-INT(LEN(A$)/2),A$;:IFQ=1T
HENPRINT"%"
100 A$=STR$(R)+" RIGHT OUT OF"+S
TR$(R+WR)+" "+CHR$(8):PRINT@336-
INT(LEN(A$)/2),A$+"!"
110 IFR>0THENE=(R+WR)/R:IE$="THA
T IS 1 :"+LEFT$(STR$(E),5)+" OR"
+LEFT$(STR$(R/(WR+R)*100),6)+"%"
:PRINT@367-INT(LEN(IE$)/2),IE$
120 RETURN
130 LINEINPUTA$:A=VAL(A$):IFA<>I
NT(A)ORLEN(A$)>2THENPRINT@130,CH
R$(31):PRINT@130,"";:GOTO130ELSE
RETURN
140 LINEINPUTB$:B=VAL(B$):IFB<>I
NT(B)ORLEN(B$)>3THENPRINT@194,CH
R$(31):PRINT@194,"";:GOTO140ELSE
RETURN
150 LINEINPUTC$:IFLEN(C$)>7THENP
RINT@160+L+9,CHR$(8):PRINT@160+L
+9,"";:GOTO150ELSEC=VAL(C$):RETU
RN
160 LINEINPUTD$:IFLEN(D$)>7THENP
RINT@160+L+LC+12,CHR$(31):PRINT@
160+L+LC+12,"";:GOTO160ELSED=VAL
(D$):RETURN
170 FORX=P TO P2STEP32:PRINT@X,C
HR$(31):NEXT:RETURN
180 IFLEN(A$)>LEN(B$)THENL=LEN(A
$)ELSEL=LEN(B$)
190 PRINT@162,STRING$(L,"-")" =
1 :";:RETURN
200 FORD=1TO1500:NEXT:RETURN
210 PRINT@487,"PRESS ANY KEY.";
220 K$=INKEY$:IFK$=""THEN220ELSE
RETURN
230 CLEAR200
240 PRINT@139,"<RATIOS>",,,,,"BY
 KEIRAN KENNY, THE HAGUE, 1985"
250 PRINT@320,"<ENTER> YOUR NAME
, PLEASE (MAX. 14 CHARACTERS==>"
;:LINEINPUTNM$
260 IFLEN(NM$)>14THENP=320:P2=41
6:GOSUB170:GOTO250
270 PRINT@422,"INSTRUCTIONS? Y/N
";
280 GOSUB220
290 IFK$="Y"THENCLS:GOTO640
300 IFK$="N"THENCLS:GOTO320
310 GOTO280
320 PRINT@128,"MAKE YOUR CHOICE:
":PRINT:PRINTTAB(5)"rANDOM NUMBE
RS, OR":PRINT:PRINTTAB(5)"eNTER
YOUR OWN NUMBERS.":PRINT:PRINTTA
B(5)"PRESS <R> OR <E>:";
330 GOSUB220
340 IFK$="R"THENX=RND(-TIMER):V=
1:PRINT"R":GOTO370
350 IFK$="E"THENV=0:CLS:GOTO380
360 GOTO330
370 PRINT:W$=NM$+" WANTS A NUMER
ATOR":PRINT@400-LEN(W$)/2,W$:PRI
NT@456,"NO LARGER THAN:";:LINEIN
PUTCR$:IFVAL(CR$)<10RLEN(CR$)>2T
HENP=448:P2=448:GOSUB170:PRINT@4
48,"";:GOTO370ELSEGOSUB200:CLS:G
OTO400
380 CLS:IFK$="E"ORK$="Y"THENPRIN
T@32,"ENTER NUMERATOR (THE TOP N
UMBER IN A FRACTION)."
390 PRINT@130,"";:GOTO410
400 IFK$="R"ORK$="Y"THENA=RND(VA
L(CR$)):A$=RIGHT$(STR$(A),LEN(ST
R$(A))-1):PRINT@130,A$:GOTO440
410 GOSUB130:P=32:P2=64:GOSUB170
420 PRINT@32,"ENTER DENOMINATOR
(THE BOTTOM   NUMBER)."
430 PRINT@194,"";:GOTO460
440 B=A+RND(VAL(CR$))
450 B$=RIGHT$(STR$(B),LEN(STR$(B
))-1):PRINT@194,B$:GOSUB180:GOTO
490
460 GOSUB140:GOSUB180
470 IFB<A THENPRINT@224,"TOO SMA
LL! TRY AGAIN!":GOSUB200:P=194:P
2=224:GOSUB170:GOTO430
480 P=32:P2=64:GOSUB170
490 PRINT@32,"ENTER NUMBER AFTER
 COLON (WITH  DECIMAL WHERE NECE
SSARY).
500 PRINT@160+L+9,"";:P=320:GOSU
B150:E=C
510 GOSUB20:GOSUB80
520 PRINT@32,"ENTER THE EQUIVALE
NT PERCENTAGE (WITH DECIMALS IF
NECESSARY)."
530 LC=LEN(C$):PRINT@160+L+LC+10
,"= ";:GOSUB160:PRINT@160+L+LC+1
1+LEN(STR$(D)),"%":E=D
540 GOSUB30:P=256:P2=352:GOSUB17
0:Q=1:GOSUB80
550 PRINT@457,"ANOTHER? Y/N"
560 GOSUB220
570 IFK$="Y"AND(R+WR)/5=INT((R+W
R)/5)THENQ=0:CLS:GOTO320
580 IFK$="Y"ANDV=1THENCLS:Q=0:GO
TO400
590 IFK$="Y"ANDV=0THENQ=0:GOTO38
0
600 IFK$="N"THENCLS:GOTO620
610 GOTO560
620 TK$="THANKS "+NM$+"!":PRINT@
112-INT(LEN(TK$)/2),TK$:PRINT@16
0," I HOPE YOU LIKED THE TEST AN
D   LEARNED SOMETHING FROM IT.":
PRINT@455,"TO END PROGRAM":GOSUB
```

```
21Ø
63Ø CLS:END
64Ø PRINT@7,"***INSTRUCTIONS***"
:PRINT@32,"GIVEN THE NUMERATOR (
THE TOP    NUMBER) OF A FRACTION
 LIKE THIS:":PRINT@1Ø1,CHR$(159)
:GOSUB2ØØ
65Ø PRINT@1ØØ,3:GOSUB2ØØ:PRINT@8
6,CHR$(31):PRINT@85,";"
66Ø PRINT@96,"THEN THE DENOMINAT
OR (BOTTOM    NUMBER), LIKE THIS
:":PRINT@164,3:PRINT@197,"-":PRI
NT@229,CHR$(159):GOSUB2ØØ
67Ø PRINT@228,4:GOSUB2ØØ:PRINT@1
35,CHR$(31):PRINT@135,";"
68Ø PRINT@16Ø,"<ENTER> THE NUMBE
R AFTER THE    COLON (':') LIKE
THIS:":PRINT@228,3:PRINT@261,"-
= 1 : "+CHR$(159):PRINT@292,4:GO
SUB2ØØ
69Ø PRINT@268,1.33:GOSUB2ØØ:PRIN
T@2Ø4,CHR$(31):PRINT@2Ø3,";"
7ØØ PRINT@224,"AND THEN CONVERT
THE FRACTION    INTO A PERCENTAGE
 LIKE THIS:":PRINT@292,3:PRINT@3
25,"- = 1 : 1.33 = "+CHR$(159):P
RINT@356,4:GOSUB2ØØ
71Ø PRINT@34Ø,"75%":PRINT@416,"I
'M NOT GOING TO TELL YOU HOW TOD
O YOUR ARITHMETIC!":K$=INKEY$:GO
SUB21Ø
72Ø CLS:PRINT@Ø,"YOU CAN <ENTER>
 YOUR OWN NUMBERSABOVE AND BELOW
 THE LINE OR LET THE COMPUTER EN
TER A NUMERATOR  AND A DENOMINAT
OR AS RANDOM     NUMBERS. IN THI
S CASE YOU CAN   CHOOSE THE HIGH
EST VALUE FOR THENUMERATOR (MAXI
MUM 99)."
73Ø PRINT:PRINT"BEGIN WITH SMALL
 NUMBERS AND TRYTO CALCULATE THE
 ANSWERS IN YOURHEAD AS AN EXERC
ISE IN MENTAL    ARITHMETIC. TRY
LARGER NUMBERS   TOO AS AN EXERCI
SE IN USING YOURCALCULATOR OR IN
 MAKING CALCUL- ATIONS ON PAPER.
":GOSUB21Ø
74Ø CLS:PRINT@Ø,"<ENTER> NUMBERS
 AS FOLLOWS:":PRINT:PRINT"NUMERA
TOR, 1-2 DIGITS, NO DECI- MAL. M
AXIMUM 99;"
75Ø PRINT:PRINT"DENOMINATOR, 1-3
 DIGITS, NO DEC-IMAL. MAXIMUM 99
9;"
76Ø PRINT:PRINT"ANSWERS, MAXIMUM
 WITH DECIMAL    SEVEN CHARACTERS
. DO NOT TYPE    THE '%' SIGN."
77Ø PRINT:PRINT"ILLEGAL ENTRIES
WILL SELF-ERASE AND YOU CAN TRY
AGAIN!"
78Ø GOSUB21Ø:CLS:GOTO32Ø
```

HARDWARE PROJECT

Do-It-Yourself Video Output Board

By Tim McIntosh

I use my CoCo quite extensively on a daily basis. The bulk of my work revolves around word processing. I quickly discovered that using a television for a monitor was inadequate.

About a year ago, I began to look around for a monitor and a way to drive it. I was unaware of the many advertisers who market just such devices. So, I set out to build my own. The project presented here required about a half day's labor and $8 worth of parts.

As the schematic shows, this monitor driver requires two common transistors and a handful of support components. Most experimenters should have an abundant supply of these. The driver works with color as well as monochrome composite video monitors. The pin numbers given in the schematic refer to the MC6847 VDG chip in the Color Computer. The descriptions of these pins are as follows: Pin 1 — GND (signal ground); Pin 17 — Vcc (+5v DC); and Pin 28 — Composite video signal.

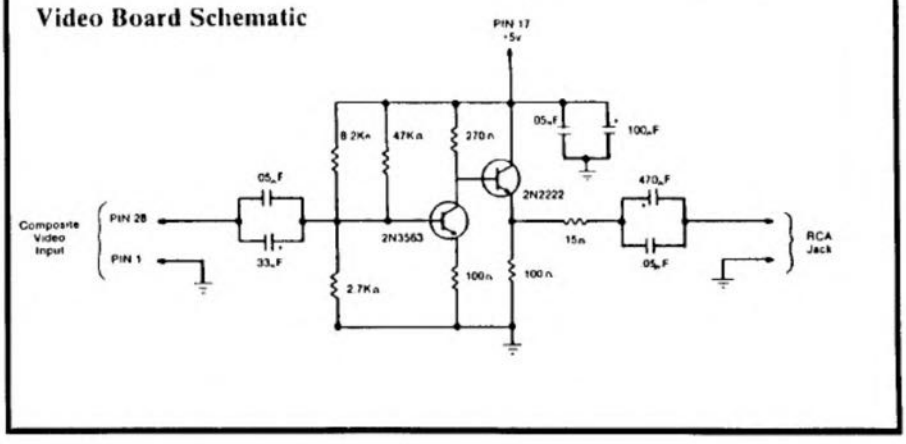
Construction of the driver is relatively simple. I used a small project board from Radio Shack to mount the components. I ran the connecting wires (any standard jumper wire will do) to the points on the circuit board where the 6847 is soldered. Be careful when counting to the proper pin locations. A mistake could damage your CoCo.

Many of you will want to run a length of shielded cable with an RCA plug on it out of the computer directly to the monitor. I chose to tidy up the project by mounting an RCA jack on the back of the CoCo and running the shielded wire to this. Not only did this improve the appearance of the project, but I can now relocate my CoCo without trailing a mess of wires behind me.

Although I use this new monitor driver on a CoCo 2, it should work on any Color Computer as well. Remember: Opening your CoCo voids the warranty. It is not advisable to attempt this project without some prior knowledge of electronics.

(Any questions you may have about this project can be sent to the author at 416 Oak St., Greenville, AL 36037, 205-382-7358. Please enclose an SASE when writing.)

Video Board Schematic



RATTLE CLATTER

by Michael B. Kromeke

This program allows you to see some of the internal workings of a running engine. The engine is started by pressing 'S'. As it starts, the car key moves into the ignition switch and turns. The oil warning light comes on, the fuel gauge goes from empty to full, the fan starts to turn and the oil warning light goes off. The four pistons start moving up and down and the spark plugs fire.

When the engine is running, press 'T' for a trouble selection. This puts one of five troubles into the engine in random and information about that kind of trouble is displayed. You will be told whether you have correctly identified the problem or not.

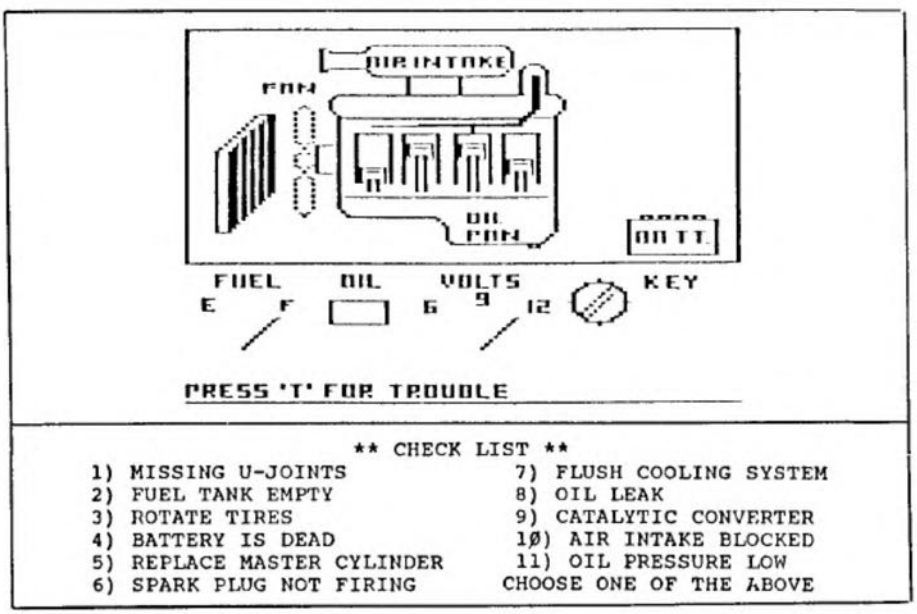
If an incorrect answer is picked, you are sent back for another look at the trouble. A correct answer returns you to a properly running engine. After correcting all five troubles, your score and skill level are displayed. The three skill levels are beginner, shade tree mechanic and master mechanic.

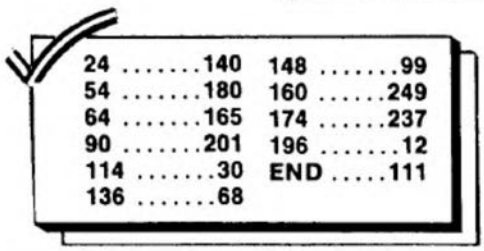
Anytime the engine is running, even improperly, you may stop and start it by pressing 'S'.

To run this program with only 16K, delete all REM statements. Deleting the following lines will not affect the program: 30, 34, 38, 42, 48, 70, 84, 88, 92, 96, 100, 106, 112, 120, 176, 180, 184, 188 and 216.

Variables

A	=	answer
B	=	miscellaneous counter
C1	=	cylinder 1
C2	=	cylinder 2
C3	=	cylinder 3
C4	=	cylinder 4
D	=	number of trouble
E	=	oil leak counter
F	=	vertical location of cylinder
G	=	picture of piston
H	=	picture of key in slot
I & M	=	trouble in engine
R	=	number of right answers
W	=	wrong answer counter
Y	=	horizontal location of cylinder
A$	=	draw picture of fan not turning
Z$	=	miscellaneous input





The Listing: ENGINE

```
2 CLEAR200:DIM A,B,C1,C2,C3,C4,D
,E,F,G(7),H(18),I,M,R:C1=12:C2=3
:C3=9:C4=6:B=RND(-TIMER)
4 GOSUB196:GOTO50
6 ON F GOSUB36,40,44,40,36,32,36
,40,44,40,36,32:RETURN
8 SCREEN1:Y=0:F=C1:GOSUB6:C1=C1+
1:IF C1=13 THEN DRAW"BM86,51C4ND
2":PLAY"T255A":GOSUB28:DRAW"C1ND
2":C1=1
10 Y=23:F=C2:GOSUB6:C2=C2+1:IFC2
=13 AND D<>3THEN DRAW"BM109,51C4
ND2":PLAY"A":GOSUB28:DRAW"C1ND2"
:C2=1
12 IF C2=13 THEN C2=1
14 Y=46:F=C3:GOSUB6:C3=C3+1:IFC3
=13 THEN DRAW"BM132,51C4ND2":PLA
Y"A":GOSUB28:DRAW"C1ND2":C3=1
16 IF D=4 THEN PSET(126+RND(44),
112+RND(7),2):E=E+1:IF E>100 THE
N:PAINT(80,144),4,3:SOUND2,20:GO
TO220
18 Y=69:F=C4:GOSUB6:C4=C4+1:IFC4
=13 THEN DRAW"BM155,51C4ND2":PLA
Y"A":GOSUB28:DRAW"C1ND2":C4=1
20 Z$=INKEY$:IFZ$=""THEN8
22 IF Z$="T"AND D=0THEN I=I+1:D=
M(I):GOTO50
24 IF Z$="C"AND D<>0 THEN122
26 IF Z$="S"THEN116 ELSE8
28 FORB=1TO100:NEXTB:RETURN
30 'CYLINDER MOVEMENT POSITION 1
32 PUT(80+Y,55)-(92+Y,71),G,PSET
:RETURN
34 'POSITION 2
36 PUT(80+Y,59)-(92+Y,75),G,PSET
```

```
:RETURN
38 'POSITION 3
40 PUT(80+Y,63)-(92+Y,79),G,PSET
:RETURN
42 'POSITION 4
44 PUT(80+Y,67)-(92+Y,83),G,PSET
:RETURN
46 PRINT@488,"PRESS <RETURN>";:L
INEINPUTZ$:RETURN
48 'MAIN SCREEN
50 PMODE3,1:PCLS:SCREEN1,0:DRAW"
BM0,0C2R255D120L255U120BM70,46C4
R100D40L100U40H4U4E4R100F4D4G4L4
4BU12U10R20E4U8H4L60G4L16H2L2D12
R2E2R16F4R16NR22D10BM0,191R255"
52 A$="BM60,68C3H10U14E4F4D14G8D
4F4NE4G4D14F4E4U14H4":DRAW"BM70,
86C4D6F4R40F6D6F2R42L2D2L3U2R5E6
U18BM70,60L10D14NR10":DRAW"XA$;B
M154,34RU12E4R4F4D12BM130,94NR4D
5R4U5BR4D5BR4NU5R3BL14BD4ND5R4D2
NL4BR4NR4D3U5R4D5BR4U5F5U5"
54 DRAW"BM36,28C2NR4D2NR4D3BR8U5
R4D2NL4D3BR4U5RF5U5BM84,19C4U3NR
4U2R4D5BR4U5BR4ND5R4D2L4UF4BR6U5
BR4ND5RF5U5BR4R2ND5R2BR4ND5R4D2N
L4D3BR4U3R2NF3NE2L2U2BR8NR3D2NR2
D3R3"
56 DRAW"BM204,98C4R40D18L40U18R6
U2R4D2R4U2R4D2R4U2R4D2R4U2R4D2L3
0BD6ND6C3R4D3NL4D3NL3BR4U6R4D3NL
4D3BR8U6L2R4BR4R4L2D6BR4R1"
58 DRAW"BM78,79C4U25R16D25BR7U25
R16D25BR7U25R16D25BR7U25R16D25BM
86,50C3R68E8U18L2D18G7L44BM14,64
C3E8NR6E4NR6E8R6G20L6D40R6NU40E2
0U39":POKE178,207:PAINT(34,55),,
3:POKE178,3:GOSUB86
60 DRAW"BM14,126C3NR4D3NR2D3BR8N
U6R4U6BR4NR4D3NR2D3R4BR4NR4U6BR3
4D6R4U6NL4BR4D6BR4NU6R4BR32H2U4B
R5D4G2BR6NR4U6R4D6BR4NU6R4BR4U6N
L2R2BR4NR4D3R4D3L4BM8,138NR4D3NR
2D3R4"
62 DRAW"BR32U3NR2U2R4BR18NR26D12
R26U12BR18NR4D6R4U3L4U3BR24BU4NR
4D3R4NU3D3NL3BR20BD4U6BR4R4D3L4D
3R4BM26,164C2H16BM214,54C4ND8R12
D2NL12D6L12R5D4BR3U6L3D2":GET(21
4,50)-(226,66),G,G:LINE(214,50)-
(226,66),PRESET,BF
64 DRAW"BM136,164C2":IF D=2 THEN
DRAW"H16"ELSEDRAW"E16"
66 CIRCLE(220,58),14,3,.9:DRAW"B
M212,64C3NE14F3E14":PAINT(220,58
),4,3:GET(206,46)-(234,70),H,G:L
INE(206,46)-(234,70),PRESET,BF:G
OSUB102
68 IF D=5 THEN PAINT(64,20),2,4:
GOTO182
70 'START ENGINE
72 IF INKEY$<>"S"THEN72 ELSEGOSU
B90
74 IF D=1 THEN186
76 IF D=2 THEN178
78 GOSUB94:FORB=0TO52STEP3:LINE(
50,42+B)-(58,42+B),PRESET:PLAY"T
140O1AF":NEXTB:PAINT(80,144),1,3
:GOSUB98
80 IF D=3 OR D=4 THEN GOSUB108 E
LSE GOSUB114
82 GOTO8
84 'DRAW KEY,CIRCLE & WORD KEY
86 CIRCLE(190,140),14,3,.9:DRAW"
BM188,150C3U20R4D20BR20NH4E4F4E4
R6F6R6E6U4H6L6G6L22ND4BR29BD2ER2
F2G2L2H2BL24BU12C2U6D3R2NE3F3BR6
NR4U3NR2U3R4BR4D2R4NU2L2D4":PAIN
T(216,144),2,3:RETURN
88 'KEY TO START POSITION
90 LINE(208,135)-(248,153),PRESE
T,BF:PLAY"T80ABCDEF":PUT(176,128
)-(204,152),H,PSET:RETURN
92 'FUEL GAGE FROM E TO F
94 PAINT(80,144),4,3:GOSUB28:DRA
W"BM26,164C1NH16":GOSUB28:DRAW"C
2NU16":GOSUB28:DRAW"C1NU16":GOSU
B28:DRAW"C2E16":GOSUB28:RETURN
96 'BLANK BOX
98 LINE(44,176)-(255,190),PRESET
,BF:RETURN
100 'DRAW PRESS S FOR START
102 DRAW"BM0,182C3ND6R4D3L4D3BR8
U6R4D3L4R2F3BR4NR4U3NR2U3R4BR4NR
4D3R4D3NL4BR4R4U3L4U3R4BR8ND2BR4
NR4D3R4D3NL4BR4BU4U2BR8R2NR2D6BR
6NR4U6R4D6BR9BU6NR4D3R4D3NL4BR6U
6L2R4BR4ND6R4D3NL4D3BR4U6R4D3L4R
2F3BR6U6L2R4"
104 DRAW"BR8NR4D6R4U6BR4ND6R4D3L
4RF3BR10BU6NR4D3R4D3NL4BR6U6L2R4
BR4NR4D6R4U6BR4ND6R4D3L4":RETURN
106 'DRAW C FOR CHECK LIST
108 DRAW"BM45,182C3ND2BR4NR4D6R4
BR4BU4U2BR7NR4D3NR3D3BR8NR4U6R4D
6BR4U6R4D3L4R2F3BR10NR4U6R4BR4D3
NR4D3BR4U6BR4NR4D3NR2D3R4BR4NR4U
6R4BR4D3R2NE3NF3L2D3"
110 DRAW"BR16NU6R4BR4U6BR4NR4D3R
4D3NL4BR6U6NL2R2":RETURN
112 'DRAW T FOR TROUBLE
114 DRAW"BM45,182C3ND2BR4R2NR2D6
BR6BU4U2BR7NR4D3NR2D3BR8NR4U6R4D
6BR4U6R4D3L4R2F3BR12U6NL2R2BR4ND
6R4D3L4RF3BR4NR4U6R4D6BR4NU6R4U6
BR4D6R4U3NL4U3NL4BR4D6R4BR4NR4U3
NR3U3R4":RETURN
116 DRAW"XA$;BM26,164C1NE16":GOS
UB28:DRAW"C2NU16":GOSUB28:DRAW"C
1NU16C2H16":LINE(176,126)-(250,1
52),PRESET,BF:GOSUB98:GOSUB86:GO
SUB102:GOTO72
118 PRINT@487,"PRESS <RETURN> ";
:LINEINPUTZ$:RETURN
120 'CHECK LIST
122 CLS:PRINT@8,"** CHECK LIST *
*":PRINT@68,"1) MISSING U-JOINTS
":PRINT@100,"2) FUEL TANK EMPTY"
124 PRINT@132,"3) ROTATE TIRES":
PRINT@164,"4) BATTERY IS DEAD"
126 PRINT@196,"5) REPLACE MASTER
 CYLINDER":PRINT@228,"6) SPARK P
LUG NOT FIRING"
128 PRINT@260,"7) FLUSH COOLING
SYSTEM":PRINT@292,"8) OIL LEAK"
130 PRINT@324,"9) CATALYTIC CONV
ERTER":PRINT@355,"10) AIR INTAKE
 BLOCKED"
132 PRINT@387,"11) OIL PRESSURE
LOW"
134 PRINT@450,"CHOOSE ONE OF THE
 ABOVE";:INPUTZ$:A=VAL(Z$)
136 IF A<1 OR A>11 THEN SOUND1,1
:GOTO134
138 ON A GOTO140,142,146,150,154
,156,160,164,168,172,174
140 CLS:PRINT@4,"** MISSING U-JO
INTS **":PRINT@65,"THE U-JOINTS
CONNECT THE DRIVE- SHAFT TO THE
DIFFERENTIAL WHICH TURN THE WHEE
LS. THE U-JOINTS  ARE NOT IN TH
IS PROGRAM .":GOTO190
142 CLS:PRINT@6,"** FUEL TANK EM
PTY **":PRINT@65,"LOCATED ON THE
 LEFT SIDE OF THE SCREEN, THE E
STANDS FOR EMPTY  AND F STANDS F
OR FULL. GASOLINE IS MADE FROM P
ETROLEUM THAT IS  A NATURAL FLAM
MABLE LIQUID"
144 PRINT" HYDROCARBON MIXTURE T
HAT COST  AN ARM AND A LEG.":GO
TO190
146 CLS:PRINT@6,"** ROTATE TIRES
 **":PRINT@65,"YOU CAN INCREASE
THE LIFE OF  YOUR TIRES BY ROT
ATING THEM  ONCE EVERY 5000 M
ILES OR SO.  CARS WITH FRONT-W
HEEL DRIVE  SHOULD NOT BE ROT
ATED. TO GET  MAXIMUM TIRE LIFE
 YOU SHOULD"
148 PRINT" CHECK AIR PRESSURE EV
ERY 30  DAYS WHEN TIRES ARE C
OLD FOR  MANUFACTURER RECOMMEN
DED PRES-  SURE.":GOTO190:GOSUB
118
150 CLS:PRINT@6,"** BATTERY IS D
EAD **":PRINT@65,"THIS CAR IS RU
NNING ON A 12  VOLT SYSTEM. C
HECKING THE  VOLTAGE READIN
G ON THE INSTRU-  MENT PANEL WE
CAN SEE IF WE  HAVE THE PROPE
R READING. ANY-  THING ABOVE 9.
6 VOLTS AFTER 15"
152 PRINT" SECONDS OF CRANKING T
HE ENGINE  SHOULD BE ADEQUATE.":
GOTO190
154 CLS:PRINT@2,"** REPLACE MATE
R CYLINDER **":PRINT@65,"THE MAS
TER CYLINDER IS NOT  SHOWN I
N THIS DIAGRAM. IT IS A  VERY IM
PORTANT PART OF THE  BRAKE S
YSTEM.":GOTO190
156 CLS:PRINT@2,"** SPARK PLUG N
OT FIRING **":PRINT@65,"THE IGNI
TION SYSTEM IS DESIGNED TO IGNIT
E THE AIR/FUEL MIXTURE  THAT IS
DRAWN INTO THE ENGINE  THROUGH
THE CARBURETOR. IF THE  SPARK PL
UGS FAIL TO FIRE THEN"
158 PRINT" YOUR ENGINE WILL NOT
RUN PROP-  ERLY.":GOTO190
160 CLS:PRINT@2,"** FLUSH COOLIN
G SYSTEM **":PRINT@65,"THE COOLI
NG SYSTEM SHOULD BE  DRAINED A
ND FLUSHED EVERY TWO  YEARS, AN
D NEW COOLANT ADDED.  THE RADIA
TOR IS SHOWN IN THIS  DIAGRAM A
ND HELPS KEEP THE"
162 PRINT" ENGINE FROM OVER HEAT
ING BUT  DOES NOT EFFECT ITS R
UNNING.":GOTO190
164 CLS:PRINT@9,"** OIL LEAK **"
:PRINT@65,"KEEP YOUR EYES ON THE
 OIL PAN . IF IT STARTS TO LEAK
WE WILL  SEE THE OIL WARNING L
IGHT COME"
166 PRINT" ON JUST BEFORE THE EN
GINE  FREEZES UP,DO TO LACK
 OF"," LUBRICANT.":GOTO190
168 CLS:PRINT@3,"** CATALYTIC CO
NVERTER **":PRINT@65,"THIS UNIT
CONVERTS HYDROCAR-  BONS AND C
ARBON MONOXIDE INTO  CARBON DIO
XIDE AND WATER VAPOR. EXPECTED S
ERVICE LIFE IS ABOUT"
170 PRINT" 50,000 MILES AND IS N
OT IN THIS PROGRAM.":GOTO190
172 CLS:PRINT@3,"** AIR INTAKE B
LOCKED **":PRINT@65,"THE AIR CLE
```

continued on Page 31

CoCo Better Again!

NEW YORK, JULY 30 —

After anticipating the new Color Computer for more than a year, the finished product is more than well worth the wait. The new machine is called a Color Computer 3 (not particularly surprising) but it might be called Warp Factor 3 instead.

First, the obvious things. The CoCo 3 general specifications go something like this:

- 128K RAM standard, upgradable to 512K in one step only.
- 64 different colors, 16 available at a time.
- Graphics resolution of 640 by 192.
- Standard Microsoft BASIC, Extended BASIC and Disk BASIC, with a number of very fancy enhancements written by Microware Systems Corp. (the OS-9 people).
- A new keyboard much the same as the deluxe keyboard now offered (with a CONTROL and ALTERNATE key and two function keys), but with a newly designed cursor keypad in a diamond configuration.
- A slightly new color scheme for the case, using the now-familiar cream color and a light gray around the edge of the keyboard.
- A choice of 32, 40 and 80 column widths with upper- and lowercase and true descenders.
- Fully compatible with all the software you own now. The caveat to that is software writers cannot have made undocumented calls to the ROM. Tandy developed guidelines for software developers years ago — if authors followed those "rules," their software will work with CoCo 3.
- OS-9 Level II operating system for a true multi-tasking, multi-using environment. More on this later.
- Fully compatible with all the hardware you own now. Just unplug your disk drive, tape recorder, printer and so on, replace your present "box" with a CoCo 3, and plug everything in again. It all works. (There will be a small problem running OS-9 with older models of the Multi-Pak Interface. Tandy will make the fix for about $6, excluding installation.)
- A new interface for a new analog RGB monitor available from Tandy. The new monitor's color scheme matches the CoCo 3.
- Retail price: $219.95.

On the surface, these changes are pretty exciting, but not astounding. Yet CoCo 3 is an astounding machine. It is not the changes in the specifications so much as what has been done with the specifications that makes the CoCo, once again, the best possible computer buy for the home or small business market, bar none.

An example: Combining the new analog RGB monitor (a CM-8 by Tandy's designation) and CoCo 3, the resolution on graphics screens appears better than that of an MS-DOS computer. Why? By using analog technology, the resolution appears to be of higher quality than it really is. I can see software writers frothing at the mouth to get at this stuff.

Another example: The CoCo 3 can run at about 2 megahertz, unlike the CoCo and CoCo 2. But, by designing and building a special chip, called a GIME Chip, *effective speed* is increased even more. In a quickie benchmark we performed here, CoCo 3 animation zapped an IBM PC and a Macintosh. CoCo 3's bouncing ball demonstration is faster and smoother than Amiga's or the Atari ST's.

There are two versions of what the term "GIME" stands for. I like the one that says Tandy went out and said they needed to be able to do this thing, that thing and so on, so please "Gimme a chip that'll work." Officially, though, GIME stands for Graphic Interrupt Memory Enhancer. Take it either way, this li'l' fellow makes CoCo 3 a super special machine.

I confess to not understanding hardware like the Tony DiStefanos and Marty Goodmans of the world. And I am certain that, over the coming months, they will be able to tell you a great deal of technical stuff about it. But suffice it to say now that this new chip manages memory through interrupts, and speeds everything along its way far more swiftly than the old VDG and SAM chips (which are not in the CoCo 3) did.

For one thing, GIME makes true multi-user, multi-tasking possible. It allows some of the most sophisticated (and easy-to-use) windowing available on any computer available today. As an example, you could be connected through a modem to someone else with a CoCo 3 in one window, and have a graphics drawing program in a second window (windows mean both programs are running at the same time and are displayed on the screen at the same time). The person on the other end could be explaining how something looks and while he is explaining it, you could be drawing it. Then, to be sure you got it right, you could send your drawing to the telecommunications program window and it would appear on the other person's screen. That person could make some changes and send it back to you, and so on!

continued on Page 31

The last installment of the point-and-pick interface

CoCo Mouse: The Final Chapter

By Steve Bjork

To refresh your memory, *The Mouse* is an assembly language program that displays and moves a cursor and prints upper- and lowercase text on the Hi-Res graphics screen. A BASIC program may access this interface by 13 user functions and the PRINT command.

Last month we covered the assembly language side of the USR function and PRINT command, along with the code for drawing the cursor (pointer). We also looked at how the joystick and button information is polled.

This month we have the second half of the source code, bell and clock sounds and the Hi-Res screen text driver and its character shape data.

Before getting started I want to remind you that *The Mouse* and its source code are copyright 1986 by SRB Software with all rights reserved. It is presented in RAINBOW magazine for private use only, and can't be used (all or part) in any marketed product without written consent from SRB Software.

Steve Bjork has been a programmer for over 15 years. In his association with Datasoft he has authored such programs as Zaxxon, Sands of Egypt *and* Mega-Bug. *He now handles product development for his own company, SRB Software, and has produced* Stellar Life Line, Ghana Bwana *and* PitFall II *among others. Steve lives in Simi Valley, California*

Makin' Some Noise!

The first subroutine of Listing 1 (MOUSE3/ASM) generates the bell sound used in function 7. The bell sound is generated by playing a steady tone that decreases in volume as it is played (see Figure 1).

So the sound can be heard, the bell routine first enables sound to the TV and selects the six-bit DAC as the sound device by calling SNDON. Next, the interrupts are disabled so the timing loops are not distorted and the starting volume is selected.

Lines 600 to 1900 generate the envelope in Figure 1. After bell sound is completed, the interrupts are turned back on and the TV sound is disabled.

The click sound does not change in volume like the bell, but does become lower in pitch as it is played. With click being so fast, the interrupts do not need to be turned off as before. Lines 4600 through 5400 flip the bits of the six-bit DAC with a longer delay each time to generate its wave form.

PRINT @

The Hi-Res screen text driver has 32 characters per line, just like the standard green text screen. But eight lines must be added to make a total of 24 and give an extra 256 positions. This does present a little problem; the PRINT command's '@' can't address these new lines. So *The Mouse* intercepts the PRINT command to handle the '@' function for screen location in NPRINT, lines 6300 to 9400 of Listing 1.

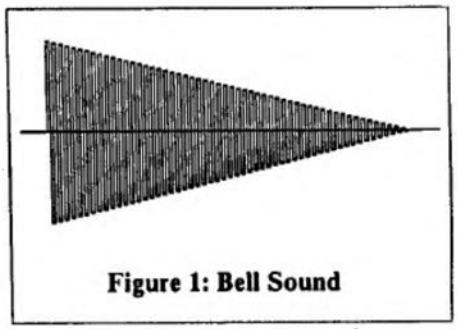


Figure 1: Bell Sound

NPRINT jumps to the old PRINT command vector (in OLDPRT) if there is no '@' after the word PRINT. If there is an '@', a 16-bit number subroutine of Color BASIC ROM is called to find the new location. If this number is larger then the window size, then an FC Error is generated. With everything OK, the cursor is moved to the new location and

Tandy ELECTRONICS

Exclusive Offer To Rainbow/Coco Readers.

Leather Disk Holder

Save $26

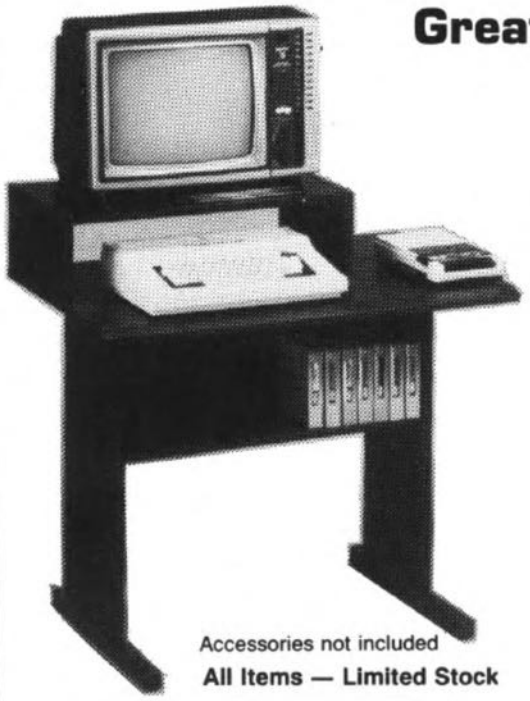
Reg 45.95

19.95

Accessories not included

Give your disks a home that's worthy of them with this genuine leather disk box. They won't last long at this remarkable saving. It'll hold up to ten 13.3cm disks in comfort. Closes securely to prevent any accidents. A stylish accessory. 26-1349

Great Value on Our Beginner's Desk



Save $100

99.95

Reg 199.95

Exclusive Offer!

Stop using your valuable desk space for your computer. Give it a permanent home that was specifically designed for it. Ideal for the Color Computer, this workstation features a monitor/printer platform and storage shelf for all your software and instruction manuals. Measures 66 x 81.2 x 50.2cm. Comes ready for easy assembly with all pieces forming simple, yet durable linkages. 26-1354

Accessories not included

All Items — Limited Stock

Versatile Computer Table

Save $100

Reg 279.95

179.95

Small enough to let you set your system up anywhere. Large enough to accommodate the Color Computer or Tandy 1000. It's easy to assemble and comes with full instructions. Measures 68.6 x 121.9 x 61cm. 26-1324

A Functional Printer Stand

Save $100

99.95

Reg 199.95

Accessories not included

Unlike just any piece of furniture, this piece was specifically designed for printers. Large enough for wide-bed printers. With bottom feed paper slot and storage shelf. A must for any computer user. 66 x 68 x 50cm. 26-1355

Our Lowest Priced Daisy Wheel is Now Even Lower

Save $300

$699

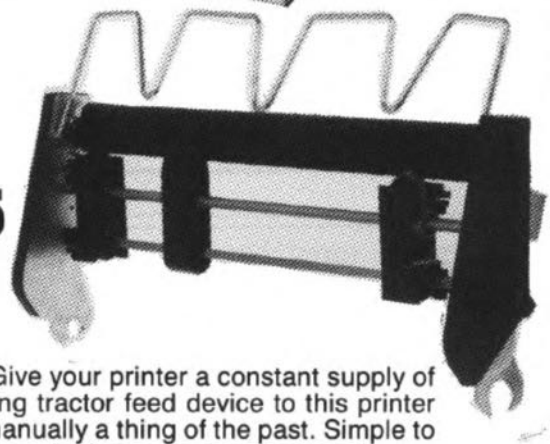
Reg **999.00**

Letter perfect printing now for less than $1000. Uses interchangeable 100-character print wheels so you can pick the font you want. Prints over 200 words per minute at 10 or 12 cpi. With forward or reverse paper-feed. Comes complete with "Courier 10 Typewriter" style wheel and carbon printer ribbon so it's ready to go. Buy now at this price and SAVE! 26-1278

Save $100

$129.95

Reg **229.95**



DWP 220 Tractor Feeder. Give your printer a constant supply of paper the easy way. Matching tractor feed device to this printer will make feeding in paper manually a thing of the past. Simple to attach or remove whenever you don't require constant feed. 26-1444

We've Got Your Paper Needs

39.95 EACH

Choose from 381 x 279cm fanfold paper with blue horizontal stripes (900 sheets) 26-9315. 241 x 279cm blank white word-processing paper (900 sheets) or 241 x 279cm fanfold paper with blue horizontal stripes (1,000 sheets) 26-9315, 26-9316, 26-9317

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

Listing 1:

```
00100 * SOUND AND SCREEN DRIVERS
00200 PING    BSR     SNDON           GET SOUND FROM 6 BIT DAC
00300         PSHS    CC              SAVE IRQ FLAGS
00400         ORCC    #$50            TUEN OFF IRQS
00500         LDA     #230            HOW LONG TO DO SOUND
00600 PING1   BSR     PING5           DO A TIME DELAY
00700
00800         TFR     A,B             GET TIME COUNT
00900         ANDB    #$F7            USE ONLY THE TOP 5 BITS
01000         ORB     #2              SET PRINTER BIT HIGH
01100         STB     $FF20           SEND IT OUT THE DAC PORT
01200
01300         BSR     PING5           DO A TIME DELAY
01400         LDB     #2              CLEAR ALL BITS BUT PRINTER
01500         STB     $FF20           ON DAC PORT
01600         DECA                    MAKE BALL SOUND SMALLER
01700         DECA                    BY 2
01800         CMPA    #18             IS BELL DONE?
01900         BHS     PING1           NO, THEN LOOP
02000         PULS    CC              TURN ON IRQS
02100 SNDOFF  LDA     $FF23           TURN OFF DAC SOUND
02200         ANDA    #$F7            BY RESETTING THE BIT
02300         STA     $FF23
02400         CLRB                    MAKE RETURN ZERO
02500         RTS                     AND RETURN
02600
02700 SNDON   LDA     $FF23           TURN ON THE SOUND BY
02800         ORA     #8              SETTING THE SOUND ON BIT
02900         STA     $FF23
03000         LDA     $FF01           GET LSB OF JOY/AUDIO
03100         ANDA    #$FF-8          PORT AND RESET IT
03200         STA     $FF01           AND PUT IT BACK
03300         LDA     $FF03           GET MSB OF JOY/AUDIO
03400         ANDA    #$FF-8          PORT AND RESET IT TOO
03500         STA     $FF03           AND PUT IT BACK
03600         RTS                     NOW EXIT
03700
03800 PING5   LDB     #180            TIME DELAY USED BY BELL (PING)
03900 PING6   DECB
04000         BNE     PING6
04100         RTS
04200
04300 *MAKE A SMALL CLICK SOUND
04400 CLICK   BSR     SNDON           TURN SOUND ON TO DAC
04500         LDA     #10             START TIME DELAY SHORT
04600 CLICK1  TFR     A,B             DO A DELAY (USE COUNT)
04700 CLICK2  DECB                    GET LONGER EACH TIME
04800         BNE     CLICK2
04900         LDB     $FF20           GET DAC PORT
05000         EORB    #$F0            FLIP TOP 4 BITS
05100         STB     $FF20           SAVE SET PORT
05200         INCA                    MAKE DELAY LONGER
05300         CMPA    #76             ALL DONE?
05400         BLO     CLICK1          NO, LOOP BACK
05500         BRA     SNDOFF          ELSE SOUND OFF AND EXIT
05600
05700 FNERR   LDB     #4*2            GO HERE FOR ?FN ERROR
05800         FCB     $8C
05900 SNERR   LDB     #2              GO HERE FOR ?SN ERROR
06000         JMP     $AC46           JUMP TO ERROR (OF BASIC)
06100
06200 *MOUSE'S NEW PRINT COMMAND
06300 NPRINT  BEQ     OPRINT          SKIP IF NO PRINT DATA
06400         CMPA    #64             IS THE FIRST CHAR. A "@"?
06500         BNE     OPRINT
06600         PSHS    A
06700         LDA     $FF22           GET GRAPHICS MODE
06800         ANDA    #$E0            IS SCREEN A PMODE 3 TO 4
06900         EORA    #$E0
07000         PULS    A
07100         BNE     OPRINT          NO, THEN USE OLD PRINT COMMAND
07200
07300         JSR     $B3E4           GET VAL FOR THE @ POSITION
07400         PSHS    U               SAVE U, NEED FOR VARIABLE BLOCK
07500         LEAU    DATA,PCR        GET VARIABLE POINTER
07600         CLR     COUNT,U         CLEAR LINE COUNT
07700 AT1     INC     COUNT,U         ADD ONE TO LINE COUNT
07800         SUBB    WDXLEN,U        SUB ONE LINE WIDTH
07900         SBCA    #0              (DO 16 BIT SUB)
08000         BCC     AT1             LOOP TILL NEG
08100         ADDB    WDXLEN,U        ALWAYS ONE TOO MANY
08200         LDA     COUNT,U         GET LINE COUNT
08300         DECA                    ALWAYS ONE TOO MANY
08400         CMPA    WDYLEN,U        IS THE LINE OUT OF THE WINDOW
08500         BHS     FNERR           FM ERROR IF OUT OF RANGE
08600         STD     YCPOS,U         SAVE THE LINE & COLUMN POSITION
08700         PULS    U               RESTORE U REG.
08800
08900         JSR     $A5             GET NEXT CHAR.
09000         BEQ     OPRINT          END IF END OF PRINT DATA
09100         CMPA    #$2C            IS THE CHAR. A ","?
09200         BNE     SNERR           NO, THEN ?SN ERROR
09300         JSR     $9F             SET FLAGS FOR CHARACTER
09400 OPRINT  JMP     [OLDPRT+DATA,PCR]
09500
09600 PMTAB   FDB     CHR1-PMTAB      TABLE USED FOR PMODE TYPE
09700         FDB     PMODE1-PMTAB
09800         FDB     PMODE2-PMTAB
09900
10000         FDB     WMODE1-PMTAB
```

Pattern	Binary	Hex	Decimal
. . . *** . .	00011100	$1C	28
. . * . . . * .	00100010	$22	34
. . *	00100000	$20	32
. . . *** . .	00011100	$1C	28
. . . . . * .	00000010	$02	2
. . * . . . * .	00100010	$22	34
. . . *** . .	00011100	$1C	28
. . . . .	00000000	$00	0

Figure 2: Dot Format for 'S'

the old PRINT command handles the rest of the line.

Outgoing Intercepters

All characters that would normally go to the text screen are intercepted by *The Mouse*. This is done by redirecting the standard output path through SPRINT at Line 10500. This routine calls CHR (Hi-Res text driver) if a Hi-Res graphics screen is selected. If a Lo-Res graphics or text screen is displayed at the time, then the old standard output path is used.

The routine that prints a character to the Hi-Res screen, CHR, must first decide what mode it is in. These modes are Standard, Set Cursor Position and Set Window. Standard prints the character, does a control code or sets up the other two modes. Set Cursor Position mode takes the next two characters (X and Y position) and moves the cursor (lines 13600 through 15100). The Set Window mode takes the next four characters and sets up a new scroll-protected window (lines 10400 through 15400 of Listing 2).

Standard mode first tests if the character is a control code, 0 to 13, with CHRTAB holding the offset vector for these routines. Some of the routines are Line Feed, Clear Screen, Carriage Return, Set Window and Set Cursor Position.

Next is a test for displayable range (32 to 127). If it is in range, then the character's shape is selected out of CHRDAT shape data. The Shape Data format is one byte (eight dots) wide and eight bytes (lines) high.

MOUSE4/ASM (Listing 2) is the CHRDAT shape data table for all 96 displayable characters. Each character shape consists of one byte (eight dots) wide by eight bytes (scan lines) high. If the bit is a zero, then a black dot is placed on the screen. And of course, if the bit is a one, then a white dot is displayed. Dot format for the letter "S" is shown in Figure 2.

Each line of the CHRDAT shape data table has eight bytes (four word length) per line. Some assemblers do not allow for more then one expression per line. If this is the case with your assembler then all 96 FDB lines must be replaced by four FDB lines with one expression each.

That's all the source code. If you have any questions about *The Mouse* or its source code just drop me a line, or better yet, look for me (6809er) at RAINBOW's Color Computer SIG on Delphi.

(You may contact Mr. Bjork at 2529 Ellington Court, Simi Valley, CA 93063. Please enclose an SASE when writing.) □

```
10100          FDB     WMODE2-FMTAB
10200          FDB     WMODE3-FMTAB
10300          FDB     WMODE4-FMTAB
10400
10500 SPRINT   PSHS    B               SAVE IT'S DATA, I NEED TO USE IT
10600          LDB     $FF22           GET SCREEN MODE
10700          EORB    #$E0            COMPLEMENT HI-RES BITS
10800          ANDB    #$E0            ONLY USE THOSE BITS
10900 * IF THE SCREEN IS IN THE HI-RES (6K) MODE THEN B=0
11000
11100          ORB     $6F             GET I/O PATH NUMBER
11200 * IF THE I/O PATH IS THE SCREEN (ZERO) THEN B STILL WILL BE ZERO
11300
11400          PULS    B               GET B OLD DATA
11500          BEQ     SPRNT1          SKIP IF FOR HI-RES SCREEN
11600 SPRNTO   JMP     [OLDIO+DATA,PCR]        ELSE USE OLD I/O PATH CALL
11700
11800 SPRNT1   LEAS    2,S             REMOVE PATCH-CALL ADDRESS
11900
12000 CHR      PSHS    U,Y,X,D         SAVE ALL REGS.
12100          LEAU    DATA,PCR        GET ADDRESS OF VARIABLES
12200          LDB     FMODE,U         GET PRINT MODE
12300          LEAX    FMTAB,PCR       GET PRINT MODE OFFSET TABLE
12400          LSLB                    MAKE PRINT MODE *2 FOR 16 OFFSET
12500          LDD     B,X             GET PRINT MODE'S 16 BIT OFFSET
12600          LEAX    D,X             MAKE X POINT THE SUBR.
12700          LDA     ,S              GET CHARACTER TO PRINT
12800          JSR     ,X              CALL IT
12900          LDB     XCPOS,U         GET PRINT CURSOR X POSITION
13000          ORB     #$E0            MAKE IT BOTTOM LINE
13100          LDA     #5
13200          STD     $88             PUT IT IN BASIC'S CURSOR POSITION
13300          PULS    PC,U,Y,X,D      RESTORE REGS AND EXIT
13400
13500 *SET TEXT CURTSOR Y POSITION
13600 FMODE2   CMPA    WDYLEN,U        IS Y POSITION BIGGER THEN THE
13700          BLO     FMODEA          WINDOW? NO, USE IT!
13800          LDA     WDYLEN,U        ELSE USE THE MAX POSITION
13900          DECA
14000 FMODEA   STA     YCPOS,U         STORE IT
14100          CLR     FMODE,U         RESET FMODE TO NORMAL
14200          RTS                     AND EXIT
14300
14400 *GET X POSITION
14500 FMODE1   CMPA    WDXLEN,U        IS THE X POSITION TOO BIG?
14600          BLO     FMODEB          NO, USE IT!
14700          LDA     WDXLEN,U        ELSE USE THE MAX X POSITION
14800          DECA
14900 FMODEB   STA     XCPOS,U         STORE IT
15000          INC     FMODE,U         MOVE TO NEXT PRINT MODE
15100          RTS                     AND EXIT
15200
15300 CHRTAB   FDB     NUL-CHRTAB      CONTROL COD. VECTORS
15400          FDB     SETPOS-CHRTAB
15500          FDB     SVIDEO-CHRTAB
15600          FDB     IVIDEO-CHRTAB
15700          FDB     SETWIN-CHRTAB
15800          FDB     LEFT-CHRTAB
15900          FDB     RIGHT-CHRTAB
16000          FDB     PING-CHRTAB
16100          FDB     BACKSP-CHRTAB
16200          FDB     TAB-CHRTAB
16300          FDB     LF-CHRTAB
16400          FDB     UPLINE-CHRTAB
16500          FDB     CLS-CHRTAB
16600          FDB     CR-CHRTAB
16700
16800 CHR1     CMPA    #13             IS THE CHARACTER A CONTROL CODE?
16900          BHI     CHR20           SKIP IF NOT
17000          LSLA                    ELSE JUMP TO IT VECTOR
17100          LEAX    CHRTAB,PCR
17200          LDD     A,X
17300          JMP     D,X
17400
17500 CHR20    SUBA    #32             IS IT A PRINTABLE CODE?
17600          CMPA    #96
17700          BHI     CHR29           NO, THEN SKIP TO EXIT
17800
17900          LBSR    HOLDIT          PUT MOUSE CURSOR ON HOLD
18000          LDB     XCPOS,U         GET X-POSITION ON SCREEN
18100          CMPB    WDXLEN,U        PAST RIGHT SIDE?
18200          BLO     CHR21           NO, JUST PRINT IT
18300          CLR     XCPOS,U         ELSE BACK TO LEFT SIDE
18400          INC     YCPOS,U         AND MOVE DOWN LINE
18500          BSR     SCROLL          IF NEEDS TO SCROLL, THEN SCROLL!
18600
18700 CHR21    LEAY    >CHRDAT,PCR     GET CHR DATA TABLE
18800          LDB     #8              INDEX TO THE CHAR. DATA
18900          MUL
19000          LEAY    D,Y
19100
19200          LDD     WDYPOS,U        GET WINDOW START POSITION
19300          ADDD    YCPOS,U         GET TEXT CURSOR POSITION
19400          ADDA    SSTART          ADD SCREEN START IN MEMORY
19500          TFR     D,X             AND POINT X TO IT
19600          LDB     #8              NUMBER BYTES PER CHAR.
19700 CHR22    LDA     ,Y+             GET DATA BYTE
19800          EORA    INVERT,U        INVERT IF FLAG IS ON
19900          STA     ,X              PUT IT ON THE SCREEN
20000          LEAX    32,X            NEXT LINE ON SCREEN
20100          DECB                    ALL BYTES DONE???
20200          BNE     CHR22           LOOP IF NOT
20300
20400          INC     XCPOS,U         NEXT POSITION ON SCREEN
20500 CHR29    RTS
```

continued from Page 25

```
ANER HAS A DISPOS-    ABLE FILTER
 ELEMENT THAT SHOULD BE CHANGED
WHEN IT BECOMES       BLOCKED WIT
H DIRT. IF AIR CAN    NOT GET TO
THE CARBURETOR WE     WILL FAIL T
O GET IGNITION.":GOTO190
174 CLS:PRINT@4,"** OIL PRESSURE
 LOW **":PRINT@65,"THE RED WARNI
NG LIGHT WILL COME ON WHEN OIL P
RESSURE IS LOW.      THIS LIGHT WI
LL COME ON MOMEN-    TARILY WHEN S
TARTING ENGINE.":GOTO190
176 'BATTERY DEAD
178 GOSUB98:GOSUB108:IF INKEY$="
C"THEN122ELSE178
180 'AIR INTAKE BLOCKED
182 IF INKEY$<>"S"THEN182 ELSEGO
SUB90:GOSUB94:PAINT(80,144),1,3
184 'NO GAS
186 GOSUB98:GOSUB108:FORY=1TO3:P
LAY"T30ABCDEEDCBA":GOSUB28:NEXTY
:IF INKEY$="C"THEN122 ELSE186
188 'CORRECT ANSWER OR NOT
190 IF A=D*2 THENPRINT@422,"* CO
RRECT ANSWER *":R=R+1:D=0:SOUND1
40,11:SOUND180,11:SOUND176,4:SOU
ND165,4:SOUND154,4:SOUND200,12:S
OUND180,17:GOSUB46:IF R=5 THEN20
8 ELSE50
192 PRINT@425,"WRONG ANSWER":SOU
ND1,4:W=W+1:GOSUB46
194 SCREEN1,0:ON D GOTO186,178,8
,8,186
196 CLS0:PRINTSTRING$(160,191);S
TRING$(160,207);STRING$(160,175)
;:PRINT@71,"4 CYLINDER ENGINE";:
PRINT@230,"EDUCATIONAL PROGRAM";
:PRINT@391,"MICHAEL B KROMEKE";
198 A=A+1
200 M=RND(5):FORB=1TO5
202 IF M=M(B)THEN200
204 NEXTB:M(A)=M:IF A<5 THEN198
206 GOSUB218:RETURN
208 D=5/(5+W):CLS:PRINT@72,"NUMB
ER CORRECT 5":PRINT@136,"NUMBER
WRONG"W:PRINT@200,"SCORE ="INT(1
004D)"%":GOSUB218:IF D<.5 THEN P
RINT@264,"BEGINNER"
210 IF D>.49 AND D<1 THENPRINT@2
62,"SHADE TREE MECHANIC"
212 IF D=1 THENPRINT@264,"MASTER
 MECHANIC"
214 GOSUB218:PRINT@328,"PLAY AGA
IN Y/N";:INPUTZ$:IFZ$="N"THEN EN
D ELSE RUN
216 'JOPLIN'S ENTERTAINER
218 PLAY"V30T202L8DD+EL4O3CO2L8E
O3L4CO2L8EO3CL2CL8CCDD+ECDEEO2BO
3DL2.CL4":RETURN
220 CLS4:PRINT@192," OIL LEAKED
OUT OF ENGINE AND IT FROZE UP. Y
OU LOST YOUR ENGINE  AND THE GAM
E. SORRY !!!"
```

continued from Page 26

CoCo 3 has, under OS-9 Level II, a user interface much like the Macintosh but far faster and in full color. For the CoCo 3, OS-9 has become virtually invisible to the user who wants it that way - in other words, you can deal with the appications and programs you want, and pretty much ignore the operating system unless you want to use it.

In short, CoCo 3 is a new generation of home and small business computer - fully as revolutionary as the original Color Computer was five years ago. Its great advantage is that it already has one of the largest bases of software (and hardware for that matter) of any machine on the market, plus a raft of OS-9 Level II software which has been around for some time.

Even at $600 (US), CoCo 3 would be the best single home and business buy in the world of computers. At $219.95 (US), it is not only a stupendous machine, but an immense bargain as well. With the Color Computer 3, Tandy has moved to the cutting edge of technology and made the technology affordable for virtually everyone.

Listing 2:

```
00100 *MOUSE5/ASM     PART 2 OF SCREEN DRIVER
00200
00300 *THIS ONE DOES A RETURN ON TEXT SCREEN
00400 CR      LDB     WDXLEN,U        GET LINE LENGTH
00500         SUBB    XCPOS,U         GET BYTE LEFT ON LINE
00600 CR1     DECB                    ANY BYTES LEFT ON LINE?
00700         BMI     CR2             SKIP IF NOT
00800         PSHS    B               SAVE COUNTER
00900         LDA     #32             GET SPACE " "
01000         LBSR    CHR20           PRINT IT ON SCREEN
01100         PULS    B               GET COUNTER
01200         BRA     CR1             LOOP TILL END OF LINE
01300
01400 CR2     CLR     XCPOS,U         PUT TEXT CURSOR AT START OF LINE
01500
01600 *DO A LINE FEED
01700 LF      INC     YCPOS,U         MOVE DOWN ONE LINE
01800 SCROLL  PSHS    A               SAVE PRINTING CHAR.
01900         LDB     YCPOS,U         GET LINE POSITION
02000         CMPB    WDYLEN,U        IS THIS LINE OFF SCREEN?
02100         BLO     SCROLL9         NO, SKIP AND EXIT
02200         DEC     YCPOS,U         ELSE MOVE UP A LINE
02300
02400         BSR     HOLDIT          TURN OFF CURSOR FOR PRINTING
02500         LDA     #15
02600         STA     HOLD,U          MAKE THE HOLD A LITTLE LONGER
02700         LDA     WDYLEN,U        GET NUMBER LINE IN WINDOW
02800         DECA                    DON'T SCROLL THE LAST LINE
02900         BEQ     SCROLL5         SKIP ONLY ONE LINE
03000
03100         LDA     WDYPOS,U        GET TEXT WINDOW LINE POSITION
03200         ADDA    SSTART          ADD THE SCREEN START POSITION
03300         LDB     WDXPOS,U        GET WINDOW C POSITION
03400         TFR     D,X             AND MAKE X POINT TO IT
03500         LDA     WDYLEN,U        GET NUMBER OF LINES IN WINDOW
03600         DECA                    DON'T SCROLL THE LAST ONE
03700         LSLA                    MUL BY 8 SCAN LINES
03800         LSLA
03900         LSLA
04000 SCROLL1 PSHS    X,A             SAVE SCAN LINE COUNT & POINTER
04100         LDB     WDXLEN,U        GET WINDOW X LENGTH
04200         LSRB                    DIVIDE BY 2
04300         BCC     SCROLL2         SKIP IF EVEN
04400         LDA     256,X           MOVE ONE BYTE IF ODD
04500         STA     ,X+
04600 SCROLL2 STB     COUNT,U         SET BYTE COUNTER
04700         BEQ     SCROLL4         SKIP IF ONLY ONE BYTE TO MOVE
04800 SCROLL3 LDD     256,X           ELSE MOVE TWO BYTES AT ONCE
04900         STD     ,X++
05000         DEC     COUNT,U         LOOP TILL ALL BYTES ARE DONE
05100         BNE     SCROLL3
05200 SCROLL4 PULS    X,A             RESET BACK TO START OF LINE
05300         LEAX    32,X            MOVE TO NEXT SCAN LINE
05400         DECA                    ALL LINE MOVED?
05500         BNE     SCROLL1         LOOP IF NOT
05600
05700 SCROLL5 LDB     #8              LINES TO BE BLANKED
05800         LDA     INVERT,U        GET BACKROUND (BLACK OR WHITE)
05900 SCROLL6 PSHS    X,B             SAVE LINE START AND COUNTER
06000         LDB     WDXLEN,U        GET NUMBER OF BYTES PER LINE
06100 SCROLL7 STA     ,X+             PUT DOWN BACKROUND
06200         DECB                    ALL DONE WITH THIS LINE?
06300         BNE     SCROLL7         NO, THEN LOOP
06400         PULS    X,B             GET COUNT & START OF LINE
06500         LEAX    32,X            MOVE TO NEXT LINE
06600         DECB                    ALL LINES DONE
06700         BNE     SCROLL6         LOOP IF NOT
06800         LDA     #2              RESET HOLD
06900         STA     HOLD,U
07000 SCROLL9 PULS    PC,A            AND EXIT
07100
07200 HOLDIT  PSHS    D,Y,U,X         SAVE ALL REGS
07300         LDA     #2              DELAY CURSOR FOR 2 IRQS
07400         STA     HOLD,U
07500         LBSR    CUROFF          TURN OFF CURSOR
07600         PULS    PC,U,Y,X,D      RESTORE AND EXIT
07700
07800 CLS     BSR     HOLDIT          PUT CURSOR ON HOLD
07900         LDA     #10             PIT IT ON HOLD A LITTLE LONGER
08000         STA     HOLD,U
08100         LDD     WDYPOS,U        GET WINDOW POSITION
08200         ADDA    SSTART          ADD START OF SCREEN MEMORY
08300         TFR     D,X             MAKE X THE POINTER TO SCREEN
08400         LDA     INVERT,U        GET BACKROUND COLOR
08500         LDB     WDYLEN,U        GET NUMBER OF LINES
08600         LSLB                    8 LINES PER TEXT LINE
08700         LSLB
08800         LSLB
08900 CLS1    PSHS    B,X             SAVE LINE COUNT & POINTER
09000         LDB     WDXLEN,U        GET NUMBER OF BYTES PER LINE
09100 CLS2    STA     ,X+             PUT BACKROUND ON SCREEN
09200         DECB                    ARE ALL BYTES DONE ON LINE
09300         BNE     CLS2            NO, THEN LOOP BACK
09400         PULS    X,B             GET LINE COUNTER & START OF LINE
09500         LEAX    32,X            MOVE TO NEXT SCAN LINE
09600         DECB                    ALL SCANS DONE?
09700         BNE     CLS1            NO, THEN LOOP
09800         CLR     YCPOS,U         PUT TEXT CURSOR AT
09900         CLR     XCPOS,U         TOP OF WINDOW
10000         LDA     #2              RESET HOLD TO STANDARD
10100         STA     HOLD,U
10200         RTS                     AND EXIT
10300
10400 SVIDEO  CLR     INVERT,U        MAKE WHITE ON BLACK
10500 NUL     RTS
10600
10700 SETPOS  LDA     #1              START GET CURSOR POSITION
10800         STA     PMODE,U
10900         RTS
11000
11100 SETWIN  LDA     #3              START GET WINDOW SIZE & POSITION
11200         STA     PMODE,U
11300         RTS
11400
11500 WMODE1  CMPA    #31             IS WINDOW X START POS <31
11600         BHS     WERROR          IF NOT THAT ERROR
11700         STA     WDXPOS,U
11800         NEGA                    GET MAX WINDOW X LEG FOR THAT
11900         ADDA    #32             START PSOITION
12000         STA     WDXLEN,U        AND USE IT FOR NOW
12100         CLR     YCPOS,U         HOME TEXT CURSOR TO TOP
12200         CLR     XCPOS,U         LEFT OF WINDOW
12300         INC     PMODE,U         MOVE TO NEXT PRINT MODE
12400         RTS
12500
12600 WMODE2  CMPA    #23             IS WINDOW Y START POS <23
12700         BHS     WERROR          IF NOT THEN ERROR
12800         STA     WDYPOS,U
12900         NEGA                    GET MAX WINDOW X LEN FOR THAT
13000         ADDA    #24             STARTING POSITION
13100         STA     WDYLEN,U
13200         INC     PMODE,U         MOVE TO NEXT PRINT MODE
13300         RTS
13400
13500 WERROR  CLR     PMODE,U         RESET PRINT MODE
13600         LBRA    FNERR           AND GIVE A ?FN ERROR
13700
13800 WMODE3  BEQ     WERROR          ERROR IF 0 BYTES PER LINE
13900         CMPA    WDXLEN,U        IS THIS WINDOW TOO BIG?
14000         BHI     WERROR          ?FN ERROR IF SO
14100         STA     WDXLEN,U        ELSE SET LINE LENGTH
14200         INC     PMODE,U         MOVE TO NE XT PRINT MODE
14300         RTS
14400
14500 WMODE4  BEQ     WERROR          ERROR IF LENGTH IS ZERO
14600         CMPA    WDYLEN,U        IS THE SIZE TOO BIG?
14700         BHI     WERROR          FN ERROR IF SO
14800         STA     WDYLEN,U        ELSE SET IT
14900         CLR     PMODE,U         RESET PRINT MODE
15000         RTS                     AND EXIT
15100
15200 IVIDEO  LDA     #255            SET BLACK ON WHITE VIDEO
15300         STA     INVERT,U
15400         RTS
15500
15600 *THIS IS A BACK SPACE CODE ($08)
15700 BACKSP  LDD     YCPOS,U         GET TEXT CURSOR POSITION
15800         DECB                    BACK UP ONE
15900         BPL     BACKS1          USE IT IF STILL ON THE SAME LINE
16000         LDB     WDXLEN,U        ELSE MOVE CURSOR TO END
16100         DECB                    OF LINE
16200         DECA                    AND MOVE UP ONE LINE
16300         BPL     BACKS1          USE IT IF NOT THE TOP OF WINDOW
16400         INCA                    ELSE BACK TOP TOP LINE
16500         CLRB                    AND START OF THE LINE
16600 BACKS1  STD     YCPOS,U         STORE NEW TEXT CURSOR POSITION
16700         LDA     #32             CLEAR (BLANK) THE NEW TEXT
16800         LBSR    CHR20           CURSOR POSITION
16900         DEC     XCPOS,U         MOVE BACK TO THAT PLACE
17000         RTS                     AND EXIT
17100
17200 LEFT    LDA     XCPOS,U         GET TEXT CURSOR'S X POSITION
17300         DECA                    MOVE BACK ONE
17400         BPL     LEFT1           IF STILL ONE LINE? THEN USE IT
17500         LDA     WDXLEN,U        ELSE MAKE IT THE MAX RIGHT
17600         DECA                    POSITION ON LINE
17700 LEFT1   STA     XCPOS,U         AND STORE IT
17800         RTS                     ALL DONE AND EXIT
17900
18000 RIGHT   LDA     XCPOS,U         GET TEXT CURSOR'S X POSITION
18100         INCA                    MOVE TO RIGHT
18200         CMPA    WDXLEN,U        GONE TOO FAR?
18300         BLO     RIGHT1          NO, THEN USE IT (SKIP)
18400         CLRA                    ELSE MOVE TO FAR LEFT
18500 RIGHT1  STA     XCPOS,U         AND STORE IT
18600         RTS                     ALL DOWN, EXIT
18700
18800 UPLINE  DEC     YCPOS,U         MOVE TEXT CURSOR UP ONE LINE
18900         BPL     UPLIN1          MOVE OFF OF TOP OF WINDOW?, NO SKIP
19000         LDA     WDYLEN,U        SET TO BOTTOM OF WINDOW
19100         DECA
19200         STA     YCPOS,U
19300 UPLIN1  RTS
19400
19500 TAB     LDB     XCPOS,U         GET TEXT CURSOR POSITION
19600         ANDB    #7              GET OFFSET FROM TAB STOP (EVERY 8)
19700         NEGB                    GET NUMBER TO NEXT NEXT ONE
19800         ADDB    #8
19900         LDA     #32             GET A SPACE " "
20000 TAB1    PSHS    D               SAVE BOTH
20100         LBSR    CHR20           PRINT THAT SPACE
20200         PULS    D               GET BOTH
20300         DECB                    AT NEXT TAP STOP?
20400         BNE     TAB1            NO, THEN LOOP
20500         RTS
```

Let's Take a Look at the CoCo 2 B

By Tony DiStefano
Rainbow Contributing Editor

This week I had the honor of repairing an old 'D' board Color Computer belonging to "KISSable OS-9" author Dale L. Puckett. Although there are a lot of old CoCos still out there, you can't get any more of them. Today, Radio Shack is peddling a CoCo with the letter 'B' in the catalog number. I don't know what the 'B' stands for, but there are a few changes inside. I bought one at the Palo Alto RAINBOWfest. What I want to do here is explain some of the changes Radio Shack has made.

The first thing I noticed when I opened the box is that it says Tandy on the computer and not Radio Shack. It also says Color Computer and not CoCo 2. This is the smallest PCB (Printed Circuit Board) I have seen for a CoCo. Small is good in many ways. First, it costs less to produce. It also has the least parts count of all the CoCos ever made. Not only is this good for production costs, it's also good for users. The lower the parts count in a computer, the less likely a breakdown. Then there is the question of heat; all electronic parts, whether digital or analog, dissipate heat. How many times have you heard that the computer crashes when it is too hot? Fewer parts mean less heat.

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

This computer does not have any regulated 12 volts, the same as the other CoCo 2s. There is no negative voltage available except on the SALT chip, which buffers and converts the RS-232 signals. In theory, RS-232 specifies that the signal be +/-12 volts. This new CoCo 2 (and all other CoCo 2s) have only +/-5 volts. While this will work with most RS-232 devices, check the specifications to be sure. Again, as with the other CoCo 2s, there are about 12 volts unregulated at the power diodes, which can be used for devices needing the voltage. The diodes are numbered D10 and D11. Remember, the side with the white band is the positive side.

The next interesting part in this CoCo 2 B is a PIA (Peripheral Interface Adapter). The first CoCos had two PIAs of the same kind. They were both MC6821s by Motorola. The next stage of the CoCo had one MC6821 and one MC6822. This 6822 is called an IIA (Industrial Interface Adapter). There is just a small difference between the two. Now the second PIA in the CoCo 2 B is no longer an MC6822, but an SC67331P. It is a Motorola part, and compatible with the MC6822. The difference is in the impedance matching between the keyboard and the PIA — custom made for Tandy, no doubt. If you happen to destroy this part, a regular MC6822 will work. The keyboard matrix is the same.

As with the CoCo 2 A, there are six jumpers, J1 to J6. One of the jumpers is used to detect the presence of 64K memory RAM. The other five jumpers are labeled 64K/128K. A lot of people think that this means you can have 128K of RAM. This is not true. Look again; there is only one place for ROM. Before, there were two sockets, one for the BASIC ROM and the other for the Extended BASIC ROM, each ROM being 8K long. A ROM's capacity is usually expressed in bits. In the CoCo, the data bus is eight bits wide. Therefore an 8K ROM has 8K times eight bits, giving you 64K bits. Starting to get the picture? Since there is only one place on the PCB for BASIC and Extended BASIC, a new chip with both 8K ROMs (or 64K bits) gives you 16K or, like the label says, 128K.

If you bought the computer without Extended BASIC, you got a socket and an 8K ROM in a 28-pin package. The jumpers are set to the 64K position. If you bought an Extended BASIC machine, you got a soldered-in 16K ROM with the jumpers set to the 128K side. In both cases you got a new version of BASIC, Version 1.3. If you have Extended BASIC, then you only see the

CN3		CN4	
Pin	Function	Pin	Function
1	GND	1	GND
2	+5V	2	A7
3	A4	3	A3
4	A5	4	A2
5	A6	5	A1
6	RAS	6	A0
7	$\overline{WE}$	7	DQ6
8	DQ1	8	DQ5
9	DQ0	9	DQ7
10	DQ3	10	DQ4
11	DQ2	11	CAS
12	GND	12	GND

Figure 1

Extended BASIC Version, 1.1. To see the BASIC version type in EXEC 41175.

To take this further, the two ROMs Tandy uses, 8K and 16K, are pin-for-pin compatible with two EPROM counterparts. The 28-pin BASIC ROM is compatible with the Intel 2764 EPROM. The 28-pin Extended BASIC ROM is compatible with the Intel 27128 EPROM. Now you can see where the 64K/128K numbers come from. If you have an EPROM programmer, modify these ROMs to suit yourself and plug them right in. Of course, if the ROM is soldered in, you will have to desolder it and put in a socket. Don't forget to change the jumpers to the right place. More on this later.

The RAM portion is quite impressive. There are three ways to add 64K to this CoCo 2 B. If you have 16K of RAM on the computer, chances are the chips Tandy used are two 4416 RAM chips. These chips are 16K by four bits each. Since the CoCo needs eight bits, there are only two of these chips. These chips are in the two 18-pin sockets between the two white connectors. The first way to upgrade this 16K computer is to change these two chips for the 64K counter part. The number to this is 4464. There are a lot of different numbers that are compatible with this chip. Just ask for a 4464, a 64K by four DRAM or an equivalent.

With the computer turned off, remove the two memory chips and install the two new ones. On the left side, there is a white box marked J6, jumper 6. You must solder a jumper across the two pins inside this box. This tells the software that there are 64K memory chips installed. That's all there is to it.

The next way to upgrade is using the two white connectors. These connectors consist of all the lines necessary to connect 64K of memory. A small PCB will be necessary. The pinouts to the connectors are in Figure 1.

There are two reasons why I'm not going into details on how to construct this piggyback board. The first is that it is available, fully assembled and tested, from CRC Inc. (514) 383-5293 for a modest price, and the other reason is that there is a third method of upgrading this CoCo 2 B.

If you have some 64K chips lying around gathering dust, you'll like the third way to upgrade. See all those holes filled with solder? Do you see the eight empty IC names soldermasked on the PCB? These eight blank areas are made for 64K memory chips. The regular run of the mill 4164s. All you have to do is add eight sockets and plug them right in. There is a small catch: The holes for these ICs are filled with solder. You must first empty the holes of their solder. You can use a device such as Radio Shack's desoldering pump (less than $20). Just heat up the hole to be cleaned with a hot soldering iron. Then bring the desoldering pump to the hole. Remove the iron, press the pump to the hole and press the pump button. Go through all the holes of each pin. It would be wise to solder in sockets, not the chips directly. On some boards, the eight decoupling capacitors are also missing. Insert eight .1 UF capacitors. As with any upgrade to 64K, don't forget to jumper the connections at J6. That's all there is to it.

There are a few more changes in the CoCo 2 B. Until now, all CoCos used the Motorola MC6847 as a display processor. This is the chip that gives the text on the screen and all of the graphics modes. Text on the screen has been green with black letters. When typing in lowercase letters, they would appear as inversed blocks of black with green letters. The new chip that Tandy uses on this CoCo 2 B is slightly different. It is an MC6847T1. (This chip might also have the part #XC80652P.) This chip is different. It has built-in real lowercase characters and you can also get rid of that border in certain cases. This is a real nifty improvement to the CoCo's display. The only problem with this is that Extended BASIC will not let you use these added features. Next month, I'll get my soldering iron out and add a few switches to change the default values.

The last change the good people at Tandy made was in the SAM (Synchronous Address Multiplexer). With all these changes to memory, video and circuitry, a new SAM chip is needed. It is the SN74LS785. A Motorola part that is upward compatible with the old SN74LS783 or the MC6883.

Back to the 28-pin ROM. Earlier, I mentioned that the ROM Tandy used is pin-for-pin compatible with an EPROM. A long time ago, a reader asked if there was a way to add a DOS chip inside the CoCo. Now there is. There are many ways to do this. Different people like to solder things together in different ways. I like the fastest and easiest way. Some people like to make it neat. The chip you must use is either a 2764 or a 27128. All of the address lines, data lines and power lines are the same. The only line that is different will be the chip select line. We'll get that line from another chip. The chip enable line on an EPROM is pins 20 and 22. These are the pins that must connect to the extra enable. The thing to do is connect all the pins except the two enable pins. Here is where some people differ. I used a 28-pin socket and soldered all the pins (except 20 and 22) to the 28-pin ROM. I bent pins 20 and 22 up and soldered them together, running a wire to Pin 12 of the 74LS138. That's the easy way.

Now, plug in the new EPROM and the cartridge area socket will be inside the CoCo. Some people don't like to solder directly to a ROM. Use a wire wrap socket and solder a second socket to the legs about halfway down. Cut pins 20 and 22 from the top socket. Solder these two pins to the 74LS138 mentioned above. Plug the ROM into the lower socket and the EPROM into the upper socket. The same results happen, but it is neater. No soldered ROM, but it is also a little more trouble. Take your pick.

Graph-O-Matic

By Jeff Harper

A fast way to plot 3-D functions using the CoCo

My program, *3DFNCPLT*, is a three-dimensional function plot which quickly graphs any function in a 3-D Cartesian coordinate system.

To achieve 3-D graphing, the program plots the function over the range of 'Y' coordinates (from smallest to largest) for each 'X' coordinate. It then graphs the function over the range of 'X' coordinates (again from smallest to largest) over the range of 'Y' coordinates. This produces a stunning graph of the function which resembles a net having been draped over a solid of the function.

The program first asks if you want to have the 'X', 'Y', 'Z' coordinate axes drawn on the screen. Then you are prompted to enter the function to be plotted. The computer displays Z= at which point you should enter a function. A typical response might be COS(Y)+SIN(X).

The computer then requests the smallest and largest 'X' and 'Y' coordinate values. These values can have any range dependent only upon the function to be plotted. I usually get a good idea of what a particular function looks like by using a range of values from -3 to 3 for both the 'X' and 'Y' coordinates. Attempts to plot off of the screen are cropped so as not to result in an error. 'X' coordinate values are graphed coming toward you from the back of the screen, 'Y' coordinate values are graphed horizontally and 'Z' coordinate values are graphed vertically.

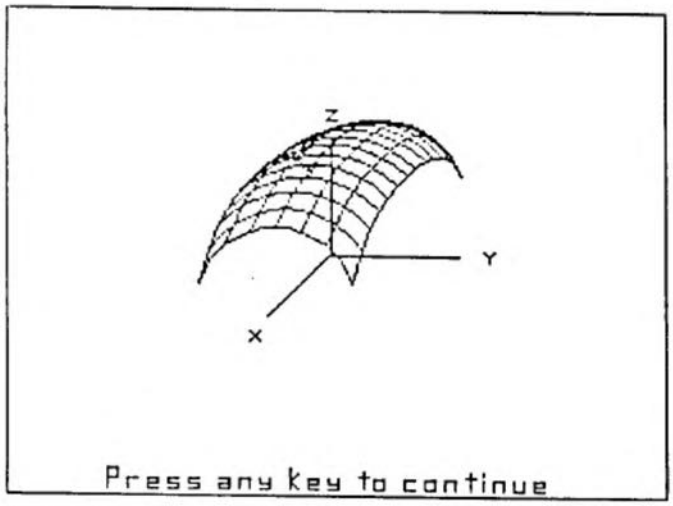
A unique and particularly useful feature of this program is that it lets the user enter the function to be plotted while the program is running. The user does not need to stop program execution, change a program line and then rerun the program. The transformation of your function into a line of BASIC code is simple but interesting.

Once the function to be plotted is entered, the routine beginning in Line 970 looks for the location of Line 410 in memory. Once located, the string containing your function is tokenized into BASIC program Line 410. Note that this is done before the program ever reaches Line 410. Once the tokenizing is complete, the program continues and eventually executes Line 410 in the subroutine beginning at Line 360. This technique can easily be applied in other programs of your own creation.

While *3DFNCPLT* requires a 16K Extended CoCo, the routine that transforms the function into a BASIC program line will work on any size or system configuration including Color BASIC.

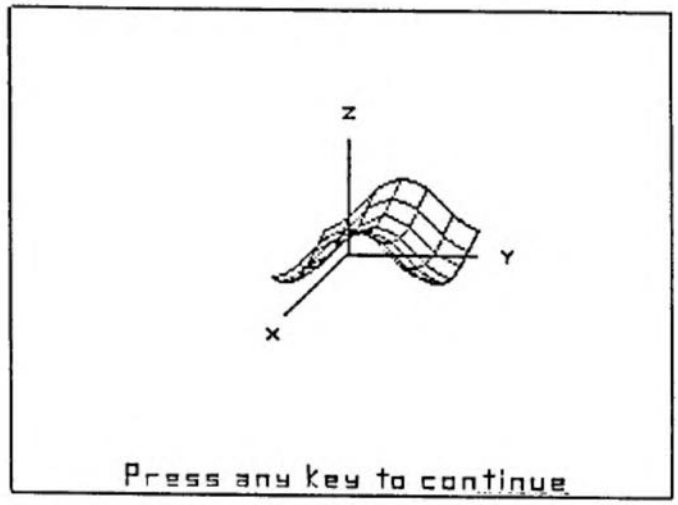
□

Jeff Harper is a high school math teacher who enjoys programming the CoCo as a hobby. He is also a computer programming instructor for enrichment classes held each summer.



$z=\sqrt{19-x^2-y^2}$

x=[-3,3], y=[-3,3]



z=cosy+sinx

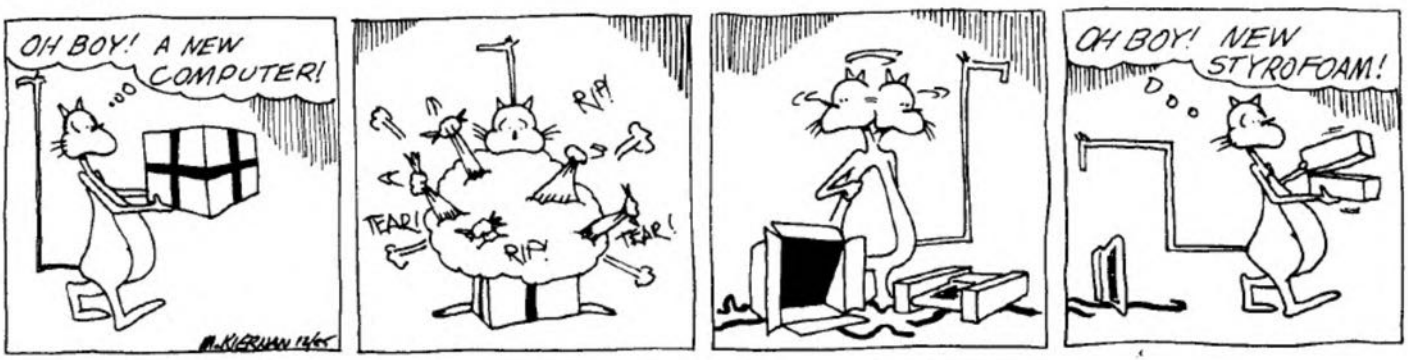
x=[-3,3], y=[-3,3]

Editor's Note: In Line 410 of the listing below, replace the REM marks (') with spaces. This line is used for entry of the function you wish to plot and the spaces are required in order to reserve space for your function.

The listing: 3DFNCPLT

```
1Ø '3 DIMENSIONAL FUNCTION PLOT
2Ø 'WITH INTERACTIVE FUNCTION
3Ø 'DESIGNATION
4Ø '
5Ø '            BY
6Ø '
7Ø '       JEFF  HARPER
8Ø '         Ø2/15/85
9Ø '
1ØØ PMODE 4,1
11Ø COLOR Ø,1:PCLS1
12Ø LINE(Ø,Ø)-(255,191),PSET,B
13Ø '*************************
14Ø '* STEP VALUES FOR DETAIL *
15Ø '* AND STRIPE SPACING     *
16Ø '*************************
17Ø S1=.5:S2=.5:S3=1
18Ø '
19Ø CLS
2ØØ PRINT"AXES DRAWN ON SCREEN (
Y/N)?";
21Ø A$=INKEY$:IF A$="" THEN 21Ø
ELSE IF A$<>"N" AND A$<>"Y" THEN
 21Ø ELSE PRINT A$:PRINT
22Ø '
23Ø '*************************
24Ø '*FIND LINE 41Ø IN MEMORY, *
25Ø '*GET EQUATION & TRANSFORM *
26Ø '*IT INTO BASIC CODE AT    *
27Ø '*LINE 41Ø BEFORE WE GET   *
28Ø '*THERE IN THE PROGRAM.    *
29Ø '*************************
3ØØ GOSUB 127Ø
31Ø GOTO54Ø
32Ø '
33Ø '***********************
34Ø '*RADIAN TRANSFORMATION*
35Ø '***********************
36Ø RADIAN=(X*-.7Ø71Ø68)
37Ø '
38Ø '*************************
39Ø '*FUNCTION TO PLOT IN 41Ø*
4ØØ '*************************
41Ø Z=X^2+Y^2'''''''''''''''''''''
'''''''''''''''''''''''''''''''''
'''''''''''''''''''''''
42Ø '
43Ø '***********************
44Ø '*SCALE IMAGE TO SCREEN*
45Ø '***********************
46Ø NX=1Ø*(Y+RADIAN)+128
47Ø NY=192-(1Ø*(Z+RADIAN)+96)
48Ø IF NX<Ø THEN NX=Ø
49Ø IF NX>255 THEN NX=255
5ØØ IF NY<Ø THEN NY=Ø
51Ø IF NY>191 THEN NY=191
52Ø RETURN
53Ø '
54Ø PRINT:INPUT "SMALLEST X VALU
E";BX
55Ø INPUT"LARGEST  X VALUE";EX
56Ø INPUT"SMALLEST Y VALUE";BY
57Ø INPUT"LARGEST  Y VALUE";EY
58Ø SCREEN 1,1
59Ø IF A$="N" THEN 72Ø
6ØØ '
61Ø '***********************
62Ø '*DRAW X, Y, AND Z AXIS*
63Ø '***********************
64Ø LINE(128,51)-(128,96),PSET:L
INE-(178,96),PSET:LINE(127,96)-(
1Ø3,12Ø),PSET
65Ø DRAW"BM19Ø,98 U2NH2E2"
66Ø DRAW"BM96,13Ø E2NH2NE2F2"
67Ø DRAW "BM126,42 NR4E4L4"
68Ø '
69Ø '***********************
7ØØ '*PLOT FUNCTION VALUES*
71Ø '***********************
72Ø FOR X=BX TO EX STEP S1
73Ø FOR Y=BY TO EY STEP S2
74Ø GOSUB 36Ø
75Ø IF Y=BY THEN LINE(NX,NY)-(NX
,NY),PSET:GOTO77Ø
76Ø LINE-(NX,NY),PSET
77Ø NEXT Y
78Ø NEXT X
79Ø '
8ØØ '**********************
81Ø '*PLOT CROSS STRIPES*
82Ø '**********************
83Ø X=BX
84Ø FOR Y=BY TO EY STEP S3
85Ø GOSUB 36Ø
86Ø X2=NX:Y2=NY
87Ø X=X+.5
88Ø IF X>EX THEN X=BX:GOTO 91Ø
89Ø GOSUB 36Ø
9ØØ LINE(NX,NY)-(X2,Y2),PSET:GOT
O 86Ø
91Ø NEXT Y
92Ø '
93Ø DRAW"BM4Ø,189 U8R4D4NL4BR4"
94Ø DRAW"ND4R4BR4"
95Ø DRAW"D4R4BU2NL4U2NL4BR4"
96Ø DRAW"NR4D2R4D2NL4BR4"
97Ø DRAW"R4U2L4U2R4BR8"
98Ø DRAW"R4D2L4D2R4U4BR4"
99Ø DRAW"ND4R4D4BR4"
1ØØØ DRAW"R4U2L4U2BR4ND2BR8"
1Ø1Ø DRAW"NU4D1ND3NE3NF3E1BR6"
1Ø2Ø DRAW"D4R4BU2NL4U2NL4BR4BD4"
1Ø3Ø DRAW"R4U2L4U2BR4ND2BR8"
1Ø4Ø DRAW"BU1R2NR2NU2D5BR4"
1Ø5Ø DRAW"U4R4D4NL4BR8"
1Ø6Ø DRAW"NR4U4R4BR4"
1Ø7Ø DRAW"ND4R4D4NL4BR4"
1Ø8Ø DRAW"U4R4D4BR4"
1Ø9Ø DRAW"BU5R2NR2NU2D5BR5"
11ØØ DRAW"U4BU2U1BD7BR4"
111Ø DRAW"U4R4D4BR4"
112Ø DRAW"NU4R4NU4BR4"
113Ø DRAW"NR4U2NR4U2R4D2"
114Ø A$=INKEY$:A$=""
115Ø A$=INKEY$:IF A$="" THEN 115
Ø ELSE 1ØØ
116Ø '
117Ø '*************************
118Ø '* A BASIC SELF-PROGRAMMER*
119Ø '* AND INTERPRETER        *
12ØØ '* BY JEFF HARPER        *
121Ø '* Ø8/Ø2/84              *
122Ø '*                       *
123Ø '*ADAPTED FOR THE ECB COCO*
124Ø '*FROM AN ARTICLE IN US-8Ø*
125Ø '* MAY/JUN 1981 PG.9Ø    *
126Ø '*************************
127Ø MM=PEEK(25)*256+PEEK(26)
128Ø LN=41Ø
129Ø MS=INT(LN/256):LS=LN-256*MS
13ØØ M=MM
131Ø IF PEEK(M+3)<>LS OR PEEK(M+
2)<>MS THEN M=PEEK(M)*256+PEEK(M
+1):GOTO131Ø
132Ø ML=M+4
133Ø PRINT"ENTER FUNCTION TO PLO
T"
134Ø LINE INPUT"Z=";E$
135Ø E$="Z="+E$
136Ø M=ML
137Ø FOR N=1 TO LEN(E$)
138Ø Q=ASC(MID$(E$,N,1))
139Ø IF Q=43 THEN Q=171:GOTO151Ø
 '+
14ØØ IF Q=45 THEN Q=172:GOTO151Ø
 '-
141Ø IF Q=42 THEN Q=173:GOTO151Ø
 '*
142Ø IF Q=47 THEN Q=174:GOTO151Ø
 '/
143Ø IF Q=61 THEN Q=179:GOTO151Ø
 '=
144Ø IF Q=94 THEN Q=175:GOTO151Ø
 '^
145Ø IF Q=67 THEN Q=255:Q1=149:N
=N+2:GOTO151Ø'cos
146Ø IF Q=84 THEN Q=255:Q1=15Ø:N
=N+2:GOTO151Ø'tan
147Ø IF Q=76 THEN Q=255:Q1=153:N
=N+2:GOTO151Ø'log
148Ø IF Q=73 THEN Q=255:Q1=129:N
=N+2:GOTO 151Ø' int
149Ø IF Q=83 THEN N=N+2:IF MID$(
E$,N-1,1)="I"THEN Q=255:Q1=133 E
LSE Q=255:Q1=155' sin or sqr
15ØØ IF Q=65 THEN N=N+2:IF MID$(
E$,N-1,1)="T"THEN Q=255:Q1=148 E
LSE Q=255:Q1=13Ø' atn or abs
151Ø POKE M,Q:IF Q=255 THEN M=M+
1:POKE M,Q1
152Ø M=M+1
153Ø NEXT
154Ø IF PEEK(M)<>Ø THEN POKE M,3
2:M=M+1:GOTO154Ø
155Ø RETURN
```

CoCo Cat



FORUM

by John Poxon

This month I have written a program intended to find the roots of a quadratic equation. Coincidentally this may help a certain student's mother who spoke recently to Graham about "quadratics".

If "quadratic equation" doesn't ring any bells with you, try this. If an equation of the form AxXxX + BxX + C = 0 exists, its roots (values of X which will result on a zero) may be found by a variety of means, including an equation method. The roots may be real, zero or complex.

I do not propose to explain the mathematics of this method of solution: instead I recommend that you consult a suitable high school mathematics text.

The program is listed below, as it appears on my screen. Note that since it employs only single length words, no value may exceed 32768. Type it in, compile it and run it by entering the variables A, B and C and the program name, i.e. A B C Q.

```
DECIMAL
VARIABLE BSQD
VARIABLE 4AC
VARIABLE 2A
VARIABLE NEGB
: A. ." A = " ROT DUP . ;
: B. ." B = " ROT DUP . ;
: C. ." C = " ROT DUP . ;
: LISTVAR A. B. C. CR CR ;
: SQBDO SWAP DUP DUP * BSQD ! SW
AP ;
: 4ACDO ROT DUP ROT * 4 * 4AC !
;
: 2ADO 2 * 2A ! ;
: -BDO NEGATE NEGB ! ;
: 1CALCS SQBDO 4ACDO 2ADO -BDO ;
: COMPLEX? BSQD @ 4AC @ - ;
: B@10* NEGB @ 10 * ;
: ROOT DUP 0 > NOT IF DROP 0 EXI
T THEN DUP 2/ 2/ 2/ 2/ 1+ BEGIN
2DUP / 2DUP - ABS 1 > WHILE + 2/
REPEAT ROT 2DROP ;
: NGETROOT BSQD @ 10 * 4AC @ 10
* - 10 * ROOT ;
: CGETROOT BSQD @ 10 * NEGATE 4A
C @ 10 * + 10 * ROOT ;
: TITLE PAGE CR ."   ROOTS OF QU
ADRATIC EQUATION" CR CR ;
: IF<1&NEG DUP 0= IF SWAP DUP 0<
IF ." -" SWAP THEN THEN ;
: PRINT 10 /MOD IF<1&NEG . ." ."
ABS . ;
: CPRINT DUP 10 < IF ." 0." . EL
SE 10 /MOD . ." ." . THEN ;
: N1ROOT B@10* NGETROOT NEGATE +
2A @ / ." 1ST REAL = " PRINT ;
: N2ROOT B@10* NGETROOT + 2A @ /
." 2ND REAL = " PRINT ;
: C1ROOT B@10* 2A @ / ." 1ST COM
PLEX = " PRINT ." +" CGETROOT 2A
@ / CPRINT ." J" ; : C2ROOT B@10
* 2A @ / ." 2ND COMPLEX = " PRIN
T ." -" CGETROOT 2A @ / CPRINT .
" J" ;
: NONCOMPLEX N1ROOT CR N2ROOT ;
: ISCOMPLEX C1ROOT CR C2ROOT ;
: Q TITLE LISTVAR 1CALCS COMPLEX
? 0< IF ISCOMPLEX ELSE NONCOMPLE
X THEN ;
```

I make specific acknowledgement of the source of the square root routine. It was written by John Redmond and appeared in the July 85 Coco on page 42.

The remaining words in the program (in order) have the following functions.

A. B. and C. print the value of the variable as entered and its name . They are contained in LISTVAR.

SQBDO squares B and stores its value in BSQD.

4ACDO calculates the value of 4AC and stores the value in the variable 4AC. Ditto 2ADO.

-BDO negates the value of B and stores it in NEGB.

1CALCS implements SQBDO, 4ACDO, 2ADO and -BDO.

COMPLEX? determines whether or not the quadratic equation has negative roots and leaves a flag on the stack. B@10* means get B and multiply it by 10.

ROOT finds the single length square root of a number on the stack.

NGETROOT calculates the (real) root of BSQD - 4AC.

CGETROOT similarly finds the (imaginary) root of BSQD - 4AC when 4AC is greater than BSQD. Note the manipulation of signs to make this possible. (Even a Coco can't find the square root of a negative number).

TITLE prints the title.

IF<1&NEG tests to see if a value is less than 1 and negative. If so a the number is manipulated to remove unwanted signs before printing.

PRINT prints a "real" number using IF<1&NEG if necessary some number formatting is included.

CPRINT does a similar job for an "imaginary" value.

N1ROOT, N2ROOT, C1ROOT and C2ROOT calculate and print the respective real or imaginary roots.

NONCOMPLEX calculates the real roots using N1ROOT and N2ROOT.

ISCOMPLEX similarly calculates the imaginary roots.

Q wraps the whole thing up: it

a) prints a title,
b) lists the values of A, B, and C,
c) does the preliminary arithmetic,
d) tests if the answer will be complex or not,
e) calculates a suitable answer.

The program unfortunately has a tendency to create errors which cannot be overcome using single length calculations. For accuracy, a double precision version needs to be written. More decimal places should be provided and truncated. How about it? Put another way, when is someone other than myself or John Redmond going to write something for this column?

That's it for this month. I don't know whether I can write anything for next month since my assignment for the U of Q (due September) are piling up, and also the Cococonf is upon us.

I hope that you like my little offering. Feel free to ring me if you want to discuss the contents of this article or other FORTH related issues. Regards, John Poxon. 07 208 7820.

Reviewing PenPal Applications

By Richard A. White
Rainbow Contributing Editor

Last month we began a discussion of the *PenPal* integrated software package from Four Star Software in Canada. This month we will discuss additional modules, starting with Calc.

Calc alone is a good, if not great, spreadsheet. It lacks an IF . . . THEN . . . ELSE statement and lookup tables, so you won't want to try to write an income tax spreadsheet with it. But it has a few neat features that make up for these omissions. If you want to plot data directly from spreadsheet files, the Graphit module is excellent. To make graphs, bar charts or pie charts, the Calc/Graphit combination may be the best available for the CoCo today.

As with all *PenPal* applications, Calc is function-key driven with a function-key strip displayed on the bottom two lines of the 51 character by 24 line screen. There are enough functions to require a primary and a secondary key strip. These are toggled by pressing CLEAR and '0'.

Spreadsheet size is 255 columns by 255 rows. Of course, you cannot use all of the cells at one time due to memory limitations. Still, you can use quite a few, since Calc is conservative in its use of memory. It seems to be quite competitive with *DynaCalc* in this regard. You start with 26,458 bytes of buffer space, so very sizable spreadsheets are possible.

Calc saves memory in other ways. Numbers are saved with nine place accuracy using only five bytes per entry. This is sufficient accuracy for most applications. If you need more, *DynaCalc* works to 16 digit accuracy, but needs a few more bytes to store each number.

When a cell is used, some memory is allocated for the cells in the block between cell A1 and the newly used cell. This is true of most spreadsheets. If you delete some of the columns or rows in that block, you don't immediately get back all the memory involved. However, if you save and then reload the sheet, cell pointers are reset for best memory economy.

Initially, label entry appears to be the same as many other spreadsheets. You can enter a long label, but only those characters that fit the column width are displayed. If you move the cursor to the cell, more of the text is displayed in the cell descriptor line (line two) at the top of the screen. During label entry, you can use the '@' key to backspace over the label to make corrections. Once the label is entered, there is no way to edit it. The same is true for numbers and formulas. This is Calc's greatest weakness, which is partially compensated for by the ability to define a text block.

A text block is new to me. An area of the screen is selected (it could be the whole screen, but not exceed screen boundaries) and defined as a text block using a function key. The cursor appears in the upper left-hand corner of the block. You now have a small text editor with which you can enter text in the block. Use the arrow keys to move over the text to do editing. Word wrap is not supported; you must do this manually. Exit a text block with the BREAK key.

Once defined, the text block remains available. When you place the cursor within the block and press the function key, you go back into the text editing mode. When not in the editing mode, labels, numbers and formulas can be put into text block cells in the normal manner. Those cells are removed from the block and work as normal spread-

---

Richard White lives in Fairfield, Ohio, and 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.*

sheet cells. This breaks the text block into pieces above and below, or right and left of the normal cells.

If you set up a text block, write text into it and then change the widths of columns passing through the block, the text display is broken up. Not to worry. Simply put the cursor inside the block and use the text function key, and the text is reformatted. If the size of the block is reduced so there is more text than space, the text is truncated. Further, without word wrap, words are broken at the right margin and you have to do some editing by hand. But it's better than typing it in from the beginning. Blocks can be defined for other purposes, including deleting, copying, printing and saving a portion of a spreadsheet either as a spreadsheet file or ASCII file.

The block copy function works much like copy in *Lotus 1-2-3* except you select the block first and then press the function key to define the block to be copied. Move the cursor to the upper left-hand cell you want to copy to and press the copy function key again. This leads us to absolute or relative addressing. Here is a simple example. This formula is in screen location F1: D1 + E1 + #E5. When I copy it to screen location F2 it becomes D2 + E2 + #E5.

Cell references D1 and E1 are treated like they refer to the second cell to the left and the first cell to the left, and are adjusted to maintain this same relation when the formula is copied to F2. E5 is preceded by a '#' which tells Calc not to adjust this cell reference. It is called an absolute address, which always is to reference the same cell irrespective of where the formula is copied.

Calc contains a typical selection of trigonometric and other functions and operands. Like *DynaCalc*, formulas are evaluated from left to right with no hierarchy of operands except the portions in parentheses which are evaluated innermost first. As I have cautioned before, this can lead to incorrect results if you do not assure that a multiplication or division is performed before the result is added or subtracted. Without hierarchy of operators this can happen unless you use parentheses to force the order of calculation.

Calc has a very limited number of built-in formulas, including AVG (average), MAX (maximum value in defined range), MIN (minimum value) and SUM (summation of all cells in the range). For each of these the range can be a block of cells. Whenever a spreadsheet offers built-in formulas like these, we need to know how each formula deals with empty cells and cells including text or labels. These functions assume that empty and label or text cells contain zeros. AVG(A1-A10) assumes that all 10 cells in the range contain numbers even if one or more don't. In such a case an incorrect average would be returned. MIN(A1-A10) gets confused in the same way and returns a zero if there is an empty or label-containing cell. MAX and SUM are not affected and always return a correct result.

Perhaps even more basic is the fact that Calc performs a calculation even when there are empty or label cells in the range. Some spreadsheets return an error, forcing the user to put zeros in empty cells and get rid of labels within the range. This means more work and untidy results.

Those whose first spreadsheet was *Spectaculator* will remember CMT and RMT for cumulative sum of a column or row. That was all we had four years ago, but why include these formulas when the more able and understandable SUM is included? If you use Calc, forget CMT and RMT.

Finally, a cell can be set up to receive a constant at the time of calculation. With this, you could set up a form and prompt the user to make entries one by one, assuring all required numbers are entered. You could type in letters or a word, but this shows up as a zero. It would be nice if words could be entered and placed in cells in a prompted mode. Then all sorts of forms could be put together in a spreadsheet, and be filled out following prompts on the entry line. Please, Four Star, add this to the Version 3 wish list.

To set up a cell for prompted entry, put the cursor on the cell, press the formula function key and then type ? and the text to be in the prompt. All formulas are entered by positioning the cursor and using the formula function key to get into formula entry mode.

I mentioned you can save a portion of a spreadsheet to a file. You can also merge that save or any spreadsheet into another. The merge starts at the current cursor location when merge is called and proceeds right and down. One obvious use is to move data from one spreadsheet to another. Another is to combine a number of smaller spreadsheets into one big one.

There are a couple of unusual defaults. If you try to print or make an ASCII save without previously selecting a range to save or print, the screen is selected as the default range. All in all, Calc is a nice piece of work and adds major value to the *PenPal* package.

We come now to Graphit that works with a Calc file to produce plots of data selected from the file. Similar capabilities are included in the Disk BASIC Version of *DynaCalc*, but not in the OS-9 version. Graphit makes line, dot or bar graphs in either horizontal or vertical directions. It also makes pie charts, and the segments can be filled with textures selected from nearly 100 available choices.

Graphit has powerful graph-labeling capabilities, including a selection of font sizes ranging from 32 to 64 columns. You also have control of the display style. For example, you might want to use light letters over a dark texture or surround dark letters with a light box over a dark background.

Graphs may be saved to a /GP file to be later loaded back into Graphit, or as a binary file that can be loaded into some other graphics program, or to a BASIC program for display or modification using BASIC's graphics commands. Finally, you can print graphs either single or double size. A number of the most popular printer protocols are supported. The pie chart prints oval on my LP VIII in double size, but is close to round in normal size. It also appears oval on the screen.

Graphit works from function-key strips displayed only when the CLEAR key is pressed. It's easy to do the basics and to redraw the graph in different ways to choose the one with most clarity or impact. Lettering the graph is easy, as is choosing and using textures for the pie chart.

Graphit has only an 8,500-byte buffer for the spreadsheet file it is to plot. Calc can deal with spreadsheets three times that size. When working with a larger spreadsheet you need to save the parts of the sheet with the data to be plotted to separate small files. Now we can better appreciate Calc's ability to make such small files.

Telecom is a full-featured telecommunication program that uses the RS-232 port on the CoCo. For me, this is unfortunate since my port is not working. It is also confusing, since the printer that works off the same plug works fine from the CoCo. Who said computers were rational? At least my diagnostics ROM agrees the port is bad.

I have the Radio Shack RS-232 ROM Pak, which both *Mikeyterm* and *DeskMate* use, so I have all the telecommunications capability I need. Still I would like to have used Telecom rather than just reporting on the documentation.

Things can get foggy in a hurry in telecommunications. This is partly due to the variety of settings and options available. Delphi, CompuServe and most microcomputer-based bulletin boards are fairly standard and work at

the default settings supplied with Telecom, *DeskMate* and other packages. Telecom has all the tools needed to deal with nonstandard beasts if you need to use them.

You do not need a modem to use Telecom. You can connect directly with another computer in the same room or nearby. You will need a special three-wire cable called a null modem. One is easily made from components stocked by all Radio Shack Stores. You will need the right plug to go into the RS-232 plug on each computer. The CoCo uses an oddball four-pin DIN plug. The best way to get one is to buy a CoCo printer/modem cable, 26-3020 for $4.95. If connecting two CoCos, swap the wires to pins 2 and 4 on one plug. Plug the cable into the two machines, load the terminal programs and go.

If you are connecting to a different computer, it will most likely need a standard, 25-pin RS-232 male plug. I would still start with the CoCo cable and replace one four-pin DIN with the RS-232 plug. Pins 1, 2, 3 and 4 on the CoCo plug go to pins 8, 2, 7 and 3 on the RS-232.

If you plan to work over the telephone, you need a modem and a modem cable. The modem cable mentioned works only with Radio Shack modems. Other modems usually require a male RS-232 plug on the cable. You can make a cable just like the null modem cable above, except the wires to pins 2 and 3 on the RS-232 plug are swapped.

Starting at the top we come to Baud rate. Some years ago, 300 Baud was fairly standard. All communications services support this rate. If you are connecting directly to another computer, you can use the highest rate commonly supported by the two computers. Telecom will run at 2400 Baud provided the screen display is turned off. Otherwise, 600 Baud is recommended to avoid losing data. Of course, I want to work the bulletin board at 1200 Baud and I could not do that very well with my screen display off. Maybe I'll find a way if I get my serial port fixed.

Telecom defaults to 300 Baud. All the other defaults look good except duplex. The default is not to echo typed characters to the screen. Most bulletin boards send back each character as they receive it and Telecom dutifully displays that character so you know what you typed. This is called full duplex. If you hook up to another computer using a package like Telecom, it most likely will not echo characters as it receives them. This also goes for time-sharing services running on IBM mainframes. So now you must turn half duplex on in Telecom to see what you are typing.

Telecom supports a powerful autolog file capability. You are able to change any communications parameters from an autolog file. Say you had been having a session on a local bulletin board and want to call up your company's office mail system running on an IBM mainframe.

First, you empty the buffer and enter the IBM autolog file. Execute this file by pressing the function key. The first thing it might do is change from full to half duplex. Next, if you have an autodial modem, it could send the modem instruction to dial including the number to dial. Once connection is made, the office machine manages the user's logon by asking for certain responses. The autolog file can contain instructions to look for prompts and to send specified character strings at each prompt which completely automates the logon. This can be particularly useful on a bulletin board with nested menus. You could be automatically logged on and have the mail reading started.

When I upload or download, I nearly always use the CPM or Xmodem protocol. Those are two names for the same thing. Basically, 128-byte buffers and a check digit are sent at one time. The receiving software recalculates the check digit and compares it to the one it received. If they check, successful transmission is acknowledged and the next buffer is sent. A mismatch means there was a transmission error and the sending computer is caused to resend the buffer. Telecom supports the Xmodem protocol.

In summary, *PenPal*'s Calc is a step or two below *DynaCalc* in functions, is comparable in memory efficiency and has a superior graphing capability in Graphit. In some ways, Telecom is a bit better than the public domain *Mikeyterm* and does everything most other terminal packages do. Couple this with the competent word processor discussed last month and the file module yet to be discussed, and at $89.95, you have one of the best software values on any computer today.

BARDEN'S BUFFER

The Adventure of the Too Many Printers

By William Barden, Jr.
Rainbow Contributing Editor

"My dear fellow," said Sherlock Holmes, as we sat on either side of the fire in his lodgings on Baker Street. "I really wish you would consider buying a Color Computer 2 in place of that older model. I know that you are exceedingly unhappy with your present machine."

I turned to look in amazement at the tall spare figure seated in the armchair next to the Tandy 3000. A glimmer of of interest was present in his eyes.

"How could you possibly know that, Holmes?"

"Elementary, my dear Watson. I know from observing you at the computer that you are a two-fingered typist, using the index fingers of both hands. The tips of those fingers and those fingers alone have calluses, which you've been peering at intently all evening with some dismay, I might add. I also observe that you've replenished your supply of diskettes with those in Radio Shack envelopes, indicating a visit to the local Radio Shack Computer Center, as our seventeen local stores never carry more than three each, or three of any product, for that matter. The Radio Shack computer catalog is also lying open to the page displaying Color Computer products. All of these minutiae point to the fact that you're seriously considering the new model. Furthermore, I saw the sales slip."

I could not help laughing at the ease with which he explained his process of deduction. "It is, indeed, obvious, Holmes."

"Quiet, Watson," my friend suddenly whispered. "Unless I miss my guess that will be our friend I observed a moment ago coming up the stairs. We are about to be visited by a stout fellow approximately six feet tall, weighing 280 pounds and wearing a T-shirt upon which is emblazoned 'I Love My CoCo.'"

A knock sounded on the door.

"Come in," Holmes replied.

The door opened and a stout fellow approximately six feet tall and about 280 pounds entered. He was wearing a T-shirt upon which was printed "I Love My Tandy 1000."

"Is this PCM Magazine?" he queried.

"I'm sorry, you have the wrong story," Holmes replied, with some brusqueness.

The caller made a quick egress. Another knock sounded at the door.

Holmes shouted out, "Come in!" The door opened and a stout fellow approximately six feet tall and about 280 pounds entered. He was wearing a T-shirt upon which was emblazoned "I Love My Color Computer 2."

"Close enough," Holmes muttered.

"Are you Sherlock Holmes, the famous detective?" the caller asked anxiously, his eyes downcast as if he were in deep despair.

"Yes, Mr. Purcell-Smith," Holmes replied.

"But how did you know my name, Mr. Holmes?"

"I'm afraid, Mr. Purcell-Smith, that you've achieved some notoriety since your recent marriage to Joan Purcell, the chief programmer of Slothware. How may I help you?"

I winced as I heard Holmes use that pat phrase gleaned from too many visits to his local Radio Shack.

"Mr. Holmes, you must help me!" Purcell-Smith blurted. "One of our programmers has been found dead and my wife has been arrested by Lestrade of the Fort Worth Yard!"

"Calm down, man!" said Holmes, sternly. "Tell us your story from the beginning."

"Well, Mr. Holmes, my wife has been working for some time now on a new project for Slothware. It's a screen dump program for the Color Computer. Never having written such a program before, she ran into several major obstacles.

"First of all, she had to figure out how points on the screen were held in memory and where that memory was located."

"That doesn't appear to be a great problem," said Holmes. "As I recall, the Color Computer holds its graphics screens starting at location 1536 decimal for nondisk systems and starting at location 3584 decimal for disk systems. Each screen is 6,144 bytes long for graphics modes 3 and 4. For graphics mode 3, 128 by 192, there are 24,576 elements of two bits each. For graphics mode 4, there are 49,152 elements of one bit each."

As Holmes spoke, he rapidly sketched two drawings, one representing the 128 by 192 two-color graphics mode 3 and the second representing the 256 by 192 two-color graphics mode 4 (see figures 1 and 2).

"I wrote a monograph on the subject after my episode involving the Giant Computer of Sumatra. But please continue, Mr. Purcell-Smith."

"The next problem my wife ran into was computer graphics. It seems that most Radio Shack printers use the same scheme for graphics, one involving seven bits per character position." Our visitor paused and shot a baffled glance at Holmes.

"Ah, yes. The singular seven-bit problem. An interesting exercise. It appears that early in the printer game someone at Radio Shack decided to design a graphics printer. Most printers at the time used a five by seven dot matrix for each character. Each ASCII, or text, character was converted by the printer electronics into a series of five vertical columns, each column consisting of seven possible dots." He sketched another rapid illustration on the bottom of a Persian slipper (see Figure 3).

"What could have been more natural than implementing graphics in the existing printers than to make each one of those seven print wires or print positions programmable in graphics mode. All that was required was a

command to 'set graphics mode' in the printer. After graphics mode was set, the printer decoded each character that came to it as a coded form of which of those seven wires to print. The only requirement for each graphics character was it could not be in the normal range of ASCII characters from zero through 127. It must be in the range of 128 through 255."

"Yes, Mr. Holmes, I understood that part of it, but could not see how the encoding took place."

"Imagine that the topmost dot of each column was represented by a one, the next by a two, the next by a four, the next by an eight, the next by a 16, the next by a 32, and the last by a 64. Once graphics mode is set, you simply have to add the dot values for the column plus 128 to mark graphics mode. Let me illustrate."

Holmes took out a pistol and aimed it at a spot on the wall next to the letters "M. T." created by bullet holes. (Victoria Regina had yielded to Maggie Thatcher years ago.)

"Suppose you wanted to print the upper three dots and the bottommost dot in a graphics column." He fired the gun four times to denote the column. "The code for the graphics column would be one plus two plus four plus 64 for the bottommost column plus 128 to mark the character as graphics. That would be a total of 191."

"But how would that 191 value be sent to the printer, Mr. Holmes? You can send text by doing a PRINT#-2 of a string, but how can a nonprintable character be sent?"

"By using the CHR$( ) function, Mr. Purcell-Smith. As a matter of fact, many printer codes are sent to the printer from BASIC by using the CHR$( ) function. To set graphics mode, for example, you'd do a PRINT#-2,CHR$(27);CHR$(XX). To print the graphics column we've been discussing, you'd do a PRINT#-2,CHR$(191);. Use a semicolon at the end if you don't wish to end the line."

"But," said he, "pray let us continue our conversation on the way to the location of the demise of that poor programmer chap."

There was a cab passing as the three of us came out of the building, and Holmes hailed it. "Beckenham, if you please."

Our driver pulled up at a greystone that housed a high-tech industry. He knocked on the door, and a tall, dour-looking man appeared.

"Ah, Lestrade," Holmes said. I see you have apprehended the killer and are putting the finishing touches to this little mystery."

"Yes, Mr. Holmes. It was a simple crime, really. Ms. Purcell was jealous of her chief programmer and afraid that even with affirmative action he would soon replace her at Slothware. She laced his coffee with arsenic as he was putting the final frills on a screen dump program."

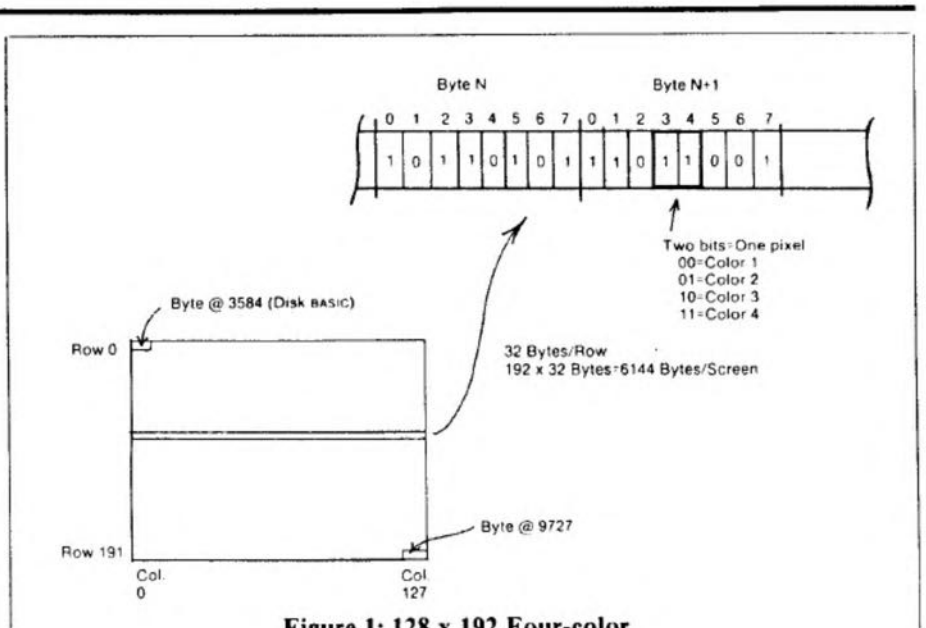


Figure 1: 128 x 192 Four-color Graphics Mode

"In that case, Lestrade, you won't mind if we have a look around, will you?".

"Of course not, Mr. Holmes, although I'm certain you won't be finding anything that I haven't seen already."

Holmes walked over to the programming area and picked up a listing next to a Color Computer. "Hmm. This appears to be a listing of the screen dump program in question, gentlemen. Let me peruse this briefly."

After intently peering at the document, the detective picked up a second listing. He then glanced around the room at several printers connected to the Color Computer. Taking out a pocket measuring tape, he made precise measurements of figures produced by the several printers.

"Watson, I think our task here is done. Lestrade, I think if you look in that programmer's desk drawer you'll find a vial of arsenic with *his* fingerprints on it, prompting you, I should think, to release Ms. Purcell from your custody. Let us be off, gentlemen."

Later that evening the three of us sat in front of a blazing fire in Holmes' Baker Street apartment.

"Mr. Holmes, I cannot thank you enough! My wife has been released and Lestrade has ruled the death a suicide."

"My blushes, Mr. Purcell-Smith. It was just a trifle."

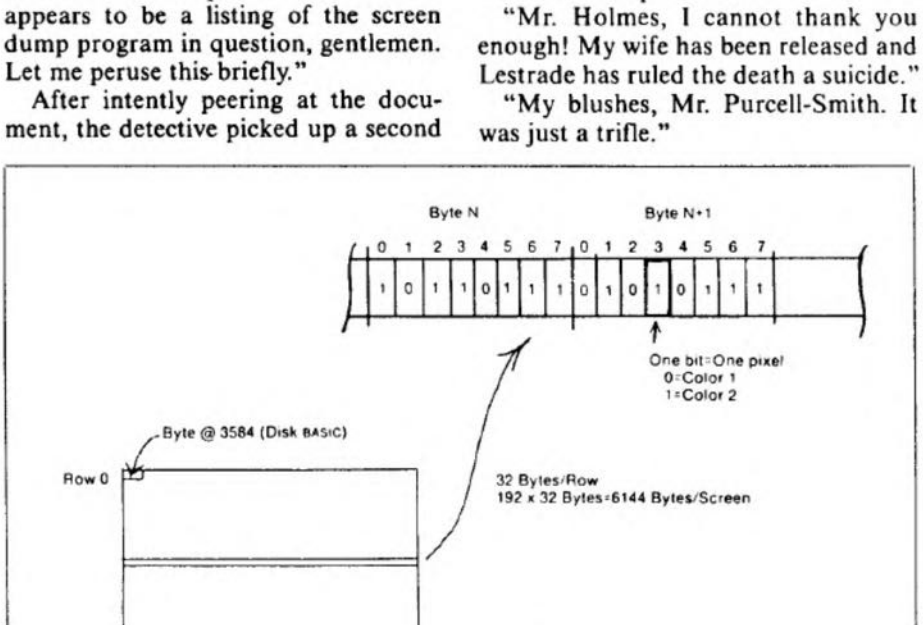


Figure 2: 256 x 192 Two-color Graphics Mode

"But, how did you know the death was a suicide?"

"I looked in some detail at the listing. It appears that the dead programmer had been trying for some time to design a screen dump program for the Color Computer. It was apparent he was having problems, however. The revision number of the BASIC program was number 45, indicating that he had attempted 44 times to write a program that successfully dumped the screen."

"But surely he must have known about the CHR$ command, memory layout of the graphics screen and coding of graphics characters?"

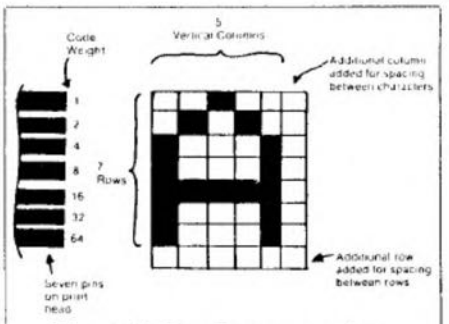


Figure 3: Dot Representation of Characters and Graphics

"Yes, he was familiar with those functions for the screen dump program. However, he encountered a host of other obstacles. First, he found a problem in translating one pixel on the screen into one printable spot. Consider one approach he might have used. If he made one pixel equal to one printable character, then he would have had a screen dump of 25.6 inches in width by 32 inches high for the 256 by 192 mode, as the usual number of characters per width horizontally is 10 characters per inch and the number of vertical lines per inch is six. He was forced to use graphics mode to get a small enough listing.

"However, he knew he could not print one pixel at a time on the printer, as seven vertical dots are printed for every graphics character. Therefore, he had to construct a graphics character made up of the seven vertical pixels on the screen. Here, I'll show you what I mean."

Holmes grabbed a nearby coal scuttle and drew a figure on it with a marking pen.

"He had to repeat that process for the entire row of 256 characters. When he coded the program, however, he found, to his amazement, that the figure produced by the screen dump was too small. The width was 256/60 inches wide, as most Radio Shack printers print 60 dots per inch in normal mode. The 4.4-inch width however, did not match the height. Most Radio Shack printers space graphics lines at 0.12 inches vertically, about eight lines per inch. As he had to print 27 3/7 graphics lines to produce 192 pixels vertically, he found the vertical dimension of the screen dump figure was 3.3 inches. A 4.4 by 3.3 inch figure wasn't to his liking.

"On his next attempt, he printed two graphics characters for every column of seven pixels." Holmes continued the sketch (see Figure 4).

"To do that, however, he had to make each graphics character represent 3.5 pixels, drawing two graphics dots for each pixel on the screen. That proved to be more laborious programming because he had to alternate between odd and even numbered lines. Each even line on the printer represented the next three pixels and the top half of the next. Each odd line represented the bottom half of the last pixel and the next three pixels.

"He wrote his next revision using these criteria. The result produced a print image of 8.5 inches by 6.6 inches. However, he encountered a strange problem. The line printer would not print the entire line of graphics characters without advancing to the next line. After much deliberation, indicated by revisions 20 through 23, he found that Color Computer BASIC counted the number of characters in each line and assumed each character was a text character. After 132 characters, in the typical case, BASIC would automatically send a new line to the printer, assuming that no printer would have greater than 13.2 inches for each line. That would be true had the printer been printing text, but not 512 graphics characters occupying 8.5 inches.

"In his next revision he cleared the BASIC variable that defined the number of characters per line to zero, eliminating *that* problem.

"The unfortunate programmer finally had a program that would print a 256 by 192 pixel screen in PMODE 4. With a few more modifications, he created a new revision that would also print in PMODE 3, the four-color 128 by 192 mode.

"In the process of running the screen dump program, however, he timed it. I would guess by the nested loops and 'overhead' of the program it would take approximately 52 minutes to print out the screen. Our friend knew this would be unacceptable as a product.

"He then set out to convert the BASIC

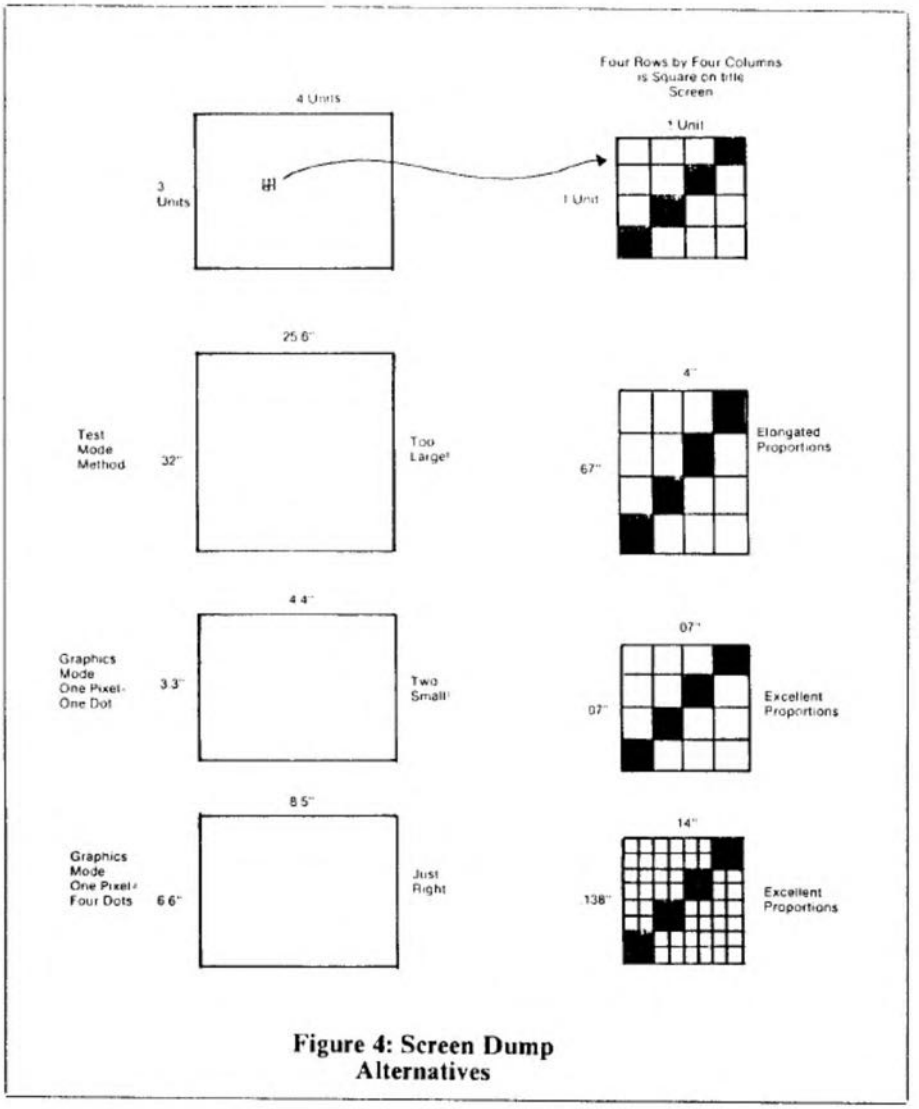


Figure 4: Screen Dump Alternatives

gram did execute quite rapidly, about eigh. minutes for each screen print. One can surmise that he went with good spirits to his employer, Ms. Purcell, to show her his accomplishment. At this point he was many days overdue. Ms. Purcell has confirmed that he did, indeed, show her the program. When they went to try it on her CGP-220 printer, however, the proportions were quite far off — 6.4 inches horizontally by 6.6 inches vertically, as a matter of fact. Our friend was aghast until he found out that the CGP-220 printer printed 80 graphics columns per inch.

According to Ms. Purcell, he then attempted to run the program on the DMP-110 dot-matrix printer. Again the proportions were off — 4.2 inches by 5.5 inches. To his dismay he found the DMP-110 used 120 graphics columns per inch! At that point, according to Ms. Purcell, he returned to his desk, in a kind of daze, repeating the words 'too many printers, too many printers.' She found him an hour later in the rictus of death clutching his last listing, which I have here." Holmes pulled out the last listing and placed it in front of us.

"The ironic part of this story, gentlemen, is that his last listing worked for most Radio Shack printers. Such is the life of a programmer! Let us hope our friend has received his eternal reward for his efforts and is employed in a heavenly capacity on a system with one, and only one printer."

How to Use the Screen Print Programs

There are three versions of the screen print program. Each runs on any Radio Shack printer that has "dot graphics" capability, including earlier printers such as the LPVIII, DMP-100, DMP-

program to Color Computer assembly language. The poor fellow had a nodding acquaintance with assembly language, but did not realize what a laborious task the conversion would be.

"After hours and many program revisions, presumably, he produced this listing." Holmes thrust a soiled, tattered piece of paper at us. "In fact this pro-

print one pixel at a time on the printer, as seven vertical dots are printed for every graphics character. Therefore, he had to construct a graphics character made up of the seven vertical pixels on the screen. Here, I'll show you what I mean."

Holmes grabbed a nearby coal scuttle and drew a figure on it with a marking pen.

"He had to repeat that process for the entire row of 256 characters. When he coded the program, however, he found, to his amazement, that the figure produced by the screen dump was too small. The width was 256/60 inches wide, as most Radio Shack printers print 60 dots per inch in normal mode. The 4.4-inch width however, did not match the height. Most Radio Shack printers space graphics lines at 0.12 inches vertically, about eight lines per inch. As he had to print 27 3/7 graphics lines to produce 192 pixels vertically, he found the vertical dimension of the screen dump figure was 3.3 inches. A 4.4 by 3.3 inch figure wasn't to his liking.

"On his next attempt, he printed two graphics characters for every column of seven pixels." Holmes continued the sketch (see Figure 4).

"To do that, however, he had to make each graphics character represent 3.5 pixels, drawing two graphics dots for each pixel on the screen. That proved to be more laborious programming because he had to alternate between odd and even numbered lines. Each even line on the printer represented the next three pixels and the top half of the next. Each odd line represented the bottom half of the last pixel and the next three pixels.

"He wrote his next revision using these criteria. The result produced a print image of 8.5 inches by 6.6 inches. However, he encountered a strange problem. The line printer would not print the entire line of graphics characters without advancing to the next line. After much deliberation, indicated by revisions 20 through 23, he found that Color Computer BASIC counted the number of characters in each line and assumed each character was a text character. After 132 characters, in the typical case, BASIC would automatically send a new line to the printer, assuming that no printer would have greater than 13.2 inches for each line. That would be true had the printer been printing text, but not 512 graphics characters occupying 8.5 inches.

"In his next revision he cleared the BASIC variable that defined the number of characters per line to zero, eliminating *that* problem.

"The unfortunate programmer finally had a program that would print a 256 by 192 pixel screen in PMODE 4. With a few more modifications, he created a new revision that would also print in PMODE 3, the four-color 128 by 192 mode.

"In the process of running the screen dump program, however, he timed it. I would guess by the nested loops and 'overhead' of the program it would take approximately 52 minutes to print out the screen. Our friend knew this would be unacceptable as a product.

"He then set out to convert the BASIC program to Color Computer assembly language. The poor fellow had a nodding acquaintance with assembly language, but did not realize what a laborious task the conversion would be.

"After hours and many program revisions, presumably, he produced this listing." Holmes thrust a soiled, tattered piece of paper at us. "In fact this program did execute quite rapidly, about eight minutes for each screen print. One can surmise that he went with good spirits to his employer, Ms. Purcell, to show her his accomplishment. At this point he was many days overdue. Ms. Purcell has confirmed that he did, indeed, show her the program. When they went to try it on her CGP-220 printer, however, the proportions were quite far off — 6.4 inches horizontally by 6.6 inches vertically, as a matter of fact. Our friend was aghast until he found out that the CGP-220 printer printed 80 graphics columns per inch.

According to Ms. Purcell, he then attempted to run the program on the DMP-110 dot-matrix printer. Again the proportions were off — 4.2 inches by 5.5 inches. To his dismay he found the DMP-110 used 120 graphics columns per inch! At that point, according to Ms. Purcell, he returned to his desk, in a kind of daze, repeating the words 'too many printers, too many printers.' She found him an hour later in the rictus of death clutching his last listing, which I have here." Holmes pulled out the last listing and placed it in front of us.

"The ironic part of this story, gentlemen, is that his last listing worked for most Radio Shack printers. Such is the life of a programmer! Let us hope our friend has received his eternal reward for his efforts and is employed in a heavenly capacity on a system with one, and only one printer."

How to Use the Screen Print Programs

There are three versions of the screen print program. Each runs on any Radio Shack printer that has "dot graphics" capability, including earlier printers such as the LPVIII, DMP-100, DMP-400, DMP-2100, and so forth. The screen print programs also run on newer Radio Shack printers when the printers are set to the Tandy character set. Newer printers have a switch selection that enables either Tandy or IBM characters to be used. The reason for this is the printers are used in the Tandy 1000, 1200, 2000 and 3000 — systems which are IBM PC compatible. These systems not only use a different character set, but also use a different escape code sequence, one which is Epson printer compatible. Unfortunately for the programmer in our story, the new printers arrived too late. The Epson escape-code sequence uses an eight-bit encoding which is a great deal easier to work with

than the seven-bit coding.

The first program, shown in Listing 1, is a BASIC program that works for either PMODE 4 (256 by 192) or PMODE 3 (128 by 192). To use the program, key it in as a subroutine starting at Line 10000. Then call it from your BASIC program at any time to print out the graphics screen.

Before you do, however, change Line 10010 to ZP=1536 for a nondisk system or to ZP=3584 for a disk system. Change Line 10020 to ZM=1 for PMODE 4 or to ZM=2 for PMODE 3. If you have a printer such as the DMP-110 which prints at a dense resolution in graphics, change Line 10030 to ZW=1; otherwise leave ZW=0. The program takes about 50 minutes to execute.

The second version of the program is an assembly language implementation of the same program shown in Listing 1. In this program we tried to do a straight translation between the BASIC version and assembly language so that you could see the flow in assembly language. Change the variables where indicated for disk/nondisk, PMODE 3 or 4, and high-density printing at the beginning of the program. Assemble and load the program using the Disk *EDTASM* or another assembler. The program is designed to execute at location $3E00, so you'll have to do a CLEAR 100,&H3DFF in your BASIC program before loading the program. You can then call the machine language code of the program by defining the location of the program with a DEFUSR0=&H3E04 and an A=USR0(0) or with a POKE 126,XX: POKE 127,XX and an A=USR(0) in non-Extended BASIC. This assembly language version takes about 8 minutes to print.

If you would rather use the program as a BASIC version, enter the program shown in Listing 3. This is the exact same version as Listing 2, except the machine code has been entered as a series of DATA values. These values are relocated to the $3E00 area the first time the program is called. Thereafter the program is executed by executing the machine code at the $3E00 area. You must first do a CLEAR 100, &H3DFF in your BASIC program before executing the program so that the $3E00 area is protected from BASIC use. Change the variables as before for the PMODE, disk/nondisk system and printer type.

The PMODE 3 version prints nonzero colors as foreground (print) and zero colors as background (no print).

A sample printout using the program is shown in Figure 5. This figure illustrates the graphics characters available in graphics mode by using the Character Generator discussed in last month's column. □

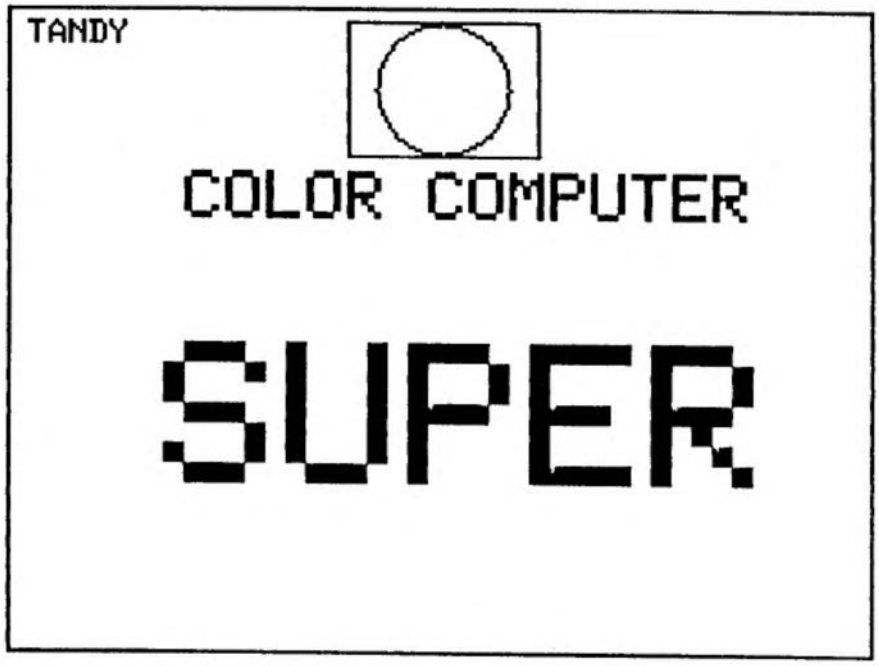


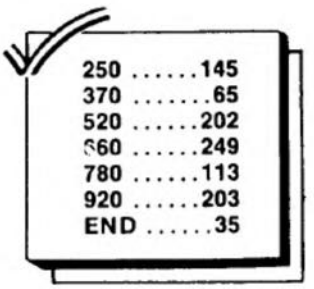
Figure 5: Sample Printout Using Hi-Res Program

Listing 2:

```
                        00100 *-----------------------------------------------
                        00110 *HI-RES SCREEN PRINT
3E00                    00120         ORG     $3E00
                        00130 *ZP=3584 'CHANGE TO ZP=1536 FOR NON-DISK SYSTEM
3E00    0E00            00140 ZP      FDB     3584
                        00150 *ZM=1 'CHANGE TO ZM=2 FOR 128X192 RES
3E02    01              00160 ZM      FCB     1
                        00170 *ZW=0  'CHANGE TO ZW=1 FOR DMP-110
3E03    00              00180 ZW      FCB     0
                        00190 *-----------------------------------------------
                        00200 *ZS=-ZM
                        00210 *-----------------------------------------------
3E04 86 3E02            00220 START   LDA     ZM
3E07 40                 00230         NEGA            -ZM
3E08 B7 3F46            00240         STA     ZS      INIT ZS
                        00250 *-----------------------------------------------
                        00260 *PRINT #-2,CHR$(16);
                        00270 *-----------------------------------------------
3E0B 86 10              00280         LDA     #16     SET GRAPHICS
3E0D BD 3F36            00290         JSR     PRINT
                        00300 *-----------------------------------------------
                        00310 *FOR ZR=0 TO 191 STEP 3.5
                        00320 *-----------------------------------------------
3E10 4F                 00330         CLRA            0
3E11 B7 3F45            00340         STA     ZR      INITIALIZE ZR
3E14 B7 3F48            00350         STA     ZT
                        00360 *-----------------------------------------------
                        00370 *FOR ZC=0 TO 31
                        00380 *-----------------------------------------------
3E17 4F                 00390 NXR060  CLRA
3E18 B7 3F42            00400         STA     ZC      INITIALIZE ZC
                        00410 *-----------------------------------------------
                        00420 *FOR ZB=7 TO 0 STEP -1
                        00430 *-----------------------------------------------
3E1B 86 07              00440 NXR070  LDA     #7
3E1D B7 3F43            00450         STA     ZB      INIT ZB
                        00460 *-----------------------------------------------
                        00470 *ZZ=2^ZB : IF ZM=2 THEN ZZ=ZZ+2^(ZB-1)
                        00480 *-----------------------------------------------
3E20 F6 3F43            00490 NXR080  LDB     ZB      GET # OF TIMES
3E23 4F                 00500         CLRA
3E24 1F 01              00510         TFR     D,X
3E26 C6 01              00520         LDB     #1
3E28 8C 0000            00530         CMPX    #0
3E2B 27 1D              00540         BEQ     NXR086  GO IF 0
3E2D 58                 00550         LSLB            SHIFT LEFT
3E2E 3A                 00560         LEAX    -1,X    DECREMENT
3E2F 26 F9              00570         BNE     NXR084  CONTINUE IF NOT DONE
3E32 B6 3E02            00580         LDA     ZM
3E35 81 02              00590         CMPA    #2
3E37 26 04              00600         BNE     NXR087
3E39 54                 00610         LSRB
3E3A FA 3F47            00620         ORB     ZZ
3E3D F7 3F47            00630 NXR087  STB     ZZ
                        00640 *-----------------------------------------------
                        00650 *FOR ZH=0 TO 3
                        00660 *-----------------------------------------------
3E40 108E 0000          00670         LDY     #0
                        00680 *-----------------------------------------------
                        00690 *IF (PEEK(ZP+ZC+(INT(ZR)+ZH)*32) AND ZZ)>0 THEN
                        00700 *ZG(ZH)=1 ELSE ZG(ZH)=0
                        00710 *-----------------------------------------------
```

```
3E44 4F          00720 HIR000  CLRA
3E45 A7  A9 3F49 00730         STA     ZG,Y     ELSE ZG(ZN)=0
3E49 1F  20      00740         TFR     Y,D
3E4B F7  3F44    00750         STB     ZN
3E4E F6  3F45    00760         LDB     ZR       ZR
3E51 FB  3F44    00770         ADDB    ZN       ZR+ZN
3E54 86  20      00780         LDA     #32
3E56 3D          00790         MUL              (ZR+ZN)*32
3E57 F3  3E00    00800         ADDD    ZP       ZP+ZC+(ZR+ZN)*32
3E5A 34  06      00810         PSHS    D
3E5C 4F          00820         CLRA
3E5D F6  3F42    00830         LDB     ZC
3E60 E3  E1      00840         ADDD    ,S++
3E62 1F  01      00850         TFR     D,X
3E64 A6  84      00860         LDA     ,X       PEEK(ZP+ZC+(ZR+ZN)*32)
3E66 B4  3F47    00870         ANDA    ZZ       AND ZZ
3E69 27  06      00880         BEQ     HIR100   IF (PEEK)... AND ZZ)=0
3E6B 86  01      00890         LDA     #1
3E6D A7  A9 3F49 00900         STA     ZG,Y     THEN ZG(ZN)=1
                 00910 *-----------------------------------------
                 00920 *NEXT ZN
                 00930 *-----------------------------------------
3E71 31  21      00940 HIR100  LEAY    +1,Y     STEP 1
3E73 1F  20      00950         TFR     Y,D      GET ZN
3E75 1083 0004   00960         CMPD    #4       =4?
3E79 26  C9      00970         BNE     HIR000   GO IF NO
3E7B B6  3F45    00980         LDA     ZR
3E7E 81  BD      00990         CMPA    #189
3E80 26  08      01000         BNE     HIR105
                 01010 *-----------------------------------------
                 01020 *IF ZR-189 THEN ZG(3)=0
                 01030 *-----------------------------------------
3E82 86  00      01040         LDA     #0
3E84 31  3F      01050         LEAY    -1,Y
3E86 A7  A9 3F49 01060         STA     ZG,Y
                 01070 *-----------------------------------------
                 01080 *IF ZR/2=INT(ZR/2) THEN ZG$=CHR$(128+ZG(0)*3+
                 01090 *  ZG(1)*12+ZG(2)*48+ZG(3)*64
                 01100 *-----------------------------------------
3E8A B6  3F48    01110 HIR105  LDA     ZT       GET ZT
3E8D 84  01      01120         ANDA    #1       TEST ODD/EVEN
3E8F 26  2B      01130         BNE     HIR110   GO IF ODD
3E91 B6  3F4C    01140         LDA     ZG+3
3E94 C6  40      01150         LDB     #64
3E96 3D          01160         MUL              ZG(3)*64
3E97 34  04      01170         PSHS    B
3E99 B6  3F48    01180         LDA     ZG+2
3E9C C6  30      01190         LDB     #48
3E9E 3D          01200         MUL              ZG(2)*48
3E9F EB  E4      01210         ADDB    ,S       ZG(2)*48+ZG(3)*64
3EA1 E7  E4      01220         STB     ,S
3EA3 B6  3F4A    01230         LDA     ZG+1
3EA6 C6  0C      01240         LDB     #12
3EA8 3D          01250         MUL              ZG(1)*12
3EA9 EB  E4      01260         ADDB    ,S       ZG(1)*12+ZG(2)*48+ZG(3)*64
3EAB E7  E4      01270         STB     ,S
3EAD B6  3F49    01280         LDA     ZG
3EB0 C6  03      01290         LDB     #3
3EB2 3D          01300         MUL              ZG(0)*3
3EB3 EB  E0      01310         ADDB    ,S+      ZG(0)*3+ZG(1)*12+ZG(2)*48+...
3EB5 CB  80      01320         ADDB    #128     128+ ...
3EB7 F7  3F49    01330         STB     ZG
3EBA 20  29      01340         BRA     HIR120
                 01350 *-----------------------------------------
                 01360 *ELSE ZG$=CHR$(128+ZG(0)*1+ZG(1)*4+ZG(2)*24+ZG(3)*96
                 01370 *-----------------------------------------
3EBC B6  3F4C    01380 HIR110  LDA     ZG+3
3EBF C6  60      01390         LDB     #96
3EC1 3D          01400         MUL              ZG(3)*96
3EC2 34  04      01410         PSHS    B
3EC4 B6  3F48    01420         LDA     ZG+2
3EC7 C6  18      01430         LDB     #24
3EC9 3D          01440         MUL              ZG(2)*24
3ECA EB  E4      01450         ADDB    ,S       ZG(2)*24+ZG(3)*96
3ECC E7  E4      01460         STB     ,S
3ECE B6  3F4A    01470         LDA     ZG+1
3ED1 C6  06      01480         LDB     #6
3ED3 3D          01490         MUL              ZG(1)*6
3ED4 EB  E4      01500         ADDB    ,S       ZG(1)*6+ZG(2)*24+ZG(3)*96
3ED6 E7  E4      01510         STB     ,S
3ED8 B6  3F49    01520         LDA     ZG       ZG(0)*1
3EDB C6  01      01530         LDB     #1
3EDD 3D          01540         MUL
3EDE EB  E0      01550         ADDB    ,S+      ZG(0)*1+ZG(1)*6+ZG(2)*24+ZG(3)*
96
3EE0 CB  80      01560         ADDB    #128
3EE2 F7  3F49    01570         STB     ZG
                 01580 *-----------------------------------------
                 01590 *PRINT #-2,STRING$(ZN*2+ZW,ZG$)
                 01600 *-----------------------------------------
3EE5 F6  3E92    01610 HIR120  LDB     ZN
3EE8 58          01620         LSLB             ZN*2
3EE9 FB  3E93    01630         ADDB    ZW
3EEC 1F  01      01640         TFR     D,X
3EEE B6  3F49    01650 HIR125  LDA     ZG
3EF1 BD  3F36    01660         JSR     PRINT
3EF4 30  1F      01670         LEAX    -1,X
3EF6 26  F6      01680         BNE     HIR125
                 01690 *-----------------------------------------
                 01700 *NEXT ZB,ZC
                 01710 *-----------------------------------------
3EF8 B6  3F46    01720 HIR130  LDA     ZS       GET ZS
3EFB 34  02      01730         PSHS    A
3EFD B6  3F43    01740         LDA     ZB
3F00 AB  E0      01750         ADDA    ,S+      STEP ZS
3F02 B7  3F43    01760         STA     ZB
3F05 1020 FF17   01770         LBGE    HIR000   GO IF ZB>=0
3F09 7C  3F42    01780         INC     ZC
3F0C B6  3F42    01790         LDA     ZC
3F0F 81  20      01800         CMPA    #32      ZC=32?
3F11 1026 FFD6   01810         LBNE    HIR070   GO IF NO
                 01820 *-----------------------------------------
                 01830 *PRINT #-2
                 01840 *-----------------------------------------
3F15 86  0D      01850         LDA     #$D      CR
3F17 BD  3F36    01860         JSR     PRINT
                 01870 *-----------------------------------------
                 01880 *NEXT ZR
                 01890 *-----------------------------------------
3F1A B6  3F48    01900         LDA     ZT       GET ZT
3F1D 84  01      01910         ANDA    #1
3F1F BB  3F45    01920         ADDA    ZR
3F22 BB  03      01930         ADDA    #3
3F24 B7  3F45    01940         STA     ZR
3F27 7C  3F48    01950         INC     ZT
3F2A 81  C0      01960         CMPA    #192     AT END?
3F2C 1026 FEE7   01970         LBNE    HIR060   GO IF NO
                 01980 *-----------------------------------------
                 01990 *PRINT #-2,CHR$(13);
                 02000 *-----------------------------------------
3F30 86  1E      02010         LDA     #30      RESET GRAPHICS
3F32 BD  3F36    02020         JSR     PRINT
                 02030 *-----------------------------------------
                 02040 *RETURN
                 02050 *-----------------------------------------
3F35 39          02060         RTS
                 02070 *-----------------------------------------
3F36 C6  FE      02080 PRINT   LDB     #-2
3F38 D7  6F      02090         STB     $6F
3F3A AD  9F A002 02100         JSR     [$A002]
3F3E 4F          02110         CLRA
3F3F 97  9C      02120         STA     $9C
3F41 39          02130         RTS
                 02140 *-----------------------------------------
3F42       00    02150 ZC      FCB     0
3F43       00    02160 ZB      FCB     0
3F44       00    02170 ZN      FCB     0
3F45       00    02180 ZR      FCB     0
3F46       00    02190 ZS      FCB     0
3F47       00    02200 ZZ      FCB     0
3F48       00    02210 ZT      FCB     0
3F49       00    02220 ZG      FCB     0
           3E04  02230         END     START

00000 TOTAL ERRORS
```



Listing 3: MLPRINT

```
1ØØ 'HIGH-RES SCREEN PRINT TEST
11Ø CLEAR 1ØØ,&H3DFF
12Ø FOR I=&H3EØØ TO &H3F49
13Ø READ A: POKE I,A
14Ø NEXT I
15Ø DEFUSRØ=&H3EØ4
16Ø SCREEN 1,Ø
17Ø PMODE 4,1
18Ø PCLS
19Ø FOR I=1 TO 5
2ØØ CIRCLE (RND(255),RND(191)),
RND(1ØØ)
21Ø LINE (RND(255),RND(191))-(R
ND(255),RND(191)),PSET,B
22Ø NEXT I
23Ø A=USRØ(Ø)
24Ø GOTO 24Ø
25Ø '═══════════════════════
26Ø DATA &HE ,&HØ ,&H1 ,&HØ
27Ø DATA &HB6,&H3E,&H2 ,&H4Ø
28Ø DATA &HB7,&H3F,&H46,&H86
29Ø DATA &H12,&HBD,&H3F,&H36
3ØØ DATA &H4F,&HB7,&H3F,&H45
31Ø DATA &HB7,&H3F,&H48,&H4F
32Ø DATA &HB7,&H3F,&H42,&H86
33Ø DATA &H7 ,&HB7,&H3F,&H43
34Ø DATA &HF6,&H3F,&H43,&H4F
35Ø DATA &H1F,&H1 ,&HC6,&H1
36Ø DATA &H8C,&HØ ,&HØ ,&H27
37Ø DATA &H1Ø,&H58,&H3Ø,&H1F
38Ø DATA &H26,&HF9,&HB6,&H3E
39Ø DATA &H2 ,&H81,&H2 ,&H26
4ØØ DATA &H4 ,&H54,&HFA,&H3F
41Ø DATA &H47,&HF7,&H3F,&H47
42Ø DATA &H1Ø,&H8E,&HØ ,&HØ
43Ø DATA &H4F,&HA7,&HA9,&H3F
44Ø DATA &H49,&H1F,&H2Ø,&HF7
45Ø DATA &H3F,&H44,&HF6,&H3F
46Ø DATA &H45,&HFB,&H3F,&H44
47Ø DATA &H86,&H2Ø,&H3D,&HF3
48Ø DATA &H3E,&HØ ,&H34,&H6
49Ø DATA &H4F,&HF6,&H3F,&H42
5ØØ DATA &HE3,&HE1,&H1F,&H1
51Ø DATA &HA6,&H84,&HB4,&H3F
52Ø DATA &H47,&H27,&H6 ,&H86
53Ø DATA &H1 ,&HA7,&HA9,&H3F
54Ø DATA &H49,&H31,&H21,&H1F
55Ø DATA &H2Ø,&H1Ø,&H83,&HØ
56Ø DATA &H4 ,&H26,&HC9,&HB6
57Ø DATA &H3F,&H45,&H81,&HBD
58Ø DATA &H26,&H8 ,&H86,&HØ
59Ø DATA &H31,&H3F,&HA7,&HA9
6ØØ DATA &H3F,&H49,&HB6,&H3F
61Ø DATA &H48,&H84,&H1 ,&H26
62Ø DATA &H2B,&HB6,&H3F,&H4C
63Ø DATA &HC6,&H4Ø,&H3D,&H34
64Ø DATA &H4 ,&HB6,&H3F,&H4B
65Ø DATA &HC6,&H3Ø,&H3D,&HEB
66Ø DATA &HE4,&HE7,&HE4,&HB6
67Ø DATA &H3F,&H4A,&HC6,&HC
68Ø DATA &H3D,&HEB,&HE4,&HE7
69Ø DATA &HE4,&HB6,&H3F,&H49
7ØØ DATA &HC6,&H3 ,&H3D,&HEB
71Ø DATA &HEØ,&HCB,&H8Ø,&HF7
72Ø DATA &H3F,&H49,&H2Ø,&H29
73Ø DATA &HB6,&H3F,&H4C,&HC6
74Ø DATA &H6Ø,&H3D,&H34,&H4
75Ø DATA &HB6,&H3F,&H4B,&HC6
76Ø DATA &H18,&H3D,&HEB,&HE4
77Ø DATA &HE7,&HE4,&HB6,&H3F
78Ø DATA &H4A,&HC6,&H6 ,&H3D
79Ø DATA &HEB,&HE4,&HE7,&HE4
8ØØ DATA &HB6,&H3F,&H49,&HC6
81Ø DATA &H1 ,&H3D,&HEB,&HEØ
82Ø DATA &HCB,&H8Ø,&HF7,&H3F
83Ø DATA &H49,&HF6,&H3E,&H2
84Ø DATA &H58,&HFB,&H3E,&H3
85Ø DATA &H1F,&H1 ,&HB6,&H3F
86Ø DATA &H49,&HBD,&H3F,&H36
87Ø DATA &H3Ø,&H1F,&H26,&HF6
88Ø DATA &HB6,&H3F,&H46,&H34
89Ø DATA &H2 ,&HB6,&H3F,&H43
9ØØ DATA &HAB,&HEØ,&HB7,&H3F
91Ø DATA &H43,&H1Ø,&H2C,&HFF
92Ø DATA &H17,&H7C,&H3F,&H42
93Ø DATA &HB6,&H3F,&H42,&H81
94Ø DATA &H2Ø,&H1Ø,&H26,&HFF
95Ø DATA &H6 ,&H86,&HD ,&HBD
96Ø DATA &H3F,&H36,&HB6,&H3F
97Ø DATA &H48,&H84,&H1 ,&HBB
98Ø DATA &H3F,&H45,&H8B,&H3
99Ø DATA &HB7,&H3F,&H45,&H7C
1ØØØ DATA &H3F,&H48,&H81,&HCØ
1Ø1Ø DATA &H1Ø,&H26,&HFE,&HE7
1Ø2Ø DATA &H86,&H1E,&HBD,&H3F
1Ø3Ø DATA &H36,&H39,&HC6,&HFE
1Ø4Ø DATA &HD7,&H6F,&HAD,&H9F
1Ø5Ø DATA &HAØ,&H2 ,&H4F,&H97
1Ø6Ø DATA &H9C,&H39,&HØ ,&HØ
1Ø7Ø DATA &HØ ,&HØ ,&HØ ,&HØ
1Ø8Ø DATA &HØ ,&HØ
```

KISSable OS-9

Good Times with OS-9 on the Hard Disk

By Dale L. Puckett
Rainbow Contributing Editor

It's hard to believe it's almost fall — especially as I start to write this edition of KISSable OS-9 during the last week of May. Yet, it's almost harder to believe I have 576K of RAM in my Color Computer with another 256K waiting to be installed, while I sit here playing with the new 15-megabyte hard disk from Tandy. This month we'll share some ideas about organizing massive amounts of data on a hard disk, and work through several patches of randomness as we try to answer some of the questions you've posed during the past few months.

I try to take my marching orders from you — that's why we attempt to answer at least a few questions each month. If you have a question, please send it to us and we'll share it with everyone. And don't forget, if you have any good ideas or tips you would like to share with fellow OS-9 users send them to us here at "KISSable OS-9" and we'll sure try to get them in print. The same goes for short procedures and program listings. Some of the ideas you have sent in the past have been clever enough to impress a lot of veteran OS-9 programmers. Keep up the good work.

Fate is often unpredictable. Less than a week after Fran McGee loaned me one of the new Tandy 15-megabyte hard disk drives and a new hard disk controller cartridge, my old gray CoCo bit the dust. To solve the problem we rushed out and bought a CoCo 2 with a matching white Multi-Pak Interface and went to work.

Installation of the Tandy hard disk drive is very easy. To use it, you need a Color Computer with 64K of memory, a Multi-Pak Interface, at least one floppy disk drive and controller, the Color Computer Hard Disk Controller, Version 2.00.00 of the OS-9 operating system and the hard disk drive itself.

After making sure the Color Computer and Multi-Pak Interface are turned off, plug the Hard Disk Controller into slot 3 of the Multi-Pak Interface. This is the second slot from the back. The floppy disk controller stays in slot 4. The two controllers must be in their assigned slots because the OS-9 floppy and hard disk drivers have been programmed to find them there. If you change slots, the drives will not work.

After you have plugged in the cartridges, connect the 50-pin ribbon cable from the hard disk controller cartridge to the hard disk drive. The ribbon cable must exit toward the rear of the Multi-Pak Interface — or toward the side of the Hard Disk Controller with the label. After the cable is installed, turn on the Multi-Pak, the Color Computer and the floppy and hard disk drives, in that order.

When turning on the hard disk drive for the first time you will not find any information on it. It must be formatted before you can use it to store OS-9 programs and data. To do this, first load the /h0 device descriptor and the hard disk driver modules. Both are stored in the modules directory of the boot/config disk supplied with OS-9 Version 2.00.00. The descriptor is stored in a file named h0—15.dd and the driver is stored in CCHDisk.dr. After these modules are loaded use the standard OS-9 format utility command to initialize the hard disk drive.

Once you have formatted the hard disk, you will probably want to use the *Config* program to make a new OS-9 boot file that contains /h0 and CCHDisk. After you do this, OS-9 will boot from the floppy containing the OS-9 boot file, but it will start up with the current execution directory and current data directory set to /h0/cmds and /h0. It's amazing!

This drive is almost as fast as a RAM disk. After you install a hard disk drive, OS-9 becomes a whole new ball game. However, if you use a number of application programs on the Color Computer that need a lot of memory, be very careful about the number of modules you load into the OS-9 boot file. To use the C compiler, for example, you must strip OS-9 boot down to the bare minimum. You can't remove the floppy disk drivers, but you can get rid of the printer descriptor, /p, and the related driver, Printer, and a few other modules. If you use

---

Dale L. Puckett, who is author of The Official BASIC09 Tour Guide *and co-author, with Peter Dibble, of* The Official Rainbow Guide to OS-9, *is a free-lance writer and programmer. He serves as director-at-large of the OS-9 Users Group and is a member of the Computer Press Association. Dale works as a U.S. Coast Guard chief warrant officer and lives in Alexandria, Virginia.*

a RAM disk in addition to the hard disk you will really find memory space full — but it's worth it.

Your approach to computing will definitely change when you get a hard disk, and your enjoyment level will soar. You will eventually need to get organized, however. If you don't, you'll soon be lost in a sea of directories and subdirectories.

I was asked an unusual question at RAINBOWfest Chicago. "How should I set up my disk directories? Is it best to use a tall skinny directory structure or should I spread them out in a horizontal fashion?"

Organizing a disk is a personal matter but perhaps we can help with an overview of the possibilities available when using OS-9. To start, remember it was a tall skinny directory structure you were forced into when using Disk BASIC, CP/M, Flex or even the first several versions of MS-DOS on an IBM PC. Now remember the long lists you had to search when finding a stray file. It took forever, didn't it? Let's move forward and show how you can use OS-9's hierarchical file system to get organized.

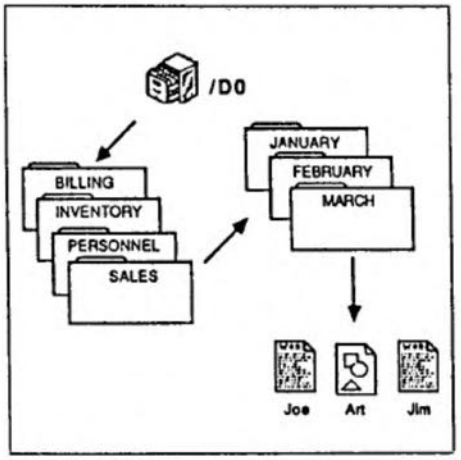
The most basic element in the OS-9 filing system is the individual file. Files most often contain data you are interested in — the text processing documents holding letters you have written or spreadsheets showing a profit and loss statement, for example. But, OS-9 files can also contain directories to tell the system how to find other information, or even the programs needed to manipulate information.

To get a handle on the OS-9 filing system, think of each mounted disk as a large filing cabinet. Inside that filing cabinet, each of the directories at the first level can be compared to a number of individual drawers. Other directories stored in these first level directories are called subdirectories. You can liken them to file folders. And finally, the individual files containing your data can be compared to the individual pieces of paper stored in a file folder.

The top level of the OS-9 filing system on any particular disk is the root directory of that disk. The directories stored in the root directory usually give a user access to application programs and other system data he may need. For example, the standard system disk with the OS-9 operating system from Tandy contains two files and three individual directories. The two files, OS-9 boot and startup, are used to start the system. The three directories hold programs and other information about the computer. The cmds directory contains all of the utility command programs to maintain data files. The DEFS file contains symbolic definitions of all key memory addresses used by the OS-9 operating system. And finally, the SYS directory contains information OS-9 uses occasionally. Included is a message of the day which is sent to every new user signing on the system and a password file that ensures only authorized users have access to the system.

On a multi-user, OS-9 based computer the system manager often sets up a directory for each user. These user directories are usually placed in the root directory of the disk. It is then up to the individual user to organize the data in his own directory. Since you are the user, the system manager's move places the ball squarely in your court. Let's look at one way to play the game.

First, we'll assume you don't have a hard disk. Set up directories to match the many jobs you have to do. For example, if you supervise a large staff, do the billing, track an inventory and keep an eye on the sales team, you will want to set up at least four directories in the root directory of your personal disk. The first few levels of the filing system might look something like this:



In the directory named Billing, you could create two sub-directories or folders — Sent and Received. In Inventory you might want to set up folders for Completed Widgets and Spare Parts. In the Personnel directory you will need at least two folders — one for Evaluations and another for Payroll. And finally, in the Sales folder you can create 12 folders or sub-directories, one for each month. Each of these folders would hold files containing each sales person's report for the month plus any charts or graphics needed to make a clear report to the boss.

If you are the only user and set up a filing system on your own disk, the structure of the disk would be similar to that in the figure above. If you are working in an office with two other managers and using a hard disk for storage, the system manager will most likely have created three directories — one for each of you in the root directory of the hard disk. In that case, simply move your top directory level to the hard disk from your floppy disk.

If your name is Fred and one of the three user directories set up by the system manager is named Fred, then the pathlist to your Billing folder would become /h0/fred/billing.

And, the complete pathlist to Jim's sales report for January would be:

/h0/fred/sales/january/jim.

It is easy to find a particular file after setting up a logical filing system similar to the one above. For example, if you need to check out Joe's last personnel evaulation, you need to look in a file with a pathlist like this:

/d0/fred/personnel/evaluations/
february/sam

It's easy to find a file when it is stored in a logical place. Typing a long pathlist like this can get old fast, but since you most likely work on all of your personnel reports at the same time, you can take advantage of one of OS-9's handy features and set the current data directory to the current month's reports with this command line:

OS9: chd /d0/fred/personnel/
evaluations/feb

Then, all you need to type is:

```
OS9: list sam
```

The first command line above sets the current data directory to:

/d0/fred/personnel/evaluations/feb.

All of these directories contain files holding data. And since you know your business better than anyone else, OS-9 lets you organize data directories the way you have organized your business. After this, you will be able to find files quickly. And, once your disk is organized, OS-9's chd command makes it easy to change the current data directory to any particular set of files.

OS-9 Helps Organize Programs Too

OS-9 files can also contain programs; its designers moved one up on UNIX and added a second current directory. This second working directory is called the current execution directory. It is used to hold files containing 6809 object code and intermediate code from one of the many OS-9 languages that can be run on your computer.

When booting OS-9 on the Color Computer on a floppy disk system, a program called *sysgo* is executed automatically and one of the first things it does is set up your current directories. After *sysgo* runs, the current execution directory will be /d0/cmds and your current data directory will be /d0.

The "current" directories apply only to the disk mounted in the drive /d0 when you boot the system. If you remove that disk and insert another, the system records are no longer "current." You must use the OS-9 chd (change current data directory) and chx (change current execution directory) utility commands before trying to do any work with the new disk. If you don't do this, the system gets lost because it will look for the directories on the new disk at the same location where it found them on the old disk.

When it doesn't find them there and loads something inappropriate into memory, strange things often happen.

By the way, if you own a Tandy hard disk and have installed its device descriptor, /h0, and its device driver, CCHDISK, in your OS-9 boot file, the *sysgo* program in OS-9 Version 2.00.00 automatically sets the current execution directory to /h0/cmds and the current data directory to /h0 each time you start the system.

If you decide to run a program, *Dir* for example, OS-9 looks for *Dir* and then runs it. However, before OS-9 looks on the disk drives, it checks to see if the program is already in memory. To do this it looks for the name typed on the command line in its module directory. If OS-9 finds *Dir* in its module directory, it links to it and runs it immediately. No disk access is needed.

But what happens if *Dir* is not in memory? OS-9 looks in the current execution directory and tries to find a file named *Dir*. If it finds a file with the right name in this directory, it assumes it is executable code, loads it into memory and runs it.

And finally, if OS-9 doesn't find *Dir*, in the current execution directory it makes one more try — this time in the current data directory. But, if OS-9 finds *Dir* in the current data directory it won't treat it like a program. It treats it like a data file. More specifically, it assumes this data file contains a procedure file.

An OS-9 procedure file is similar to a UNIX script file. A procedure file contains a list of OS-9 commands which are read into the shell. Each time a command line is read, it is run just as if you had typed it. The process continues until OS-9 receives an end of file signal from the procedure file.

In the case of the intermediate code from OS-9 languages, execution of the language's run time package is also automatic. Packed BASIC09 programs, for example, are executed by a run time interpreter named Runb. When you type the name of a file stored in the current execution directory containing packed BASIC09 code, OS-9 loads this i-code into memory just like it were 6809 object code.

However, before OS-9 runs the code in any module it checks the information in the module header to find out what type of code is in the module. When it finds out you have loaded packed BASIC09 i-code, it knows that Runb is needed to run the program. So, OS-9 automatically loads Runb and executes it with the name of the module as a parameter. All of this work is transparent and all you see on the screen is the output of the BASIC09 program.

When you first purchase Tandy's version of OS-9 there are enough utility command files stored in the directory /d0/cmds on the OS-9 system master to fill several screen pages when you list the filenames. After you have added a few dozen of your own favorite application programs and third party utility command packages, it becomes almost impossible to find a file in a directory listing on the screen. The problem is complicated by the fact that the *Dir* utility command in the 6809 version of OS-9 does not alphabetize the directory listing for you. The new 68K version of OS-9 lists the contents of the directories in alphabetical order.

So, if you use a hard disk and own hundreds of programs, you need to organize a set of directories on the hard disk using a method similar to the one for organizing your data directories.

For example, to keep current I try to take a look at most all OS-9 software. Many of the third party utility packages contain similar programs and more than one developer may use the same name. The standard UNIX-like utilities, *ls* and *mv* are good examples.

But, there's a hitch. While all of these utilities may have the same name, they require a different syntax on the command line. Also, you can't have more than one program stored with the same filename in the same directory.

While I was exercising the Tandy 15 megabyte hard disk, I decided to organize my utility programs. I did this by creating subdirectories in the current execution directory, /h0/cmds. For typing ease I used two or three letter names for the directories to store programs and utilities from the various third party vendors. Here is a look at the program side of my filing system.

```
/H0/CMDS
CW DPJ FHL MW  RS SG
```

Computerware's utilities are stored in the directory, CW; D. P. Johnson's hackers kits and utility packages live in DPJ; products from Frank Hogg Laboratory are stored in FHL; Microware's toolkit is saved in MW; Tandy products live in RS and finally Steve Goldberg's *Utilipak* programs are run from SG.

Using the standard OS-9 shell from the Tandy release of OS-9, I must type the complete pathlist to a utility in one of these directories. That's why I used the short directory names. For example, if I want to run Steve Goldberg's version of *ls*, I merely type:

```
OS9: sg/ls
```

If I wanted to use Brian Lantz's verison of *ls* from Computerware, I would have typed:

```
OS9: cw/ls
```

But, by using the new kshell in the *Advanced Utilities* package from Computerware, I make this operation automatic, going as far as to tell OS-9 which vendors command I want to have priority. To do this I use the new path command built into kshell.

```
OS9: path=cw:sg:mw:dpj:fhl:rs
```

After typing this command line, I receive very few "Error #216 Program Not Found" errors. The kshell first searches /h0/cmds for the filename typed. If it doesn't find it there, it looks in the subdirectories, CW, SG, MW, DPJ, FHL and RS in that order. It runs the first program stored in a file with the name I have typed. In other words, if the CW subdirectory contains an *ls* command, the *ls* in the *DPJ* directory will never be run as long as this path definition is active. However, I could always select it manually and override the default path by typing:

```
OS9: dpj/ls
```

While getting the hard disk organized you will sometimes move a lot of files back and forth from one directory to another. The *DirCopy* utility from Computerware's *Disk Fix and Utilities* package is almost indispensable during this process. Use a command like this:

```
OS9: dircopy /H0/FIRST_DIREC
  TORY /H0/SECOND_DIRECTORY
```

DirCopy can be told to let you confirm each copy command. This means you can select the files you want to copy from one directory to another on the fly. You can also tell the program to copy files stored in subdirectories while it is copying a directory, and this feature is really handy — tell it to place the files in the new directory in alphabetical order while it is being created.

You can also tell *Dircopy* to automatically rewrite any file in the destination directory with the same name as a file in the source directory. However, if you don't feel safe with this approach, have *Dircopy* ask you each time it discovers this problem.

Ask 100 people how they organize their hard disk and you'll get 100 different answers. But there are some basics you should consider. To get in the mood, study these approaches from two members of the OS-9 SIG.

```
1.  /H0/LANGUAGES/BASIC09
   /SOURCE/INVENTORY/PROGRAMS
2.  /H0/INVENTORY/PROGRAMS
   /SOURCE/BASIC09
```

What is more important — the job or the program that runs the program that does the job? The first example emphasizes the computer instead of the job. The latter takes the opposite approach. I really can't say it any better than Kevin Darling who is one of the most active and knowledgeable individuals on the OS-9 SIG.

"The fact that a file is 'inventory related' is much more important to me than the language," he said. In other words, some nouns are more important to me than their adjectives."

Darling also posed a very relevant question. "Could someone other than you find things quickly when everything is stored relative to the language it is written in? I would hate to go looking all over my different language subdirectories, just to find one of my inventory programs."

Other good ideas came from Pete Lyall

and Jonathan Cluts who help SysOp Wayne Day manage the SIG. "I think we need to standardize and use a number of common directories like the UNIX crowd," Lyall said. "If we do this, then program authors can make certain assumptions about directories in this set."

Lyall's proposed directory set includes cmds, defs, etc, help, lib, src, sys, tmp and user. We are already familiar with cmds, defs and sys; they come predefined with OS-9. The purpose of the help directory is obvious, as is src. User would be used for all logins and would contain subdirectories for individual users. Etc would be a directory where you could store OS-9 procedure files. Library files obviously would be stored in the lib directory. This might also be a good place to put dictionaries and other common files. And finally, tmp would be used to store temporary work files.

Cluts' idea has to do with starting the system. "I run *tsmon* at startup so I always log in. By logging in as different users, I can restrict myself to certain directories on the hard disk. This means I can only accidentally destroy some of my files. If I stay in the root directory I could destroy them all," he said.

On the same evening Cornelius Seon, who frequents RAINBOW's CoCo SIG on Delphi added the most relevant comment about his new Tandy hard disk. "Several packages which were only marginally useful on floppies are just wonderful now," he said. I agree.

But before closing, he left everyone with a problem. "There is a problem with Mother Tandy's Godfrey Daniels' helpfulness," he said. "I'm finding many programs they either wrote or commissioned looking for data on floppy drive /d0 after I've already transferred everything to /h0 using identical directory names."

Maybe we can help here. The problem probably isn't Tandy's at all. It relates more to the fast pace at which hardware technology is racing past the software. When most of the guilty programs were written, hard disks didn't even exist. If early software designers had made one change, programs would have moved right over from /d0 to /h0 automatically. We print that change here so anyone developing software now can avoid our earlier mistake.

Most of the programs causing a problem contain a pathlist to a device and directory. *DeskMate* is a good example. *DynaSpell* is another. For example, when *DynaSpell* needs the dictionary, it looks for /d0/spell/dictionary.dat. The future solution is for the programmers to take advantage of OS-9's anonymous filenames. In other words, instead of coding a pathlist named "/d0/spell/common.dat," I should have made that pathlist read,". . ./spell/common.dat."

The first period tells OS-9 to look for the file in the current data directory, two periods means look at the parent directory of the current data directory and three periods means look in the parent of the parent of the current data directory. Is that the grandparent directory? By including five or six of these periods, I could have forced OS-9 to look clear back in the root directory. Then it wouldn't have mattered whether I had the file stored on /d0 or /h0, because it would be anonymous. I guess we all learn sometime — better late than never!

"It is also now possible to give the utilities an optional command line argument list rather than use standard input."

The fix for programs already written is to change the 'D' in the offending pathlists to an 'H' or to change the "/d0" to ". . ." Sounds easy enough, but how do you go about it?

There are several approaches. First, in many of the programs the strings defining the pathlist are located early in the program and you can locate them easily with the OS-9 dump utility. For example if you need to find these strings in *DynaSpell*, you would type:

```
OS9: dump spell
```

In the first page of the Hex dump you will see the strings and can jot down the number of bytes that each 'D' is offset from the beginning of the module. After you do this, load the program:

```
OS9: load spell
```

Then, call up the OS-9 *Debug* utility and link to spell.

```
OS9: debug
db: l spell
```

Now, move the *Debug* pointer to the first 'D' and use the '=' command to change it to an 'H'. After you change all the /d0s to /h0s and quit *Debug* with its Q command, you can save the file into a tempory file.

```
OS9: save tempspell spell
```

You're almost home free now. Rename or delete the original spell file and run this command.

```
OS9: verify<tempspell>spell u
```

Don't forget the 'u', it tells the OS-9 verify utility command to update the CRC of the module in the file you are verifying. After you have done this you can load or run the new *spell* file. Even though we used the *DynaSpell* program as an example here, keep in mind this same technique can be used with any program you need to modify for operation with a hard disk.

If you purchased Computerware's *Disk Fix and Utilities*, you will find it much easier to use the *Patch* utility from that package. For example, to make this change you first type:

```
OS9: patch spell
```

Then uses *Patch's* 'D' — for display — command to find the strings defining the bothersome pathlists.

```
>D 0000
```

Again, jot down the location of the offending 'D' and type:

```
>M 00XX
```

The XX would be replaced by the offset that you jotted down above. At this point when you see patch display the 'D' (as 44, by the way) you can type:

```
>'H
>/
```

Go through this sequence for each of the pathlists that contain /d0. Then, type:

```
>V
```

The V command tells *Patch* to validate the crc of the file you have patched and correct it. After you have done this, you can exit *Patch* with its Q command.

Sometimes the strings containing the floppy based pathlists aren't stored in the first page. In this case *Patch* is really a help because it gives you a find command. After starting *Patch* you can type:

```
>F'D0
```

After you press the ENTER key, *Patch* prints a list of every occurrence of that string in the program you are adapting to the hard disk. After you have the list of addresses, you must use the M command again to change the 'D' to an 'H' in each case. One note of caution. Take a close look at each 'd0' you find and make sure that it is, in fact, part of an actual pathlist. It could just be a natural occurrence in the program's code itself.

Dan Johnson Markets Enhanced Package

SDisk author Dan Johnson (7655 SW Cedarcrest St., Portland, OR 97223, 503-244-8152) has upgraded all of his OS-9 filter

and hacker's kits and placed them in one package called the *LI Utility Pak*. Don't let the name fool you; all but three of the programs in this package run on both Level I and Level II OS-9 systems. These three deal with absolute memory manipulations.

Johnson hopes to release a supplement Level II utility package sometime later. I bet he'll have an incentive when Tandy's new Color Computer hits the market. He has also added several new programs to the package including *Modbuster, Afmt, Upall, Grep* and *Sectedit*.

Dan has made major improvements to the template matching algorithm in his new *ls* command and most of the utilities now accept options at the beginning or end of a command line. It is also now possible to give the utilities an optional command line argument list rather than use standard input. The *MacGen* program has been expanded and Johnson gives the *MacGen* source for a useful program, *UpAll*. The package sells for $49.95 and is shipped on a CoCo formatted OS-9 disk. However, you can buy the package in other formats by paying an additional $2. Johnson calls it a deal as good as *Turbo* PASCAL on MS-DOS.

Two More Goldberg Tips

Earlier this year, Steve Goldberg discovered that the modification date was changed every time he changed directories while using OS-9 Version 2.00.00. To solve the problem, he uses the following procedure file on the shell module and builds a new boot file with the patched shell.

```
debug
l <SPACEBAR> shell
. <SPACEBAR> .+269
=81
. <SPACEBAR> .+28D
=E9
=49
=67
q
cobbler /d0
```

Goldberg noted that since he has run this procedure his directory changes are faster and the drives don't grind while attempting to write the date to a write protected disk when he changes directories.

I thought we published the changes to CCDisk which let you step your disk drives at a faster rate earlier this year. But I couldn't find them, and we have been seeing a lot of traffic on the Delphi CoCo SIG and getting a few requests in the mail, so we'll give it another try. This time, thanks to Steve Goldberg's homework, we'll compare them to the identical changes you had to make with OS-9 Version 1.00 and 1.01. The new values change the step rate of your drives to six milliseconds. See *The Complete Rainbow Guide to OS-9* for the value for other step rates.

CCDISK OFFSETS

Version 1.0	Version 2.0	New Value
$1FE	$1F9	$10
$204	$1FF	$08
$205	$200	$8B
$2DD	$2DD	$40
$2E9	$2E8	$00

Goldberg is brave and pushed the value of the word at $1FF down to $0001. I don't think I would push it that far. He also cut the drive startup delay since his drives seemed to be fast. To do this he changed the $A0 at an offset of $2F9 to $0A.

BASIC09 Beginners Notes

Henry J. Proffitt of Haley Station, Ontario wrote to ask how he could enter a short BASIC09 program he had stored in a standard OS-9 text file into BASIC09. Proffitt entered the program with the *Build* utility, but you could use any OS-9 editor including *TSEdit* or the *DeskMate* editor to enter a BASIC09 program.

The thing you must remember is that BASIC09 expects to see the word "procedure" as the first thing in a file it is trying to load. Therefore, if you type the line `procedure myprogram` as the first line in your file, BASIC09 will be able to load it. Incidentally, the 'p' in procedure must be the very first character in the file.

After you have created the text file following the directions above, you can load it from the BASIC09 system mode — you'll know you are there because BASIC09 prompts with "B:". Just type:

```
B: load myprogram
```

BASIC09 assumes there is a file named "myprogram" stored in your current data directory and that the file contains a BASIC09 procedure. If you have not saved your program there, use the CHD command from within BASIC09 to set the current data directory or type a complete pathlist:

```
B: load /D1/MYDIRECTORY/
   MYPROGRAM
```

We received a note from Robert Gault of Grosse Pointe Woods, Michigan about a solution to another BASIC09 programming problem you may have noticed while working with BASIC09. How do you send a character with the value $FF to the screen or printer? This is a problem that must be solved if you plan on sending graphics to the printer.

There are two solutions. You can use the BASIC09 shell command and use OS-9 to send out the character or you can use a PUT statement instead of a PRINT statement.

```
SHELL "display FF"
SHELL "display FF >/p"
PUT #PRINTER, $FF
```

Gault submitted a demonstation procedure and a rough and ready screen dump program to illustrate his techniques.

Next Month

Next month we'll dig back in the mailbox and see what's been bothering you the most. Hopefully, we can come up with a tip or two. And, we'll cross our fingers and hope we get to play with a new OS-9 Level II Color Computer soon! 'Til then, keep on hacking! □

Listing 1: ***strings***

```
PROCEDURE strings
DIM ss:STRING[256]
DIM i,memory:INTEGER

memory:=ADDR(ss)

(* Now fill memory with $FF's *)
FOR i=memory TO memory+255
POKE i,255
NEXT i

(* Note that BASIC09 thinks ss is an empty string *)

PRINT "The length of string ss ="; LEN(ss)
PRINT "ss =("; ss; ")"
PRINT "Now we will PUT ss.  Note the difference!"
PUT #1,ss
(* Use these lines to send ss to a printer, then *)
(* OPEN #printer, "/p" *)
```

Listing 2: ***printer***

```
PROCEDURE printer
(* This procedure is a rough and ready graphics screen dump *)
(* It assumes a preceding graphics program has used         *)
(* GFX("mode",format,color)                                 *)
(* If your printer uses an eight pin graphics format change *)
(* the "pin" loop to 0 to 7.                                *(

DIM dev:BYTE
DIM color,x,y,pin,send,location:INTEGER
DIM code:BYTE

OPEN #dev,"/p":WRITE
(* Now set printer to graphics mode *)
PRINT #dev,CHR$(18)
location:=ADDR(code)

FOR y=191 TO 7 STEP -7
FOR x=0 TO 255

(* tell printer code is graphics *)

send:=128

FOR pin=0 TO 6
(* Test Pixel *)
RUN gfx("gcolr",x,y-pin,color)
(* Printer used has top pin 2^0 bottom pin 2^6  *)
send:=LOR(send,MOD(color,2)*2**pin)
NEXT pin
POKE location+x,send
NEXT x
PUT #dev,code
(* Send carriage return *)
PRINT #dev,CHR$($0D)
NEXT y
```

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 part 2. 32K GAMES.

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

Best of CoCoOz #3. UTILITIES.

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

Best of CoCoOz #4. BUSINESS

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

Next:
Best of CoCoOz #5. Adventure Games
Best of CoCoOz #6. Preschool Education

PRICES:– TAPE $10/DISK $16 EACH

BEST OF COCO OZ 7

LIL'COCO	THE ROOM	BACK ST
MARTHA	BAD MOON	MCC
LOCO	COCO ART	KANGA
THE BOAT	SAD COCO	TOWER
WINDYDAY	SAILING	OUTHOUSE
EAGLE	BLASTER	FOGHORN
SMURF	SUNSTATE	HELICOPT

GRAFIX

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	$ 9.50 .....
CoCoOZ on Disk	12 mnths	$102.50 .....
	6 mnths	$ 58.00 .....
	1 mnth	$ 10.95 .....
MiCoOZ on Tape	12 mnths	$ 75.00 .....
	6 mnths	$ 42.00 .....
	1 mnth	$ 9.50 .....
RAINBOW ON TAPE (AUST/US)	12 mnths	$144.00 .....
	6 mnths	$ 81.00 .....
	1 mnth	$ 15.00 .....
RAINBOW ON DISK (AUST/US)	12 mnths	$172.00 .....
	6 mnths	$ 96.75 .....
	1 mnth	$ 15.00 .....

Or charge my credit card monthly TAPE/DISK ONLY

CoCoOZ on Tape	$ 9.50 .....
CoCoOZ on Disk	$ 10.95 .....
MiCoOZ on Tape	$ 9.50 .....
Rainbow on Tape (AUST/U.S)	$ 15.00 .....
Rainbow on Disk (AUST/U.S)	$ 15.00 .....

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

Credit Card No.: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Authorised amount $ .....

Signed : .....

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 quick reference guide for CoCo users. BYTE: Guide for new CoCo users. MiCo HELP: A quick reference for owners of MC 10 computers.		$9.95 $4.00 $9.95
Say the Wordz: by Oz Wiz & Pixel Software Two curriculum based speller programs for your Tandy Speech/Sound Pack.	Tape 32K ECB	$29.95
Bric a Brac: Blank tapes . . . 12 for $18.00 or $1.70 each. Cassette cases . . . . . 12 for $3.50 Disks . . . (they work!) . . $2.50 each or $25.00 per box of 10.		

HOW TO ORDER

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

User Group Contacts

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

ACT:

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

NSW:

SYDNEY:

BANKSTOWN	CARL STERN	02 646 3619
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSKO MCKAY	02 624 3353
CHATSWOOD	BILL O'DONNELL	02 419 6081
CLOYTON	HERMAN FREDRICKSON	02 6236379
FAIRFIELD	ARTH PITTARD	02 72 2881
GLADESVILLE	MARK ROTHWELL	02 817 4627
HILLS DIST	ARTHUR SLADE	02 622 8940
HORNSBY	ATHALIE SMART	02 848 8830
KENTHURST	TOM STUART	02 654 2178
LEICHHARDT	STEVEN CHICOS	02 560 6207
or	GORGE ECHEGARAY	02 560 9664
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACQUARIE FIELDS	BARRY DARNTON	02 618 1909
ROSEVILLE	KEN UZZELL	02 467 1619
SUTHERLAND	IAN ANNABEL	02 528 3391
SYDNEY EAST	JACKY COCKINOS	02 344 9111

ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	DOUG BARBER	067 72 7647
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BROKEN HILL	TERRY NOONAN	080 88 2382
CAMDEN	KEVIN WINTERS	046.66.8068
COFFS HARBOUR	BOB KENNY	066 51 2205
COOMA	ROSS PRATT	0648 23 065
COORANBONG	GEORGE SAVAGE	049 77 1054
COOTAMUNDRA	CHERYL WILLIS	069 42 2264
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 2095
FORBES	JOHANNA VAGG	068 52 2943
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	PETER LINDSAY	066 42 2503
GUYRA	MICHAEL J. HARTMANN	067 79 7547
JUNEE	PAUL MALONEY	069 24 1860
KEMPSEY	RICK FULLER	065-62-7222
LEETON	BRETT WALLACE	069-53-2081
LISMORE	ROB HILLARD	066 24 3089
LITHGOW	DAVID BERGER	063 52 2282
MAITLAND	BILL SNOW	049 66 2557
MOREE	ALF BATE	067 52 2465
MUDGEE	BRIAN STONE	063-72-1958
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 2095
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
ORANGE	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PORT MACQUARIE	RON LALOR	065 83 8223
SPRINGWOOD	DAVID SEAMONS	047 51 2107
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
URALLA	FRANK MUDFORD	067 78 4391
WAGGA WAGGA	CES JENKINSON	069 25 2263
WYONG	JOHN WALLACE	043 90 0312

NT:

DARWIN	BRENTON PRIOR	089.81.7766

QLD:

BRISBANE:

BIRKDALE	COLIN NORTH	07 824 2128
BRASSALL	BOB UNSWORTH	07 201 8659
EAST	ROB THOMPSON	07 848 5512
IPSWICH	MICK MURPHY	07 27 1777
NORTH	JACK FRICKER	07 262 8869
PINE RIVERS	BARRY CLARKE	07 204 2806
SOUTH WEST	BOB DEVRIES	07 375 3161
SANDGATE	MARK HIGHELL	07 269 5090
SCARBOROUGH	PETER MAY	07 203 6723

AIRLIE BEACH	GLEN EVANS	079 46 1264
BIGGENDEN	ALAN MENHAM	071 27 1272
BLACKWATER	ANNIE MEIJER	079.82.6931
BOWEN	TERRY COTTON C/O	077 86 2220
BUNDABERG	RON SIMPKIN C/O	TANDY
CAIRNS	GLEN HODGES	070 54 6583
DALBY	ANDREW B. SIMPSON	074.62.3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 51 0015
HERVEY BAY	LESLEY HORWOOD	071 22 4989
MACKAY	LEN MALONEY	079511333x782
MARYBOROUGH	JOHN EFFER	071 21 2651
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280
MURGON	PETER ANGEL	071 68 1628
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
TARA	STEVEN YOUNGBERRY	
TOOWOOMBA	GRAHAM BURGESS	076 30 4254
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
WHITEROCK	GLEN HODGES	070 54 6583

SA:

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

TAS:

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

VIC:

MELBOURNE:

MELBOURNE CCC	JOY WALLACE	03 277 5182
DANDENONG	DAVID HORROCKS	03 793 5157
DONCASTER	JUSTIN LIPTON	03 857 5149
FRANKSTON	BOB HAYTER	03.783.9748
NARRE WARREN	LEIGH EAMES	03 704 6680
NTH EASTERN	PETER WOOD	03 435 2018
MELTON	MARIO GERADA	03 743 1323
RINGWOOD	IVOR DAVIES	03 758 4496
SUNBURY	JACK SMIT	03.744.1355

BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
CHURCHILL	GEOFF SPOWART	051 22 1389
DAYLESFORD	DANNY HEDJI	054 24 8329
EMERALD	LEIGH EAMES	059 68 3392
GEELONG	DAVID COLLEN	052 43 2128
HASTINGS	MICHEAL MONCK	059 79 2879
MAFFRA	MAX HUCKERBY	051 45 4315
MOE	JIMMY WELSH	051 27 6984
MORWELL	GEORGE FRANCIS	051 34 5175
SALE	BRYAN McHUGH	051 44 4792
SHEPPARTON	ROSS FARRAR	058 25 1007
SMYTHESDALE	TONY PATTERSON	053 42 8815
SWAN HILL	BARRIE GERRAND	050.32.2838
TONGALA	TONY HILLIS	058 59 2251
TRARALGON	MORRIS GRADY	051 66 1331
WONTHAGGI	LOIS O'MEARA	056 72 1593
YARRAWONGA	KEN SPONG	057 44 1488

WA:

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

CANADA - CoCo & MODEL 100:

Ontario	Richard Hobson	416 293 2346

BUSINESS:

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

OS9 GROUPS:

NATIONAL OS9 USERS' GROUP	GRAEME NICHOLS	02 451 2954

NSW

SYDNEY

BANKSTOWN	CARL STERN	02 646 3619
CARLINGFORD	ROSKO MCKAY	02 624 3353
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY EAST	JACKY COCKINOS	02.344.9111
COOMA	FRED BISSELING	0648 23263

QLD

BRISBANE	JACK FRICKER	07 262 8869

VIC

LATROBE VLY	GEORGE FRANCIS	051 34 5175

WA

KALGOORLIE	TERRY BURNETT	090.21.5212

MC-10 GROUPS:

GOLD COAST	JIM ROGERS	075 91 4239
LITHGOW	DAVID BERGER	063 52 2282
ORANGE	DAVID KEMP	063 62 2270
PORT LINCOLN	BILL BOARDMAN	086 82 2385
ROCKHAMPTON	TIM SHANK	079 28 1846
SYDNEY	RAJA VIJAY	02 519 4106
WARRNAMBOOL	GARY FURR	055 62 7440

TANDY 1000 / MS DOS:

QLD:

BRISBANE NORTH	BRIAN DOUGAN	07 30 2072
BRISBANE SOUTH	BARRY CAWLEY	07 390 7946
GOLD COAST	GRAHAM MORPHETT	075 51 0015
MARYBOUOUGH	NORM GREENAWAY	

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

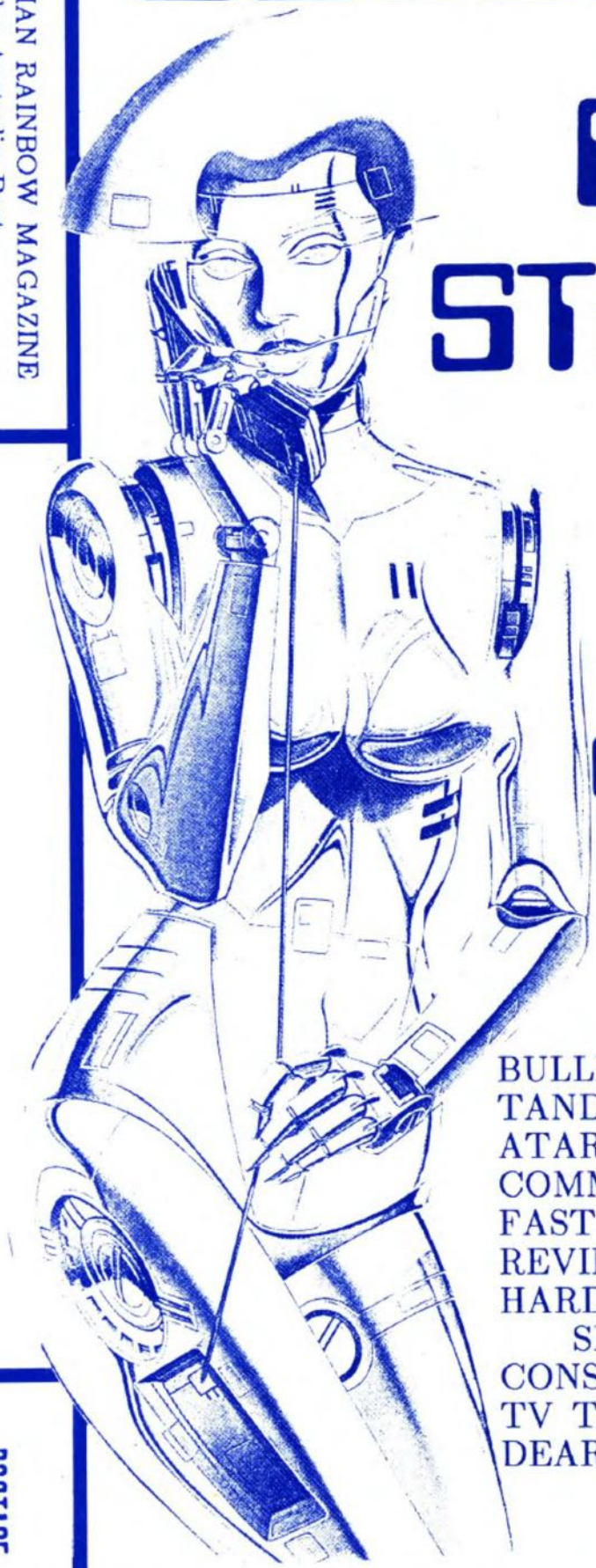
VIATEL:

GOLDLINK	*642#
VTX 4000	03 329 2936

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.

GOLDLINK

COM. STATION 692 ON VIATEL



BULLETIN BOARDS
TANDY COMPUTERS
ATARI COMPUTERS
COMMODORE COMPUTERS
FAST FLORIST
REVIEWS
HARDWARE & SOFTWARE SHOP
CONSUMER WATCH
TV TALK
DEAR MARTHA

YES WE EVEN TALK BACK !!