September 1983 $2.95

RAINBOW®

THE COLOR COMPUTER MONTHLY MAGAZINE

...CUS ON ...UCATION

...In The Classroom
...hers' Helpers
...k Talk For Parents

Co... Break...

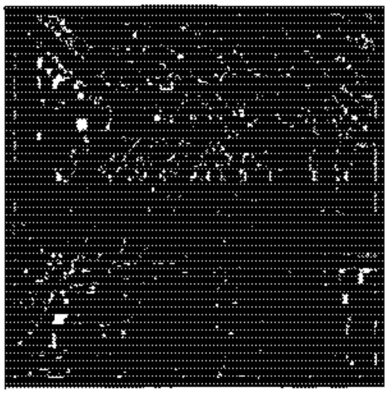
Two New Color Computers
And, OS-9

...erpatch For EDTASM
...DATA Statement Rag
...catenating In Hi-Res
...RAINBUG Takes Wing

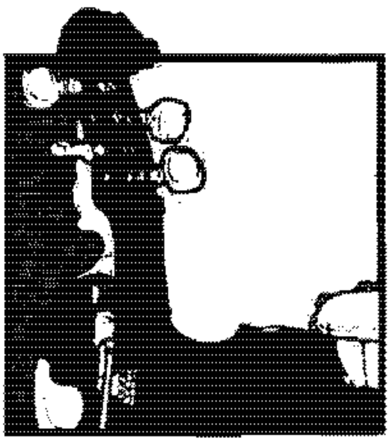
More Than 30 Product Reviews

Under the Rainbow

Feature Articles



206



218

252

COVER art © by Fred Crawford

THIS MONTH'S PIPELINE, PAGE 174, TAKES A LOOK AT RADIO SHACK'S TWO NEW COLOR COMPUTERS

Departments

Bob Albrecht's GameMaster's Apprentice will return next month.

Product Reviews

NEXT MONTH: Our graphics issue! Right in time with autumn, nature's most colorful season, we'll emphasize the color in our Color Computers. One graphic special is a flag-waving treat, and others provide you a Halloween trick or two, as well.

October also brings a new column on OS-9 as Dale Puckett, a leading authority on this operating system, joins *the Rainbow's* staff of contributing editors.

Plus . . . dozens of articles, dozens of features and dozens of hardware and software reviews — more information on the Color Computer than is available anywhere else.

Make October even brighter. Look for *the Rainbow!*

The Rainbow

Editor
Lawrence C. Falk

Managing Editor James E. Reed
Senior Editor [illegible]
Technical Editor Dan Downard
Copy Editor [illegible]
Submissions Editor [illegible]
Editorial Assistants [illegible]
Contributing Editors Bob Albrecht, Steve Blyn, Tony DiStefano, Don Inman, Joseph Kolar, Dennis Lewandowski, Bill Nolan, Dale Peterson, Michael Plog, Ph.D., Charles Roslund, Dick White

Art Director Sally Nichols
Assistant Art Director [illegible]
Designer Peggy Henry

Advertising Manager Charlotte Ford

General Manager Patricia H. Hirsch
Assistant General Manager for Finance [illegible]
Billing Clerk [illegible]
Customer Service Manager [illegible]
Research Assistants [illegible]
Dispatch [illegible]
Transportation [illegible]

[illegible]

letters to RAINBOW

UNDER THE RAINBOW

Editor:

Not that I've run into a CoCo Club or anything, but thought you might like this picture. I'm sitting behind the ponderosa pine with my pole in the water and reading my *Rainbow*.

Don Fruhwald
Cincinnati, OH

COCO CUTIE

Editor:

My 20-

ded a new word to her spoken vocabulary—" 'puter." With all the computer owners around, I can't help but wonder how many other toddlers have that word as part of their vocabulary. For about six months now she has been using a musical keyboard program, seldom pounding the keyboard. Her favorite key is, naturally, the red BREAK key which is the highest note.

Claire Miller
Rocky Point, NY

MORE NON-EXTENDED

Editor:

Let's see more non-Extended programs in your magazine.

The record idea is good but if it will only have Extended BASIC programs I can't use it. Yes, Virginia and Lonnie, there are some non-Extended CoCo owners in this world.

I have not seen a listing of the quantities of the different CoCos sold by Radio Shack since they were introduced. How many 4Ks, 16Ks Extended and non-Extended, etc.

If this information is not available to you, why not have a poll from your readers?

William M. Vance
Crockett, TX

Editor's Note: We recognize the need for non-Extended Color BASIC programs and encourage our readers to submit such programs to *the Rainbow*. For further details read PRINT #-2.

KUDOS

Editor:

I enjoyed the article and program in the June issue on solving cryptograms by Chris Reid. I typed in the programs and when it did not work I tried everything I knew. Then I sent a letter to Mr. Reid. He was very helpful in the reply he sent. I want to commend Mr. Reid for his patience (two letters offering suggestions) and help.

The entire *Rainbow* magazine is excellent, keep up the good work.

Richard Urich
Sellersville, PA

Editor:

I love your magazine! When I first subscribed in March, I thought the subscription rate a little high. But it was worth every penny. Compared to the other magazines, this magazine is the greatest! I especially like all of your assembly language articles and tutorials. Keep it coming.

Mike Brenneman
Leola, PA

Editor:

As a gift for Father's Day my three children gave me $22 to buy a subscription to *Rainbow* magazine. Their reasons were that they saw me drive 26 miles three times a month just to insure getting *one* of the three magazines *(Rainbow)* the bookstore would get. Can you believe that? Three magazines for a town of over 59,000 population.

You have a good thing going—please don't stop. Your magazine is worth twice the price.

Rick Maurer
Rantoul, IL

HINTS AND TIPS

Editor:

Phillip Beistel's program *Shuffle* in the July issue of *Rainbow* is a dazzling modernization of Sam Loyd's famous 14-15 puzzle. Like the original 14-15 puzzle, if there is one small error with some of the starting arrangements, the puzzle is unsolvable.

Sam Loyd intentionally presented the puzzle with all the tiles in order except for 14 and 15, which were reversed. He offered a $1000 reward to anyone who could find a sequence which would put the 14 and 15 in their correct order. It is said that ships were wrecked and trains derailed while their respective pilots tried to solve this confounding puzzle.

To spare the reader from a similar fate, I present below a randomization routine which always generates a solvable arrangement. These program lines should replace the corresponding lines in Beistel's listing.

```
13 Z=1
14 FOR X=1 TO 4: FOR Y=1 TO 4:
       A(X,Y)=Z:
       Z=Z+1:
   NEXT Y,X:
   SCREEN 0,0
15 FOR X=1 TO 4: FOR Y=1 TO 4:
       IF X=4 AND Y=>3 GOTO 18
16 RX=RND(4): RY=RND(4):
   IF (RX=X AND RY=Y) OR (RX=4
   AND RY=4) GOTO 16
17 Z=A(X,Y): A(X,Y)=A(RX,RY): A(RX,
   RY)=Z
18 NEXT Y,X: X16=4: Y16=4
```

The reason for this situation is that Beistel's randomizati

arranges the titles in a completely random order. This type of algorithm produces both odd and even permutations of the titles. Any set of moves on the 14-15 puzzle, on the other hand, only produces even permutations. Combinations of even permutations only produce more even permutations.

This was the problem with the original 14-15 puzzle. By swapping just one pair of titles, Loyd induced an odd permutation on the puzzle which made it unsolvable. An excellent discussion of permutations as related to puzzles of this type (including that mindbogglinghexahedron) may be found in *Puzzle It Out: Cubes, Groups, and Puzzles* by John Ewing and Czes Kosniowski (Cambridge University Press.)

James W. Stelly
Houston, TX

Editor:

With the advent of word processors and other software programs with upper and lower case screen displays with up to 80 columns, there may be a problem of poor resolution.

I have a tip that may help by eliminating most interference on your TV type monitor. Carefully wrap the computer output cable with heavy duty aluminum foil and then ground this shield to various places on the TV set by trial until you have eliminated the screen interference. (Try attaching to one lead of the internal antennae.) Grounding to an earth ground may also be helpful.

We really enjoy your magazine!

Ben H. Little
Portage, WI

Editor:

I have been experimenting with the speed up *POKE* commands and have found that you can *CSAVE* and *CLOAD* while in the sped up mode. You first load in the program you want to save. Second, *POKE 65495,0* and then *CSAVE* your program. When you want to *CLOAD* it, first *POKE*

65497,0 and then *CLOAD* it. The volume on your eassette player should be set louder than usual. Note: this method has not always worked with all my programs, but did work with a lot of them. After your cassette player stops, *Poke 65496,0* to put the computer back to normal.

Keep up the good work, *Rainbow*.

Anthony Graves
Louisville, KY

Editor:

I read a great book, "*Assembly Language Graphics for the TRS-80 Color Computer*" by Don Inman and Kurt Inman. It is an excellent tutorial and great referenee book. Hats off to B. Dalton Bookseller stores for really getting into personal eomputing hooks.

Rusty Bernard
Shreveport, LA

Editor's Note: Many B. Dalton Bookseller stores now also carry *the Rainbow*.

Editor:

I would like to bring to your attention two TV shows dedicated to the computer. One can be seen on PBS, the other on cable systems offering SPN. Although neither of the two programs concentrate on any particular brand name or system, new software and hardware are featured on each show.

The "Computer Programme" on PBS was originally produced in England and offers a wealth of information including some wonderfully elear explanations of BASIC programming as well as featuring somc of the latest applications for the computer. It is a serialized show that "builds" from week to week. It appears to be designed to increase the computer literacy of the general public.

SPN's "Personal Computer Show" is produced in Texas (home base for Radio Shack, T.I., and others) and is a "magazine" type show. Besides offering stories and interviews on software, hardware, and applications, they are giving away computers. Because the show is still in production, they do answer questions on the program.

Both shows are a real service to computerists in every level of endeavour. Check your TV listings. Yes, Virginia, there are other uses for a CRT.

James E. Barnett
Latham, NY

SOUNDSHEET SOUNDOFF

Editor:

I want to compliment you on your fine magazine. You have the best Color Computer magazine around. The record inside the 2nd Anniversary issue was a great idea that I would like to see again. Keep up the good work!

Barry Martin
Houma, LA

Editor's Note: It appears that we will not have the soundsheet in *the Rainbow* on a regular basis. For more details, please read *PRINT#-2*.

Editor:

I have been a *Rainbow* subscriber for eight months nowand your 2nd Anniversary issue was the best yet. I get the other CoCo magazines, but *the Rainbow* beats them all, in quality as well as quantity.

I especially enjoyed the sound recording. All three programs loaded without an error. I think the inclusion of a record on a regular basis would be well worth a price increase.

I am looking forward to your next issue.

J. Paul Fountain
Denton, MD

Editor:

I loved the soundsheet in your 2nd Anniversary issue. I hope you can continue to include them, especially for the longer programs. I often hesitate in typing in long programs unless I am certain of their value to me.

I am uncertain about your price increase for the soundsheet. You said something about "increase prices a little" and "very expensive" in the same sentence in your *PRINT #-2* column. The soundsheet is of obvious value but I hope it is well under the price of *Rainbow on Tape* and other tape magazines.

Keep up the good work. Your magazine is the best around but you have competition. Our little CoCo has started to get the attention it deserves.

Dwight A. Spitzer
Mio, MI

PLAY IT AGAIN, SAM CHIP

Editor:

Congratulations on your innovative July 1983 issue. I transferred the programs from record to tape without any difficulties at all. I hope you will consider repeating the inclusion of a soundsheet on a regular basis.

Keep up the great work.

Paul Kapaldo
N. Olmsted, OH

Editor:

I began my subscription to *the Rainbow* in March of this year so I have no idea what the magazine was like in the "old days," but I want to compliment you on the appearance and content of the Anniversary issue. The record is a great idea and if you could include records regularly without increasing the cover cost more than $1, you'd have a good thing going. Perhaps you could include a longer record every other month.

To play the record, I bypassed by stereo and plugged the turntable output directly into the computer record. The first time I was able to load the last two programs on the record, but it took several re-recordings and *CLOADs* before I got the first program to load.

Dick Wilson
Auburn, MA

Editor:

Please hear an impassioned plea from a subseriber with little money and less time. I'm working my way back through school with my eye on a Bachelor's in Information Systems, while working part time (have to keep up the mortgage payment somehow), and the only thing in shorter supply around my house than cash (can't afford tapes) are minutes. Let it be that we can see more of these datacordings in future issues of your magazine. Even if you were to include only half the programs that your tape edition does, I could rest more easily at night, knowing my monthly issues of *Rainbow* weren't going to waste while I frittered away my time with such mundane concerns as college, work, and family.

A little more money per issue I can handle, although the prices of taped editions will probably remain out of my reach until sometime after graduation.

Richard Williams
Tuscon, AZ

Editor:

Just received the July issue and it is superb. I was blown away by the record. It transferred to tape and was readable the first time! Great idea! Why hasn't someone done this before?

Yes, I would pay more for the magazine if it always had a record enclosed. Naturally how much more depeuds on the quality of your programs. The Speeial Adventure issue would have been a natural.

Keep up the good work.

Phil Humbaugh
Hartsdale, NY

Editor:

As I flipped through the 306 pages of the July issue of *the Rainbow* I was very pleased to find a soundsheet (record) contained within its informative pages. And then after reading your column I wrote you to tell you that I think it would be an absolutely fabulous idea to include one of these soundsheets in every issue of *the Rainbow*. I also feel that this kind of an "additive" to *the Rainbow* would easily substantiate a subscription price increase.

Mark Abela
Agincourt, Ontario

Editor:

I wanted you to know that I thought the soundsheet in July's *Rainbow* was a great idea. It would be worth a little extra each month to see you continue and expand it.

Thomas Ebling
East Wenatchee, WA

Editor:

Okay you guys, so you've proved your point. Your magazine is the best computer magazine going. Only the best would actually publish a record. Keep it up! But tell us, what's next? Perhaps a computer radio station that "spins" programs? I leave it up to you. Thanks to this ingenious idea of yours, I will renew my subscription for a few more deeades when it runs out.

Chris Harland
Regina, Saskatchewan

MOTIVATION—NOT REWARDS

Editor:

In the July issue of *the Rainbow*, Carol Kueppers wrote a review of our *Fraction Math Quiz* program. In the title of that review, the word "defects" was used incor-

reetly and, therefore, unfairly. Defect implies a defective product, one that does not function as it was designed to function, and that certainly is not the case with *Fraction Math Quiz*. However, rather than take issue with that unfortunate choice of words, I would like to present to your readers the philosophy behind the design of *Fraction Math Quiz*.

Whether or not an educational program's attributes are defects, as Ms. Kueppers contends in her review of *Fraction Math Quiz* (*the Rainbow*, July 1983, p. 167), or assets depends on one's point of view and, more importantly, on one's educational philosophy. Ms.
program does not provide a strong reward, such as a game to play after correctly working a number of exercises, children will have no motivation for using the program for math drill. I, along with many other educators and most parents, believe that a child's primary motivation for any task should come from the satisfaction of a job correctly done and not from some variation of a carrot-tied-to-a-stick (as in "if you finish your homework, you can have an extra dessert"). This philosophy does not exclude a "pat on the back" for a job well done, nor does it exclude some fun along the way. It simply says that the fun should not be the primary motivation for the child's good performance. Indeed, Ms. Kueppers sums this philosophy up very well when she says that "you should provide the motivation and reward that the program lacks."

Fraction Math Quiz was designed, written, and child-tested with exactly this philosophy in mind. The personalized prompts and musical rewards provide the pat on the back, while the "Starship Commander" menu choice provides a short, enjoyable diversion, as well as an object lesson for children who wonder why they are learning math when a computer can do it so much faster. From this point of view, the program's "defects" become assets that will help a child develop a strong sense of self-motivation.

J.D. German, President
Creative Technical Consultants
Cedar Crest, NM

INFORMATION PLEASE

Editor:

I would like to install a TRS-80 MOD I keyboard in my CoCo. Do you know of any instructions for this, which includes cutting traces, etc.? I have a "retired" MOD I with a perfect keyboard, and I could figure out the modification in time, but if someone has already done it, it could be very helpful and a time-saver.

Also, I did get a chance to attend one day of the *Rainbowfest*—it was super! It looks as though a follow-up next year could be twice as big—and I'll attend again if it is held at Woodfield. Thanks very much for sponsoring the "CoCo Expo."

Clay Howe
Sturgis, MI

Editor:

I'm thinking of buying a Tally Mannesmann printer Model MT160L Revision C and would like to match it to my CoCo.

1) Can I connect it directly to my CoCo as is with no additional interface? 2) Should I expect a problem finding a cable or making one? 3) Will I be able to use the full potential of both printer and computer? 4) Can I use and print information with my CompuServ ROM pack?

Any information would be greatly appreciated.

William Burns
7215 Juniper Drive
Everett, WA 98203

Editor:

I definitely like the new look. I also am increasingly impressed with the quality of your programs and articles.

This letter is being written on my Rainbow word processor using the spooler program in the June issue. What a super spooler! I have been watching *the Rainbow* grow since February 1982 and am pleased to say that every issue has been like opening a present. Speaking of presents! My family has decided to give me a subscription to *the Rainbow* for my Father's Day gift. That's terriffic except that now I need another excuse to visit my friendly neighborhood computer store.

Does anyone know how to insert lines in this word processor? Sometimes it would be good to open up the text.

F.W. McKenzie
Saugerties, NY

Editor:

I am a new subscriber to *the Rainbow* but rest assured that as long as I have my CoCo I shall remain on your list.

I am looking for a disk based RTTY program for the Color Computer but am not having much luck.

James A. Chandler
Abilene, TX

Editor:

I need help solving *Bedlam!* One of my problems is getting the green key. If anyone can help me get out of *Bedlam*, please contact me at 215 North 39th Street. Also, how do you get the people in *Bedlam* to help you?

Brian Gantert
Allentown, PA 18104

CONTRIBUTIONS PLEASE

Editor:

I represe
ization which uses TRS-80 Model I equipment in virtually every aspect of its affairs. We would be grateful if your readers would consider contributing additional Model I equipment: keyboards, expansion interfaces, drivers, and printers would be welcome. All contributions would be fully tax-deductible, since we hold "public charity" status with the IRS.

If you are in a position to make such a contribution or would like more information, please drop me a note or call me (collect, if you like) at (617) 495-9020.

Robert Epstein, Ph.D.
Cambridge Ctr. for Behavioral Studies
11 Ware Street
Cambridge, MA 02138

Editor:

On the way to work recently I spotted this license tag on the car in front of me. A "source" in the County Clerk's Office tells me the tag is a regular issue and not a special personalized plate.

Jack Garner
Memphis, TN

Kueppers's belie

LITTLE PLEASURES

Editor:

This is why I bought a CoCo: I was having a keyboard problem. Sound familiar? I took the computer in Thursday, and on Friday I had a new keyboard in good working order. Maybe I am still at the stage where small things amaze me. But I do doubt, very seriously, that any of the competition in Lancaster could fix this or similar comparatively minor problem(s) at the store. I probably would need all sorts of mailing materials in hand.

Now, more than ever, I will tout the advantages of owning a CoCo! Your magazine is great! The mix of simplicity with sophistication is just fine. I only wanted to find out what all the screaming about the computer revolution was all about. Now I'm hooked!

John Spencer
Lancaster, PA

JAPANESE PHONEMES

Editor:

I am writing in answer to R.W. Odlin's letter in your April issue. I am quite sure the fellow in Houston purposely added those strange hieroglyphics to the CGP-115 to increase international sales. It is the Japanese phonetic system—Katakana. Computers in Japan have the katakana on their keyboard for print, input statements (programming is still done in BASIC) and a printer without this capability would be useless.

Toni Strong
Okinawa, Japan

A FINE EXAMPLE

Editor:

I would like to thank Brian James for a fair and accurate review of our *CoCo-Accountant* program. He mentioned a number of functions he would like to see included in the program, and I am pleased to say that the revised version, which was not available at review time, has a number of improvements. It will allow the entry of both checks and credit card expenditures, list accounts to the printer, allow the setup of income accounts and provide a printed spreadsheet showing expenditures by month and account for the entire year.

Mr. James' review was a fine example of constructive criticism, and we had reached many of his conclusions ourselves.

Michael J. Himowitz, President
Federal Hill Software
Baltimore, MD

MAGIC ANSWER

Editor:

In a recent edition of *the Rainbow*, a reader asked for an explanation of the *Magic Numbers* on the front of the NANOS reference card for the Color Computer. Here is an answer, direct from NANOS.

"The Magic Numbers work as follows: When you subtract a graphics code from the magic number, the result is the number of the graphic which is the inverse or opposite of the original code.

"So, you can peek a number from the screen, subtract it from the original 'Magic Number,' and poke the result out, thus, reversing your screen display."

Hope that answers the question.

Harold L. Laroff
Monsey, NY

INPUT IRRITATION

Editor:

I can't believe it, you've done it again. Please, please, please do not label programs 16K if they are for Extended Color BASIC. I just spent about three hours coding in your *Finding Enrakian Treasure* game when I suddenly ran into ECB commands. This has happened to me many times! I have written before on this subject and am beginning to get irritated. Those of us with BASIC get very few programs from *Rainbow*, and it is very disappointing when the program that appears to be for us is not

Kathy Maurer
Hamilton, OH

Editor's Note: Sorry, Kathy. We regret the error and will be more careful in future issues.

16K, TOO

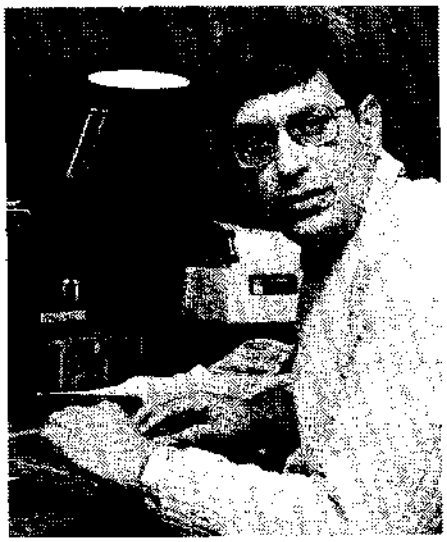
Editor:

Thank you for your prompt, accurate, and gratifyingly glowing review of Spectrosystems' *Fastape* utility (July) for high-speed cassette operations. I would, however, like to correct a small inaccuracy that appeared in the Received and Certified section of the same issue, when *Fastape* was described as a 32K program. As correctly stated in the review, *Fastape* adjusts itself for the memory capacity of the machine it is running on, and thus is compatible with 16K as well as 32K CoCos.

Anyone desiring further information about the program is welcome to leave mail to me on Compuserve, addressed to 72355,-487.

Arthur J. Flexser
Spectrosystems
Miami, FL

PRINT #-2,



We're just about to pop for telling you the news that we've planned **four** of our **RAINBOWfest** shows for the coming year. And the first one? It'll be in Ft. Worth — home of Radio Shack.

RAINBOWfest in Ft. Worth will he October 14-16 at the Hyatt-Regency Ft. Worth. It (and the others to come) will follow the same pattern that made the very first **RAINBOWfest** so suceessful this spring in Chieago. There will be a breakfast with a nationally-known speaker, a series of seminars and the biggest and best show of Color Computer software, hardware and other materials ever assembled in one place.

We are very pleased to announce that Ed Juge, director of computer marketing for Radio Shack, will be our breakfast speaker in Ft. Worth. Ed is extremely well-known and knowledgable and we are proud to have him with us.

Other **RAINBOWfest** shows will be in Long Beach, California, February 17-19; just east of Philadelphia, Pa., April 6-8 and in Chicago June 22-24. All will be at Hyatt House Hotels and all will have the same general format of a big exposition of Color Computer products, a well-known breakfast speaker and a series of seminar sessions.

The way we selected this year's sites is simple: Ft. Worth goes without saying, as headquarters for Radio Shack; we thought Southern California would be a nice place to be in February; Chicago was so successful last year, and so close to Canada and the Great Midwest, that we just had to do it again; while the Philadelphia site (just 6.3 miles from Independence Hall and the Liberty Bell) brings CoCo Community to the populous East Coast for the first time!

Subject to some modification, exhibit times will be 7-10 p.m. on Friday, 10 a.m.-6 p.m. on Saturday and 11 a.m.-4 p.m. on Sunday. Tickets can be had for $8 for a three-day ticket in advance ($12 at the door) or $6 for a one-day ticket ($8 at the door). Breakfast tickets are $11. There is a $1.00 handling charge for each advance *order*.

There are details on another page, and an order form (look for **RAINBOWfest** in the advertiser's index) with deadline information and the like. But, I hope you will come to one or all of these shows. Last spring, **RAINBOWfest** was one of the most exciting things in the CoCo Community. Please come to one or more of them this year!

As you can no doubt tell from our cover, this is our Annual Education Issue. But, as the wizardry of cover artist Fred Crawford shows, too, there is something else afoot this month.

In a way, we designed the cover of this month's issue of *the Rainbow* on purpose to combine a report on the new Color Computers from Radio Shack and what has become our longest-running special issue theme.

We all should be gratified by the new CoCos coming our way from Radio Shack. Not only are they excellent machines, but they should lay to rest — for now and forever — the words of the doomsayers who have, off and on, predicted the demise of the Color Computer.

What we see here is not so much two new Color Computers but a *line* of Color Computers. In making a commitment of this sort to CoCo, it seems to me that Tandy is saying, for once and for all, that the Color Computer is here to stay.

For, as you know, there is more to this than just the hardware of the 64K Color Computer and the Color Computer 2. You can bet that other products are in the works to support these machines —one of which is certainly the OS-9 operating system which will be available with the 64K model.

And not-so-incidentally, these are sleek, well-designed computers. Again, Radio Shack has done a most excellent job. Check them out and I am sure you will agree the new Color Computers are truly a "cover" story!

You will see all sorts of interesting articles on the subject of education in this month's issue, but there are some things missing.

(continued on page 172)

BUILDING SEPTEMBER'S RAINBOW

Our annual education issue . . . Plus, two new CoCo's introduced . . . And, Radio Shack's OS-9 arrives . . .

When my mother gets in a story-telling mood, one of her favorites is about how I spent most of my first school year disrupting class; you see, she was the teacher. It wasn't a one-room schoolhouse—it had two, . . . but the outhouse was way down at the bottom of the hill. Yes, most kids were barefoot and we drank well water from a community dipper. Mom taught three grades. The first, second and third grades each occupied a single row of seats. While two rows were involved in seatwork, Mom gave lessons to the third.

The thought of a computer in that old oiled-floor, white frame tinder box perched on stone columns fairly boogles the mind. To begin with, the nearest electric outlet was about six miles away. Well, I like to think I learned something that first year in school, but the only interactive instruction I can recall came from the tip of a hickory switch. I went the entire year thinking "L-M-N-O-P" was a single word in the middle of the alphabet.

About the only aspect of education that hasn't changed since those days is that squeaking chalk still brings shivers.

In education, *these* are the good, *new* days, so pay attention now; there may be a pop quiz. After all, this is *Rainbow's* education issue. And, the text is a thick one, 324 pages. That means you'll have to read 11 pages a day just to keep from falling behind!

Students, teachers, parents—we have articles and programs for all of you. Need help with a composition? Stephen Tchudi's *Write Idea* will help you develop ideas for a theme, and Larry Konecky's *Bibliography File* will help you complete your research. Teachers will want to check out Paul Luetke's grade averaging utility as well as Norman Garrett's random lab scheduler program. Parents should read the commentaries by Steve Blyn, Dale Peterson (Uncle Bert), Dr. Michael Plog and Larry Kheriaty (author of Radio Shack's *Pilot, Logo* and *Game Writer*, among other works). Educators, especially, should note Dr. Plog's survey form—which will help him help you with software evaluations.

Music, math, spelling, metric conversion, science drills and even educational games are complemented by a dozen-plus reviews of commercial educational programs. Also, every author, it seems, is on the soapbox with an opinion on "the real purpose of computers in education."

Study hard now, and if you're good, we'll tell you, and show you, all about Radio Shack's very own, just-released 64K machine. We have one and it's a honey, "fawn gray," (bone white) with a trim new style and a new keyboard. Along with the 64K CoCo comes the CoCo 2, a new compact version of the Color Computer. Check out *the Pipeline* for Editor Lonnie Falk's impressions, then, for another insider's look at all sides of the long-awaited 64K CoCo, read Dan Downard's observations, not only of the hardware itself, but also his preview of what *OS-9* can do. (Next month, Dale Puckett, nationally-know expert on *OS-9*, will begin a monthly column on *OS-9!)*

Finally, if you've been following Roger Schrag's series on patching *EDTASM+*, this month is icing on the cake--with ice cream on the side. His latest update supports 64K!

Next month graphics, then, in November, the winners of our simulation contest plus a special focus on data communications. But, since this is the education issue, here's a tip in *Economics I:* subscribe to *the Rainbow* while it's still only $22 a year and you'll get more than 15 pounds of *Rainbow* for less than half the cover price. Let's see, in kilos, that's uh . . . page 226, isn't it!

—Jim Reed

Random Lab Scheduler Utility For Frustrated Teachers

By Norman Garrett

Most teachers have had, at one time or another, the problem of scheduling lab sessions where the students far outnumbered the lab stations. Recently, I faced the problem of teaching a programming class of 34 students with only nine microcomputers available. This presented me with quite a scheduling problem. As I pondered how to divide up the limited time that was available in a fair manner, I decided to let my Color Computer do the work for me with *Random Lab Scheduler*.

Letting the computer do the work solved a couple of problems. First, since the schedule was created randomly, I couldn't be accused of juggling the schedule to suit the desires of a few students. The assignment system was totally impartial. Second, the printed output allowed me to have a printed schedule without doing any typing.

During the design stage, I set down some parameters for the program that would make it flexible and yet insure that some of the peculiar needs of an educational setting would be satisfied. First of all, I designed the program so that the number of alternate selectees could be chosen. This solves the problem of absentees among the primary selectees and insures that lab time will not be wasted. Second, in most educational settings, the teacher must insure that all students are guaranteed a minimum amount of time for lab. If a truly random system were used, some students would have second or third chances before others had any at all. To solve that problem, the program uses a "decreasing universe" idea, such as the logic that is used in programming card games. When a selection is made, that item is no longer available for selection, decreasing the universe of available selectees by one.

(Norman Garrett is an Academic Computing Specialist at Arizona State University and a senior consultant for Software Design Concepts. He holds a master's degree in education.)

Student information consists of last name and first name. Array N$ is set up for three entries in the second dimension so that a student number or other piece of information could be inserted. The print logic would have to be modified to print such data. The student information is contained in *DATA* statements. I suggest that this be modified to accept tape or disk files, thus making the program more flexible and usable for multiple classes. Array N$ is loaded in lines 600—640. After loading the array, the program asks for user input concerning the lab sessions (lines 680—810). The user will need to supply the following information:

1) How many lab sessions will be planned?
2) How many work stations are available?
3) How many alternates should be selected?
4) What is the date of the session?
5) What time does the session begin?
6) What time does the session end?

Items 2—6 repeat for each session, allowing the user to alter the data between sessions. For example, sometimes I may want to reserve a station for demonstration during only one of four labs. This allows me to do it. Items 4—6 are for the header on the report.

Lines 880—980 test the status of each student in the array. The idea here is to determine whether each student has had a selection opportunity before opening it up to allow subsequent selections. If the number of students who have not yet been selected is less than the number of stations available for this session, a branch is made to a routine which will assign the remainder (lines 1780—1880). Then, to schedule the rest, go to a routine which will randomly select from all possibilities without regard to a previous selection (lines 1590—1650). That routine fills the rest of the quota for the session.

If there are still students to be assigned for the first time, and that number is equal to or exceeds the number of stations available for the current session, a branch is made to the normal selection routine (lines 1690—1740) for the selection process.

Note that the selection logic is such that two flags are available for testing. The Selection Flag indicates that the student has already been selected for this session. This flag allows you to avoid the pitfall of scheduling the same person twice in the same session. The Primary Flag indicates that the student has already been selected one time. This is checked until the condition is met where everyone has been selected once.

After the primary selection process is complete for the session, the alternate selection process takes place (lines 1920—2010). Once all selectees for a session are tabled, the print routine takes place.

The print logic does not include a paging routine. After each session is printed, you must manually set top-of-form before proceeding to the next session. Paging logic to fit your particular printer would be a relatively simple enhancement to make.

Actual uses for this program are many and varied, especially in educational settings. It can be used to schedule oral presentations, speeches, foreign language lab sessions, industrial arts projects, physical education activities or musical presentations. With modification, output could go to the screen when no printer is available. A screen would have to be formatted and paging logic inserted, neither of which should present major problems.

There is one limitation that you should be aware of: if you inadvertently enter more stations than there are students, an infinite loop will result. The bug could be resolved by an edit routine where the number of stations is input by the user. In reality, though, a situation like that would present a scenario for which this program was not designed.

Hardware Required

16K Color Computer with Extended Color BASIC Printer (DMP-100)

LAB SESSION ROSTER

DATE: 2/28/83 **TIME: 7 PM TO 8 PM**
Session Number 1

PRIMARY SELECTIONS	ALTERNATES
1. Volckmann, Steven	1. Reay, Christie
2. Williams, Olive	2. Martin, Rosemary
3. Rigoni, Ruby	3. Armendariz, Myra
4. Fahey, John	4. Sims, Lucy
5. Soules, Carole	
6. Wood, Patricia	
7. Zelinski, Pat	
8. Kelly, Colleen	
9. Turnmire, Ruth	
10. Koenig, Suzanne	
11. Rigoni, Herbert	
12. Lockhart, Edgar	
13. Moore, Kay	

360....04B9
640.....0753
970....09F9
1280...0C66
1550...0F77
1840...11EC
END....1433

```
10 '*****************************
20 '*   RANDOM LAB SCHEDULER    *
30 '*-------------------------*
40 '* BY NORM GARRETT  9/82   *
50 '*    COPYRIGHT (C) 1983   *
55 '* SOFTWARE DESIGN CONCEPTS*
60 '*****************************
70 '
80 ' ========================
90 ' ===  VARIABLE  LISTING  ===
100 '========================
110 '
120 ' N$(40,3)        NAME TABLE (0
-40)
130 '     0           NOT USED
140 '     0             NOT USED
150 '     2             FIRST NAME
160 ' R                 RECORD COUNT
170 ' S                 # STATIONS
180 ' SS                # SESSIONS
190 ' A                 # ALTERNATES
200 ' SC                SESSION COUNT
ER
```

```
210 ' N(40,2)          RECORD TABLE
220 '    0               SELECT FLAG
230 '    1               PRIMARY FLA
G
240 ' AT$(20,2)        ALTERNATE SEL
ECTIONS
250 '    0               LAST NAME
260 '    1               FIRST NAME
270 ' PT$(20,2)        PRIMARY SELEC
TIONS
280 '    0               LAST NAME
290 '    1               FIRST NAME
300 ' X,Y,Z,C1-C5      NUMERIC WORK
FIELDS
310 ' D$                DATE
320 ' S1$               START TIME
330 ' S2$               STOP TIME
340 ' PF$               PRIMARY FIRST
 NAME
350 ' PL$               PRIMARY LAST
NAME
360 ' AF$               ALTERNATE FIR
ST NAME
370 ' AL$               ALTERNATE LAS
T NAME
380 ' AN$               ALTERNATE NUM
BER
390 ' AP$               LITERAL ". "
400 ' AA$               LITERAL ", "
410 ' AC                ALTERNATE COU
NTER
420 ' PC                PRIMARY COUNT
ER
430 '
440 '
450 '
460 ' ---- INITIALIZATION ----
470 '
480 DIM N$(40,2):DIM N(40,2):R=0
:S=0
490 DIM AT$(20,2)
500 DIM PT$(20,2)
510 POKE 149,0:POKE 150,41
520 SS=0:A=0:SC=0:X=0:Y=0:Z=0:AC
=0:PC=0
530 FOR X=0 TO 30
540 FOR Y=0 TO 2
550 N(X,Y)=0
560 NEXT Y:NEXT X
562 CLS:PRINT "PRESS ANY KEY TO
RANDOMIZE"
564 A$=INKEY$:X=RND(0):IF A$=""
THEN 564
570 '
580 ' ---- READ DATA INTO N$ ARR
AY ----
590 '
600 FOR X=1 TO 50
```

```
610 READ N$(X,1)
620 IF N$(X,1)="XXXXX" THEN 650
630 READ N$(X,2):R=R+1
640 NEXT X
650 '
660 ' ---- INPUT RUN DATA ----
670 '
680 CLS:PRINT@8,"* LAB PLANNER *
"
690 PRINT@40,"----------------"
700 PRINT@96,"HOW MANY SESSIONS"
;:INPUT SS
710 PRINT@160,"HOW MANY WORK STA
TIONS";:INPUT S
720 PRINT@224,"HOW MANY ALTERNAT
ES";:INPUT A
730 SC=SC+1:GOSUB 1320
740 IF SC > SS THEN 1090
750 CLS:INPUT "SESSION DATE";D$
760 PRINT:INPUT "SESSION START T
IME";S1$
770 PRINT:INPUT "SESSION STOP TI
ME";S2$
780 PRINT:PRINT "SET TO TOP OF F
ORM"
790 PRINT"   THEN PRESS <ENTER>"
800 INPUT A$
810 GOSUB 1110
820 '
830 '
840 ' ---- CHECK STATUS & SELECT
 ----
850 '
860 ' << HAS EVERYONE HAD A PRIM
ARY CHANCE? >>
870 '
880 Z=0
890 FOR X=1 TO R
900 IF N(X,1)=1 THEN 920
910 Z=Z+1
920 NEXT X
930 IF Z<>0 THEN 950
940 QQ=1:GOTO 1560
950 IF Z<S THEN 1750
960 FOR C1=1 TO S
970 GOSUB 1660
980 NEXT C1
990 '
1000 ' << SELECT ALTERNATES >>
1010 '
1020 GOSUB 1890
1030 GOSUB 2020
1040 PC=0:AC=0:IF (SC+1) > SS TH
EN 1090
1050 CLS:PRINT@96,"SESSION NUMBE
R ";(SC+1)
```

```
1060 GOTO 710
1070 '
1080 ' ---- END PROGRAM ----
1090 '
1100 END
1110 '
1120 ' ---- PRINT HEADER ROUTINE
 ----
1130 '
1140 PRINT#-2," ":PRINT#-2," "
1150 PRINT#-2,TAB(31)"LAB SESSIO
N ROSTER"
1160 PRINT#-2," "
1170 PRINT#-2,TAB(8)"DATE: ";D$
1180 PRINT#-2,TAB(56)"TIME: ";S1
$;" TO ";S2$
1190 PRINT#-2,TAB(8)"SESSION NUM
BER ";SC
1200 PRINT#-2," "
1210 PRINT#-2,TAB(11)"PRIMARY SE
LECTIONS";
1220 PRINT#-2,TAB(54)"ALTERNATES
"
1230 PRINT#-2,TAB(10)"----------
----------";
1240 PRINT#-2,TAB(53)"----------
--"
1250 RETURN
1260 '
1270 ' ---- PRINT DETAIL ROUTINE
 ----
1280 '
1290 PRINT#-2,TAB(8)C1;". ";PL$;
", ";PF$;
1300 PRINT#-2,TAB(51)AN$;AP$;AL$
;AA$;AF$
1310 RETURN
1320 '
1330 ' ---- INITIALIZE SELECT FL
AGS ----
1340 '
1350 FOR X=1 TO R
1360 N(X,0)=0
1370 NEXT X
1380 RETURN
1390 '
1400 '
1410 '********** D A T A ******
****
1420 '
1430 '
1440 DATA ARMANDARIZ,MYRA,ARMSTR
ONG,AUDREY,BAUER,BARBARA
1450 DATA BROWN,DANNETTE,BROWN,T
HOMAS,BURR,MARILYNN
1460 DATA CASSIDY,KAY,CONIAN,SHE
RRY,COOK,PATRICIA
1470 DATA DAVIS,JANE,EVANS,JOAN,
```

```
GOLLIHARE,WILLIAM
1480 DATA HENDRIX,WARD,HEYMAN,KA
THY,JUDGE,CATHY
1490 DATA KOENIG,SUE,MACY,KARLA,
MARTIN,ROSEMARY,MASON,RIC,MATECK
I,GRACE
1500 DATA MCCARTHY,THOMAS,MENDOZ
A,HENRY,MROSKI,MELODY
1510 DATA NASH,LOIS,NELSON,DEBBY
,NELSON,MARK
1520 DATA SANDERS,CHARLES,SATTER
FIELD,CRAIG,SATTERFIELD,JANET
1530 DATA SCIOTTA,KATHY,STAUBER,
MARIE,THORLEY,EDMUND
1540 DATA ZELINSKI,PAT,MARTIN,MA
RIANNE
1550 DATA XXXXX
1560 '
1570 ' ---- EVERYBODY IS ELIGIBL
E FOR PRIMARY SELECTION ----
1580 '
1590 FOR C2=QQ TO S
1600 SL=RND(R)
1610 IF N(SL,0)=1 THEN 1600
1620 PT$(C2,0)=N$(SL,1):PT$(C2,1
)=N$(SL,2)
1630 N(SL,0)=1:N(SL,1)=1
1640 NEXT C2
```



```
1650 GOTO 1020
1660 '
1670 ' ---- NORMAL PRIMARY SELEC
TIONS ----
1680 '
1690 SL=RND(R)
1700 IF N(SL,0)=1 THEN 1690
1710 IF N(SL,1)=1 THEN 1690
1720 PT$(C1,0)=N$(SL,1):PT$(C1,1
)=N$(SL,2)
1730 N(SL,0)=1:N(SL,1)=1
1740 RETURN
1750 '
1760 ' ---- SCHEDULE REMAINDER -
---
1770 '
1780 C3=1
1790 FOR C5=1 TO S
1800 IF N(C3,0)=1 THEN 1850
1810 IF N(C3,1)=1 THEN 1850
1820 PT$(C5,0)=N$(C3,1):PT$(C5,1
)=N$(C3,2):N(C3,0)=1:N(C3,1)=1
1830 C3=C3+1:IF C3 > R THEN 1870
1840 NEXT C5:GOTO 1020
1850 C3=C3+1:IF C3 > R THEN 1880
1860 GOTO 1800
1870 QQ=C5+1:GOTO 1560
1880 QQ=C5:GOTO 1560
1890 '
1900 ' ---- SELECT ALTERNATES --
--
1910 '
1920 FOR C1=1 TO 20
1930 AT$(C1,0)=" ":AT$(C1,1)=" "
1940 NEXT C1
1950 FOR C1=1 TO A
1960 SL=RND(R)
1970 IF N(SL,0)=1 THEN 1960
1980 AT$(C1,0)=N$(SL,1):AT$(C1,1
)=N$(SL,2)
1990 N(SL,0)=1
2000 NEXT C1
2010 RETURN
2020 '
2030 ' ---- SET UP DETAIL ROUTIN
E ----
2040 '
2050 FOR C1=1 TO S
2060 PRINT#-2," "
2070 PL$=PT$(C1,0):PF$=PT$(C1,1)
2080 IF C1 > A THEN 2110
2090 AN$=STR$(C1):AP$=". ":AL$=A
T$(C1,0):AA$=", ":AF$=AT$(C1,1)
2100 GOTO 2120
2110 AN$=" ":AP$=" ":AL$=" ":AA$
=" ":AF$=" "
2120 GOSUB 1260
2130 NEXT C1
2140 RETURN
```

Copyrighting Your Software
Part 1

By Tom Nelson
Rainbow Contributing Editor

You've seen it so many times. Copyright © 1983 by E. Z. Pickler. ALL RIGHTS RESERVED. blah, blah, blah. Every time you open the front cover of a book, plug in your computer or load a piece of software, that same "©" symbol pops out at you. The reason is simple—this symbol is part of the shield given to authors, and so authors will almost always use it [illegible]

As a software author you will need to understand copyright principles since an understanding of the fundamentals of copyright law is essential to [illegible] understanding of one basic means to protect your software. The principles of copyright law also form the framework for nearly every contract dealing with software. [illegible] what is a copyright?

Copyright is the special form of protection provided to authors by the United States government. Copyrights were contemplated in the Constitution, which allows Congress to pass laws to aid in the promotion of literary works. It is purely federal law, governed by a federal statute and rules promulgated under that statute. [illegible]

Copyright law has a long history [illegible] date [illegible] a new copyright law [illegible] old law. This new law [illegible] which plainly was not [illegible] the new technologies. In fact, in 1980 a special amendment [illegible] which covers the copyright of computer software [illegible] computer software is explicitly recognized by [illegible] all is well and good. (Well, we'll see about [illegible])

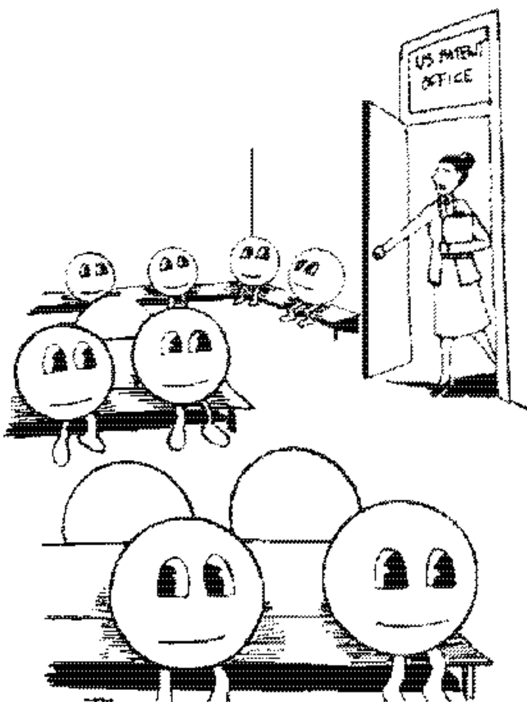
This new amendment will be dealt with [illegible] first the basics. The copyright law is contained in Title 17 of a compendium of all federal statutes called the United States Code (abbreviated "U.S.C."). It covers sections 101 through 509. The statute is administered by the Patent, Trademark and Copyright Office in Washington D.C. The Copyright Office has the power to promulgate rules interpreting the copyright statute and to set policies for registration requirements. A set of regulations governing copyright matters is contained in Title 37 of the Code of Federal Regulations, the compendium of rules issued by federal agencies. A copy of the copyright statute and its regulations should be available in larger libraries.

Copyright law is designed to cover works of authorship. Since computer programs are works of authorship they are covered by copyright law. Computer programs are defined by section 106 of the statute as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." This definition is broad enough to include just about any computer program. Copyright law is devoted to clarifying copyright owner rights, the mechanics of transfer of those rights, methods to obtain those rights and methods to enforce the rights.

Unlike under the old law, a copyright exists in a work from the moment it is created. A work doesn't have to be registered to be protected by copyright law, but for the copyright protection to remain in force, the work cannot be "published" without a proper copyright notice. "Publish" has a technical meaning. It includes selling the work, or even lending it [illegible] you should be careful to include a proper copyright notice at the beginning of your programs. In fact, the first line of code in every program should be your copyright notice.

Now that I have your here, [illegible] you what a proper copyright notice consists of. It should start with the word "Copyright" or the abbreviation "Copr." then the "©" symbol, followed by the year of creation, then the attribution here, "by E. Z. Pickler." The full notice would then be:

(Tom Nelson was formerly a Special Assistant Attorney General for the State of Minnesota. He currently is General Counsel for Softlaw Corporation, makers of the Super "Color" Library.)



"Next!"

"Copyright © 1983 by E. Z. Pickler." This form may he shortened to eliminate the word "Copyright," but the "©" symbol, the date and the attribution must be there. Just to add a little fun, you have to be careful about how the copyright notice is written if you are marketing your product in international markets. The requirements vary with the country, and are often dictated by international copyright conventions subscribed to by a large number of countries. Cross that bridge when you have to. If you need some immediate advice, however, contact a copyright attorney to check on your particular problem.

As I said, your program does not have to be registered with the Copyright Office in order for you to benefit from the protection of the copyright laws. So why register? Because some of the protections afforded by the copyright laws will not apply unless you register. In particular, you cannot obtain attorney's fees and court costs for violations which occur after the first 90 days subsequent to first publication of the program unless you register. This is *very* important, since often attorney's fees can be so great that they actually prohibit you from hringing any action unless you ean eount on being reimbursed when (and if) you win.

Before you rush out and register your copyright for that program, you'd better consider whether you want copyright protection at all. That's right. It is entirely possible that you may decide that copyright protection is not exactly what you want for your program. Although I do not feel it wise for anyone to make that decision before eonsulting an attorney who specializes in intellectual property law, you should at least have an idea of the basic consideration involved in the decision.

Copyright protection is just one means to protect your software creation. Another popular legal shield is trade secret protection. Trade secret protection is afforded those who keep a piece of information secret. It provides a remedy against the unauthorized disclosure of trade secrets. I will discuss this topic later. Here, suffice it to say that the essential element of trade secret protection is total unavailability of your program to the public. Copyright law is almost the antithesis of this concept. When you copyright a piece of software you are usually selling the object code to the public. This is making your program public. In fact, if you register your program with the Copyright Office, this is arguably a publication of your program to the public. How can anything explicitly made public be considered a trade secret?

Many authorities feel that if the source code, your assembly language program, is kept secret, copyright of the object code alone is not sufficient to destroy trade secret protection for your program since object code is not direetly translatable into your precise source code. Like other areas of copyright law in relation to computer software, this thesis is untested, and you should weigh carefully the factors in deciding whether to register your eopyright in your program. If trade seeret protection is more important to you than eopyright protection, perhaps you will not wish to register your copyright in the program, but instead will rely on the trade secret protection provided by the laws of your state. Again, you will want to consult with you attorney before making a final decision.

Actual registration requirements will be left to next month's column. Here let's discuss a bit more what the copyright laws will and will not do for you.

Copyright protection protects the actual expression of ideas, but not the ideas themselves. This is a crucial distinction, especially with games. An idea is a concept or a generalized schema relating to a specific topic. The expression of an idea is just that; your particular manifestation of the idea.

This distinction can be shown best with the so-called "maze-chase" games. There are many maze-chase games of the Pac Man™ variety in which some beasty chases a gobbling good guy which innocently is munching some goody or other. The good guy can eat a power pill or the like to be able to eat the beasty, and so on and so forth. We've all played this one a thousand times. What I have just described is the idea of a game, a concept. This concept is not copyrightable. If it were, the first to think of it could unfairly corner the market on an idea, stopping others from using it in any way in the future.

Copyright law was not designed to provide this protection; patent law was. Patent law, however, does not apply to this kind of application software. (This will be discussed to some extent in a later column.) Instead, copyright law was designed to protect an author's particular expression of the idea. Thus, courts have devoted much effort to determine what is the expression in each particular case, and whether there has been an infringement of that expression. I will tell you the saga of Pac Man™ in a later column to show just how the courts have dealt with the "knock off" problem in the software area.

By now you are beginning to get a feeling that copyright law may not be very clean cut and simple. In the computer software area, an especially new and complex area, this feeling is well founded. Copyright law was not well designed for computer software which is a hybrid kind of intellectual property. Software is too new, too novel even for the new 1976 and 1981 additions to the copyright law.

Copyright law for software grew out of software for huge mainframes. No thought has been devoted to the special problems inherent in microcomputer software which is increasingly taking a larger market share, and there are real, different problems. Mainframe software was and is designed to be user modifiable. Customization is presumed. Customization of microcomputer software is not presumed; quite the opposite is true. It is assumed that the user will find the offered software totally sufficient for his or her needs. If this is not true, another piece of software is sought and bought. Since microcomputer software is relatively cheap, this has become the standard business practice.

Thus, copyright laws designed for mainframes cannot hope to be directly applicable to microcomputers. Yet, the copyright laws have not changed. There is much growing to be done in the next few years in the software industry and in the laws relating to the protection of software. You, as an author, can play a significant part in that growth. You can help to shape the software and the laws of the future to make your efforts ever more rewarding and profitable.

In my next column I will discuss the steps for registering the copyrights for your programs. In the following months I will discuss more about how the copyright laws work to define the author-publisher relationship. For those of you who can't wait, try to find the work *The Copyright Kit: How to Copyright Your Computer Software* by Noel D. Adler and Steven A. Novani, National Attorneys' Publications, Inc., 1981. Until then, good computing.

Spectran Offers New Dimension to *Spectaculator*

Sometimes it's hard to cope with the frustration of incompatible software. Having two useful programs which can't talk to each other is like being in a foreign country. You really need a translator to get things going. *Spectran* solves the problem of data incompatibility when loading *Spectaculator* files into a word processing program. Most word processors, including Radio Shack's *Scripsit,* expect ASCII text files. *Spectaculator* produces its own unique files.

Spectran translates *Spectaculator* files into standard ASCII files or vice versa. Thus, you can easily incorporate spreadsheet data into a report prepared with a word processor, or you can add data from a text file to a new spreadsheet model. This flexibility makes a wide range of applications available to the *Spectaculator* user.

Running *Spectran* is easy. You select menu options (1) *Spectaculator* to ASCII or (2) ASCII to *Spectaculator* file conversions. For option (1) you set a line length for the converted file (32 to 132 columns wide) and then give the name/SPC of the *Spectaculator* file to be converted. The program directs you to change to your word processing disk and to name the new ASCII file. The newly converted file is then written and you are ready to edit or incorporate the data into a report.

Option (2) reverses the above process and allows you to convert an ASCII file to one which can be read by *Spectaculator*. The new *Spectaculator* file has columns set to width ten. Only data values or text are transferred. You cannot transfer equations as column or row relationships.

The whole process is straightforward. The instructions thoroughly cover all the operating points. There are only three basic limitations, which are discussed in the instructions. First, the program only runs from files stored on Drive 0 (obviously not a problem for most of us single drive users). Second, the maximum width of a ASCII data column is 10. Files will still require some reformatting when entered into *Spectaculator*. Third, memory space is limited to 8K for users with a 16K disk system. Larger files will lose data during the file transfer.

My experience with the program emphasized the data formatting limitations. In going from *Spectaculator* to ASCII files, everything worked out well. But reversing the process caused some minor irritations. Blank spaces used to justify columns of data in ASCII tables caused most of the problems. These blanks were converted to a single blank. In addition, some text headings were hopelessly scrambled due to the 10 character per column limit. My suggestion, also in the instructions supplied with the program, is to only transfer data tables, without text, to *Spectaculator*. You can add the text headings after reading the data into *Spectaculator* itself.

Another problem that can cause difficulty is getting file types confused in the conversion process. When I specified a non-*Spectaculator* file to be converted to ASCII, the system crashed! Only a full, power-down reset could restore operation. I also tried converting a non-ASCII file to *Spectaculator*. While the system didn't crash as before, I did get some nonsense results in the subsequent *Spectaculator* program run. These problems did not, however, result in any data being lost or in any subtle errors being propagated.

As a bonus feature, Crimson Software offers two utility programs in the *Spectran* package. Their first bonus is *Disklook,* a disk file utility which reads files and directory information, byte by byte. This program also allows you to modify existing disk files and create new directory entires. A nice discussion of the storage format for the CoCo disk is also provided. The second bonus program is *Dirdupl,* which will save a backup copy of the disk directory on unused sectors of track 17. If you ever need to restore a *KILL*ed file or recover a bad directory, the program will use this backup copy to rewrite the original directory.

I recommend *Spectran* to anyone who needs to transfer ASCII data to *Spectaculator,* or who wants to move *Spectaculator* data files into a word processing system. Used with care, this program solves the file incompatibility problems inherent in Radio Shack's Color Computer software.

(Crimson Software, 32 Beverly Heights, Tuscaloosa, AL 35404, 16K or 32K Color Computer Disk, $25)

—Stuart Hawkinson

Elements

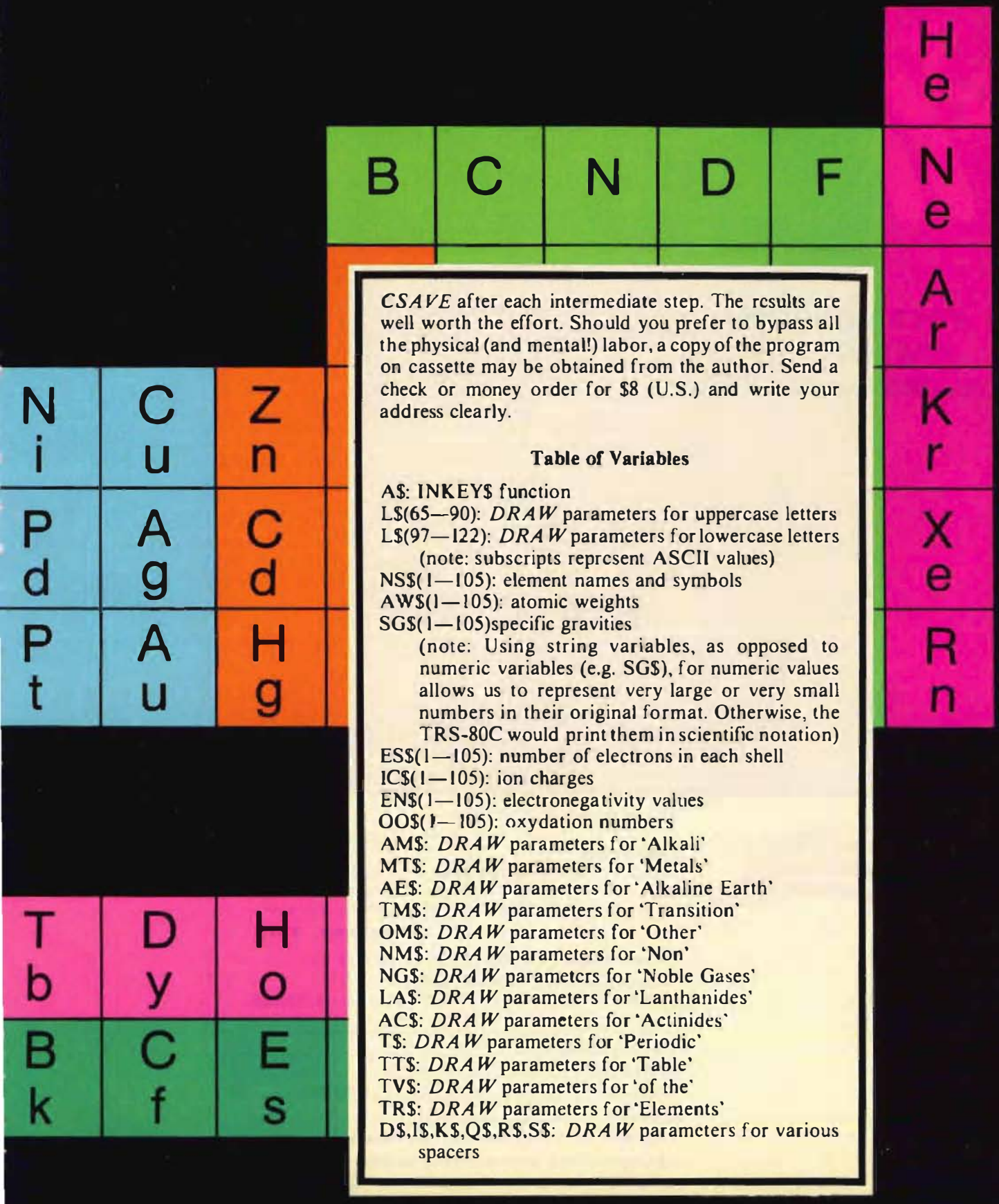
By Gary Benoit

Elements, requiring 32K and Extended Color BASIC, will be of value to all those with an interest in the sciences, more specifically to chemistry teachers and students.

It displays the complete periodic table of the elements and the eight classes of elements on the graphics screen, in both upper- and lowercase letters. Uppercase letters were designed by M.H. Endres (*the Rainbow*, May 1982). I found his letters clear and eye pleasing and decided to use them in my program. Only one minor correction had to be made: his letter C *DRAW* parameters should end with BD4, not BD. Lowercase letters were designed by me.

The program also provides valuable information on each of the 105 known and accepted elements. This information can be displayed on the text screen by simply entering the element's symbol (entering symbols can bedone whether you are in the graphics mode or in the text mode, ie, you can request information at any time after initialization). If you are in the text mode, simply pressing the Spacebar will return you to the last graphics screen displayed. Pressing the Spacebar again will advance you to the next graphics screen (pressing the Spacebar continuously will cycle the display through all nine graphics screens). Entering a nonexistent symbol will result in a timed "Entry Error" display and a return to the last displayed graphics screen. As the program requires a two symbol entry for the elements, you must repeat the symbol a second time for all single symbol elements. The total program design, including the liberal use of the INKEY$ function, has resulted in a very user friendly program. The extensive graphics and large amounts of data have a price: the program is very long (it uses most of the available memory in a 32K CoCo), and will require time and patience, not to mention accuracy, in order to enter the complete listing. For this reason, it is strongly recommended that the program be entered in two or more steps. Needless to say, don't forget to

(Gary E. Benoit, a supervisor for the telephone company in Calgary, Alberta, Canada, has been working with computers for 12 years. His three teenagers and four-year-old are all very interested in computers.)



CSAVE after each intermediate step. The results are well worth the effort. Should you prefer to bypass all the physical (and mental!) labor, a copy of the program on cassette may be obtained from the author. Send a check or money order for $8 (U.S.) and write your address clearly.

Table of Variables

A$: INKEY$ function
L$(65—90): *DRAW* parameters for uppercase letters
L$(97—122): *DRAW* parameters for lowercase letters
(note: subscripts represent ASCII values)
NS$(1—105): element names and symbols
AW$(1—105): atomic weights
SG$(1—105)specific gravities
(note: Using string variables, as opposed to numeric variables (e.g. SG$), for numeric values allows us to represent very large or very small numbers in their original format. Otherwise, the TRS-80C would print them in scientific notation)
ES$(1—105): number of electrons in each shell
IC$(1—105): ion charges
EN$(1—105): electronegativity values
OO$(1—105): oxydation numbers
AM$: *DRAW* parameters for 'Alkali'
MT$: *DRAW* parameters for 'Metals'
AE$: *DRAW* parameters for 'Alkaline Earth'
TM$: *DRAW* parameters for 'Transition'
OM$: *DRAW* parameters for 'Other'
NM$: *DRAW* parameters for 'Non'
NG$: *DRAW* parameters for 'Noble Gases'
LA$: *DRAW* parameters for 'Lanthanides'
AC$: *DRAW* parameters for 'Actinides'
T$: *DRAW* parameters for 'Periodic'
TT$: *DRAW* parameters for 'Table'
TV$: *DRAW* parameters for 'of the'
TR$: *DRAW* parameters for 'Elements'
D$,I$,K$,Q$,R$,S$: *DRAW* parameters for various spacers

RAINBOWfest

FREE RAINBOW POSTER For First 200 Tickets Sold.

The fun and excitement of *RAINBOWfest* is coming your way . . . and now there will be a *RAINBOWfest* near you!

For the 1983-84 season, we've scheduled four *RAINBOWfests* in four parts of the country. Each one will offer fun, excitement, new products, seminars and information for your CoCo! And for those who (perish the thought) don't like CoCo as much as you, we've scheduled each *RAINBOWfest* in an area that will provide fun and enjoyment for the whole family.

Just look at this great lineup:

Fort Worth, Texas — October 14—16. We'll kick off the *RAINBOWfest* "season" with a visit to CoCo's home — Radio Shack's headquarters in Fort Worth. Our CoCo Community Breakfast will feature Ed Juge, Radio Shack's director of computer merchandising, as the guest speaker. And, we hope to be able to offer tours of Tandy Center for those who will be at what promises to be an exceptional *RAINBOWfest* kickoff.

Long Beach, California — February 17-19. What a way to get away from the winter doldrums! And what better place than sunny Southern California with thousands of fellow CoCo owners! Fly with CoCo to the sun for the winter. Top flight seminars will draw on many local CoCo experts.

Cherry Hill, New Jersey — April 6—8. Just 6.3 miles from the Liberty Bell, *RAINBOWfest* comes to the populous northeast. It's a close drive from New York, Boston, Philadelphia, Washington and Baltimore.

Chicago — June 22—24. We'll play *RAINBOWfest* again, CoCo! This is the site of CoCo's *very first show* this spring. And right next to the world's largest indoor shopping mall.

Every show will be held at a Hyatt-Regency Hotel and all will be offering special rates for *RAINBOWfest*. Every show will open at 7—10 p.m. Friday, run 10 a.m.—6 p.m. Saturday and close with an 11 a.m.—4 p.m. session Sunday. Each will have a CoCo Community Breakfast featuring an outstanding national speaker from the Color Computer World. And each exhibition will be interspaced with a number of seminar sessions on all aspects of CoCo —from writing in machine language to making your BASIC work better.

But most of all, there will be exhibitors. Lots of them. All ready to demonstrate products of every kind. Some with special programs and hardware items to introduce. Others with show specials.

Tickets can be secured directly from *the Rainbow*. We'll also send you a special reservation form so you can get your special room rate.

Come to *RAINBOWfest* . . . help us all celebrate CoCo Community at its finest!

RAINBOWfest—Fort Worth
DATES: Oct. 14—16, 1983
HOTEL: Hyatt Regency, Ft Worth
ROOMS: $55 per night single/double
KEYNOTE: Ed Juge
Director
Computer Merchandising
Radio Shack
Advance Ticket Deadline: Oct. 10, 1983

RAINBOWfest—Long Beach
DATES: Feb. 17—19, 1984
HOTEL: Hyatt Regency, Long Beach
ROOMS: $59 per night single/double
KEYNOTE: To Be Announced
Advance Ticket Deadline: Feb. 13, 1984

RAINBOWfest—Cherry Hill, NJ
DATES: April 6—8, 1984
HOTEL: Regency-Hyatt Cherry Hill
ROOMS: To Be Announced
KEYNOTE: To Be Announced
Advance Ticket Deadline: Mar. 30, 1984

RAINBOWfest—Chicago
DATES: June 22—24, 1984
HOTEL: Regency Hyatt Woodfield
ROOMS: To Be Announced
KEYNOTE: To Be Announced
Advance Ticket Deadline: June 18, 1984

RAINBOWfest

YES, I'm coming to *RAINBOWfest* in

☐—Ft. Worth ☐—Long Beach ☐—Cherry Hill ☐—Chicago.

Please send me:

_____ three day tickets at $8 each total _____
_____ one day tickets at $6 each total _____
_____ breakfast tickets at $11 each total _____

Handling Charge $1.00

TOTAL ENCLOSED (U.S. FUNDS ONLY, PLEASE) $ _____

—Also send me a hotel reservation card for:
☐—Ft. Worth ☐—Long Beach ☐—Philadelphia ☐—Chicago

NAME *(please print)* _____
STREET & NUMBER _____
CITY & STATE _____
TELEPHONE _____ ZIP CODE _____
COMPANY _____

Orders sent less than two weeks prior to show opening will be held for you at the door.

VISA, MasterCharge, American Express accepted.

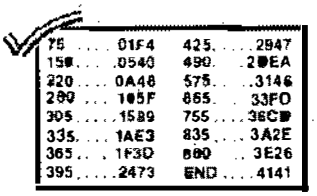
My Account # _____ Ex. Date: _____

Signature _____

Make checks payable to:
the Rainbow

MAIL TO:

RAINBOWfest
P.O. BOX 209
Prospect, KY 40059



75	01F4	425	2947
150	0540	490	2BEA
220	0A48	575	3146
280	105F	665	33FD
305	1589	755	36CB
335	1AE3	835	3A2E
365	1F3D	880	3E26
395	2473	END	4141

The listing:

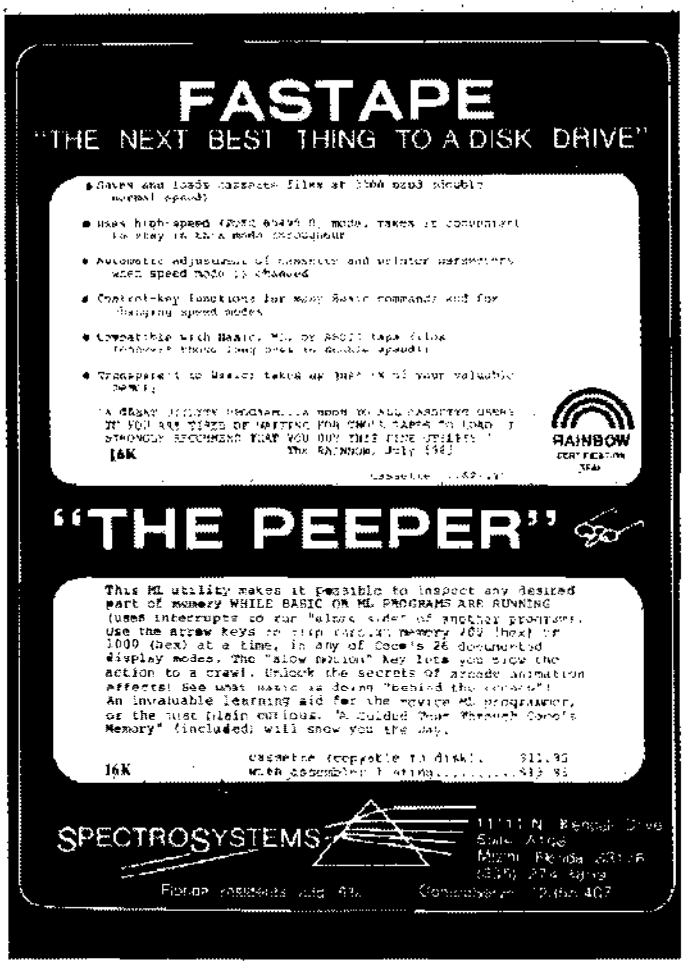
```
10 REM elements
15 REM GARY BENOIT
20 REM 343 TEMPLEBY PL.N.E.
25 REM CALGARY,AB
30 REM CANADA T1Y 5H2
35 REM JAN.83
40 CLS:PRINT@225,"DO YOU WANT IN
STRUCTIONS?(Y/N)"
45 A$=INKEY$:IFA$=""THEN45ELSEIF
A$="Y"THEN850ELSEIFA$<>"N"THEN40
50 CLS0:PRINT@230,"ONE MOMENT, P
LEASE";
55 CLEAR1800:DIML$(122),NS$(105)
,AW$(105),SG$(105),ES$(105),IC$(
105),EN$(105),OO$(105)
60 FORV=65TO90:READL$(V):NEXT:FO
RV=97TO122:READL$(V):NEXT:GOSUB4
45
65 FORX=1TO105:READNS$(X),AW$(X)
,SG$(X):NEXTX:FORX=1TO105:READES
$(X):NEXTX:FORX=1TO105:READIC$(X
),EN$(X),OO$(X):NEXTX
70 REM SET UP SCREEN
75 PMODE4,1:GOSUB80:GOTO100
80 PCLS:CLS0:SCREEN0,0:FORX=14TO
238STEP14:LINE(X,0)-(X,191),PSET
:NEXT
85 FORX=42TO154STEP14:LINE(X,0)-
(X,61),PRESET:NEXT:FORX=168TO224
STEP14:LINE(X,0)-(X,21),PRESET:N
EXT:FORX=84TO238STEP14:LINE(X,12
2)-(X,141),PRESET:NEXT:FORX=14TO
238STEP14:LINE(X,142)-(X,150),PR
ESET:NEXT
90 LINE(14,150)-(14,191),PRESET:
LINE(238,150)-(238,191),PRESET:L
INE(28,0)-(28,21),PRESET
95 FORY=21TO141STEP20:LINE(0,Y)-
(255,Y),PSET:NEXT:LINE(0,150)-(2
55,150),PSET:LINE(0,171)-(255,17
1),PSET:SCREEN1,0:RETURN
100 GOSUB195:GOSUB200:GOSUB205:G
OSUB210:GOSUB215:GOSUB225:GOSUB2
35:GOSUB255:GOSUB265:Z=1
105 A$=INKEY$:IFA$=""THEN105ELSE
IFA$=" "THEN110ELSE520
110 GOSUB80:GOSUB205:DRAW"BM72,1
0;XAM$;XI$;XMT$;":PSET(109,4):Z=
2
115 A$=INKEY$:IFA$=""THEN115ELSE
IFA$=" "THEN120ELSE520
120 GOSUB80:GOSUB210:DRAW"BM46,1
0;XAE$;XI$;XMT$;":PSET(83,4):Z=3
125 A$=INKEY$:IFA$=""THEN125ELSE
IFA$=" "THEN130ELSE520
130 GOSUB80:GOSUB235:DRAW"BM62,1
0;XTM$;XI$;XMT$;":PSET(99,4):PSE
T(113,4):Z=4
135 A$=INKEY$:IFA$=""THEN135ELSE
IFA$=" "THEN140ELSE520
140 GOSUB80:GOSUB225:DRAW"BM78,1
0;XOM$;XI$;XMT$;":Z=5
145 A$=INKEY$:IFA$=""THEN145ELSE
IFA$=" "THEN150ELSE520
150 GOSUB80:GOSUB215:DRAW"BM92,1
0;XNM$;XMT$;":Z=6
155 A$=INKEY$:IFA$=""THEN155ELSE
IFA$=" "THEN160ELSE520
160 GOSUB80:GOSUB200:DRAW"BM80,1
0;XNG$;":Z=7
165 A$=INKEY$:IFA$=""THEN165ELSE
IFA$=" "THEN170ELSE520
170 GOSUB80:GOSUB255:DRAW"BM86,1
0;XLA$;":PSET(137,4):Z=8
175 A$=INKEY$:IFA$=""THEN175ELSE
IFA$=" "THEN180ELSE520
180 GOSUB80:GOSUB265:DRAW"BM94,1
0;XAC$;":PSET(117,4):PSET(131,4)
:Z=9
185 A$=INKEY$:IFA$=""THEN185ELSE
IFA$=" "THEN75ELSE520
190 REM DRAW CLASSES OF ELEMENTS
195 DRAW"BM22,10;XT$;BM82,10;XTT
$;BM123,10;XTV$;BM171,10;XTR$;":
PSET(44,4):PSET(65,4):DRAW"BM5,9
;XL$(72);":RETURN
200 DRAW"BM244,9;XL$(72);XD$;XL$
(101);BD13BL4;XL$(78);XD$;XL$(10
1);BD12BL4;XL$(65);XD$;XL$(114);
BD12BL4;XL$(75);XD$;XL$(114);BD1
2BL4;XL$(88);XD$;XL$(101);BD12BL
4;XL$(82);XD$;XL$(110);":RETURN
205 DRAW"BM5,30;XL$(76);XD$;XL$(
105);BD12BL4;XL$(78);XD$;XL$(97)
;BD12BL4;XL$(75);BD20BL4;XL$(S2)
;XD$;XL$(98);BD12BL4;XL$(67);XD$
;XL$(115);BD12BL4;XL$(70);XD$;XL
$(114);":PSET(7,32):RETURN
210 DRAW"BM19,30;XL$(66);XD$;XL$
(101);BD12BL4;XL$(77);XD$;XL$(10
3);BD12BL4;XL$(67);XD$;XL$(97);B
D12BL4;XL$(83);XD$;XL$(114);BD12
BL4;XL$(66);XD$;XL$(97);BD12BL4;
XL$(82);XD$;XL$(97);":RETURN
215 DRAW"BM173,30;XL$(66);BR10;X
L$(67);BR10;XL$(78);BR10;XL$(79)
;BR10;XL$(70);BL46BD20;XL$(83);X
D$;XL$(105);XK$;XL$(80);BR10;XL$
(83);BR10;XL$(67);XD$;XL$(108);"
:PSET(189,52)
220 DRAW"BM201,70;XL$(65);XD$;XL
$(115);XK$;XL$(83);XD$;XL$(101);
XK$;XL$(66);XD$;XL$(114);BL18BD1
2;XL$(84);XD$;XL$(101);XK$;XL$(7
```

```
3);BD20BL4;XL$(65);XD$;XL$(116);
":RETURN
225 DRAW"BM173,50;XL$(65);XD$;XL
$(108);BD12BL18;XL$(90);XD$;XL$(
110);XK$;XL$(71);XD$;XL$(97);XK$
;XL$(71);XD$;XL$(101);BL32BD12;X
L$(67);XD$;XL$(100);XK$;XL$(73);
XD$;XL$(110);XK$;XL$(83);XD$;XL$
(110);"
230 DRAW"BM201,90;XL$(83);XD$;XL
$(98);BL46BD12;XL$(72);XD$;XL$(1
03);XK$;XL$(84);XD$;XL$(108);XK$
;XL$(80);XD$;XL$(98);XK$;XL$(66)
;XD$;XL$(105);XK$;XL$(80);XD$;XL
$(111);":PSET(203,112):RETURN
235 DRAW"BM33,70;XL$(83);XD$;XL$
(99);XK$;XL$(84);XD$;XL$(105);XK
$;XL$(86);BR10;XL$(67);XD$;XL$(1
14);XK$;XL$(77);XD$;XL$(110);XK$
;XL$(70);XD$;XL$(101);XK$;XL$(67
);XD$;XL$(111);":PSET(49,72):PSE
T(133,72)
240 DRAW"BM131,70;XL$(78);XD$;XL
$(105);XK$;XL$(67);XD$;XL$(117);
BL116BD12;XL$(89);BR10;XL$(90);X
D$;XL$(114);XK$;XL$(78);XD$;XL$(
98);XK$;XL$(77);XD$;XL$(111);XK$
;XL$(84);XD$;XL$(99);"
245 DRAW"BM103,90;XL$(82);XD$;XL
```



```
$(117);XK$;XL$(82);XD$;XL$(104);
XK$;XL$(80);XD$;XL$(100);XK$;XL$
(65);XD$;XL$(103);BL116BD12;XL$(
76);XD$;XL$(97);XK$;XL$(72);XD$;
XL$(102);XK$;XL$(84);XD$;XL$(97)
;"
250 DRAW"BM75,110;XL$(87);BR10;X
L$(82);XD$;XL$(101);XK$;XL$(79);
XD$;XL$(115);XK$;XL$(73);XD$;XL$
(114);XK$;XL$(80);XD$;XL$(116);X
K$;XL$(65);XD$;XL$(117);BL116BD1
2;XL$(65);XD$;XL$(99);XK$;XL$(82
);XD$;XL$(102);XK$;XL$(72);XD$;X
L$(97);":RETURN
255 DRAW"BM33,159;XL$(67);XD$;XL
$(101);XK$;XL$(80);XD$;XL$(114);
XK$;XL$(78);XD$;XL$(100);XK$;XL$
(80);XD$;XL$(109);BR9BU8;XL$(83)
;XD$;XL$(109);BR9BU8;XL$(69);XD$
;XL$(117);XK$;XL$(71);XD$;XL$(10
0);XK$;XL$(84);XD$;XL$(98);"
260 DRAW"BM145,159;XL$(68);XD$;X
L$(121);XK$;XL$(72);XD$;XL$(111)
;XK$;XL$(69);XD$;XL$(114);XK$;XL
$(84);XD$;XL$(109);BR9BU8;XL$(89
);XD$;XL$(98);XK$;XL$(76);XD$;XL
$(117);":RETURN
265 DRAW"BM33,180;XL$(84);XD$;XL
$(104);XK$;XL$(80);XD$;XL$(97);X
K$;XL$(85);BR10;XL$(78);XD$;XL$(
112);XK$;XL$(80);XD$;XL$(117);XK
$;XL$(65);XD$;XL$(109);BR9BU8;XL
$(67);XD$;XL$(109);BR9BU8;XL$(66
);XD$;XL$(107);"
270 DRAW"BM145,180;XL$(67);XD$;X
L$(102);XK$;XL$(69);XD$;XL$(115)
;XK$;XL$(70);XD$;XL$(109);BR9BU8
;XL$(77);XD$;XL$(100);XK$;XL$(78
);XD$;XL$(111);XK$;XL$(76);XD$;X
L$(114);":RETURN
275 REM DRAW LETTERS
280 DATA"U4E2F2D2L4R4D2","RU6LR3
FDGL2R2FDGL3BR4","BR4BUGL2HU4ER2
FDBD4"
285 DATA"RU6LR3FD4GL2BR3","U3R4L
4U3R4BD6L4R4","U3R4L4U3R4BD6","B
R2BU3R2D2GL2HU4ER2FBD5","U6D3R4U
3D6","BRR2LU6LR2BRBD6","BU2DFR2E
U5BD6","U6BR4G3F3","R4L4U6BR4BD6
","U6F2E2D6","U6DF4U5D6","R4L4U6
R4D6"
290 DATA"U6R3FD8L3BR4BD3","BUU4E
R2FD4GL2HBR2BUF2","U6R3FDGL3RF3"
,"BUFR2EUHL2HUER2FBD5","BU6R4L2D
6BR2","U6D6R4U6D6","BU6D4F2E2U4B
D6","U6D6E2F2U6D6","UE2H2UDF2E2U
DG2F2D","BU6DF2E2UDG2D3BR2","BU6
R4DG2LR2LG2DR4"
295 DATA"BU4R3FD2GL2HUR4BD2","U5
D2R3FDGL3BR4","BUU2ER3BG3BLFR3",
```

```
"BRHUER3U2D5L3BR3","BRHU2ER2FDL2
BF2L3R3","BR2U5EBD3L2BR3BD3","R3
EU3HL2GDFR2EBD3","U6BD3R3FD2","B
RRU4LRD4RBR","BUFREU3LR2BD4","BR
U5BR3G3RF2","BRRU5LRD5RBR","BLU4
R3D4U4R3D4"
300 DATA"U4R3FD3","BRR2EU2HL2GD2
FBR3","U5R3FDGL3BD2BR4","BR4U5L3
8DFR3BD2","U4DER3BD4","R3EUL4UER
3BD4","BU4R3L2U2D5FRBR","BU4D3FR
2EU3D4","BU2U2BR4D2G2H2BR4BD2","
U4BD4E2F2U4BD4","E4BL4F4","E4BL4
F2BF2","BU4R4G4R4"
305 REM ELEMENT INFO
310 DATA HYDROGEN h,1.00797,0.00
009,HELIUM he,4.0026,0.00018,LIT
HIUM li,6.939,0.534,BERYLLIUM be
,9.01222,1.85,BORON b,10.811,2.4
5,CARBON c,12.0111,3.52,NITROGEN
 n,14.0067,0.00125,OXYGEN o,15.9
994,0.001429,FLUORINE f,18.9984,
0.0017
315 DATA NEON ne,20.183,0.0009,S
ODIUM na,22.9898,0.97,MAGNESIUM
mg,24.312,1.74,ALUMINUM al,26.98
15,2.70,SILICON si,28.086,2.4,PH
OSPHORUS p,30.9738,1.83,SULFUR s
,32.064,2.0,CHLORINE cl,35.453,0
.0032,ARGON ar,39.948,.00178
```




```
320 DATA POTASSIUM k,39.102,0.86
,CALCIUM ca,40.08,1.55,SCANDIUM
sc,44.956,2.5,TITANIUM ti,47.90,
4.5,VANADIUM v,50.942,5.96,CHROM
IUM cr,51.996,7.1,MANGANESE mn,5
4.938,7.2,IRON fe,55.847,7.87,CO
BALT co,58.9332,8.9,NICKEL ni,58
.71,8.9
325 DATA COPPER cu,63.54,8.92,ZI
NC zn,65.37,7.14,GALLIUM ga,69.7
2,5.9,GERMANIUM ge,72.59,5.36,AR
SENIC as,74.9216,5.7,SELENIUM se
,78.96,4.8,BROMINE br,79.909,3.1
2,KRYPTON kr,83.80,0.0037,RUBIDI
UM rb,85.47,1.53,STRONTIUM sr,87
.62,2.6
330 DATA YTTRIUM y,88.905,3.8,ZI
RCONIUM zr,91.22,6.4,NIOBIUM nb,
92.906,8.4,MOLYBDENUM mo,95.94,1
0.2,TECHNETIUM tc,(97),,RUTHENIU
M ru,101.07,12.2,RHODIUM rh,102.
905,12.5,PALLADIUM pd,106.4,12.1
6,SILVER ag,107.87,10.5,CADMIUM
cd,112.40,8.65
335 DATA INDIUM in,114.82,7.3,TI
N sn,118.69,7.3,ANTIMONY sb,121.
75,6.684,TELLURIUM te,127.60,6.2
4,IODINE i,126.9044,4.93,XENON x
e,131.30,0.0058,CESIUM cs,132.90
5,1.9,BARIUM ba,137.34,3.5,LANTH
ANUM la,138.91,6.15,CERIUM ce,14
0.12,6.9
340 DATA PRASEODYMIUM pr,140.907
,6.5,NEODYMIUM nd,144.24,6.9,PRO
METHIUM pm,(145),,SAMARIUM sm,15
0.35,7.7,EUROPIUM eu,151.96,,GAD
OLINIUM gd,157.25,,TERBIUM tb,15
8.924,,DYSPROSIUM dy,162.50,,HOL
MIUM ho,164.930,,ERBIUM er,167.2
6,4.77
345 DATA THULIUM tm,168.934,,YTT
ERBIUM yb,173.04,5.51,LUTETIUM l
u,174.97,,HAFNIUM hf,178.49,,TAN
TALUM ta,180.948,16.6,TUNGSTEN w
,183.85,19.3,RHENIUM re,186.2,20
.53,OSMIUM os,190.2,22.57,IRIDIU
M ir,192.2,22.4,PLATINUM pt,195.
09,21.37
350 DATA GOLD au,196.967,19.3,ME
RCURY hg,200.59,13.55,THALLIUM t
l,204.37,11.85,LEAD pb,207.19,11
.34,BISMUTH bi,208.98,9.8,POLONI
UM po,(210),,ASTATINE at,(210),,
RADON rn,(222),0.00973,FRANCIUM
fr,(223),,RADIUM ra,(226),5.0
355 DATA ACTINIUM ac,(227),,THOR
IUM th,232.038,11.2,PROTACTINIUM
 pa,(231),,URANIUM u,238.03,18.7
,NEPTUNIUM np,(237),,PLUTONIUM p
u,(244),,AMERICIUM am,(243),,CUR
```

```
IUM cm,(247),,BERKELIUM bk,(247)
,,CALIFORNIUM cf,(251),,EINSTEIN
IUM es,(254),
360 DATA FERMIUM fm,(257),,MENDE
LEVIUM md,(258),,NOBELIUM no,(25
5),,LAWRENCIUM lr OR lw,(256),,R
UTHERFORDIUM rf,(261),,HAHNIUM h
a,(262),
365 REM ***ENTER FOLLOWIN8 DATA
EXACTLY AS SHOWN, INCLUDING SPAC
ES***
370 DATA 1,2,2   1,2   2,2   3,2
   4,2   5,2   6,2   7,2   8,2
 8   1,2   8   2,2   8   3,2   8
   4,2   G   5,2   8   6,2   8
 7,2   8   8,2   8   8   1,2   8
   8   2,2   8   9   2,2   8  1Ø
   2,2   8  11   2,2   8  13   1
375 DATA 2   8  13   2,2   8  14
   2,2   8  15   2,2   8  16   2
,2   8  18   1,2   8  18   2,2
 8  18   3,2   8  18   4,2   8
18   5,2   8  18   6,2   8  18
 7,2   8  18   8,2   8  18   8
 1,2   8  18   8   2,2   8  18
 9   2
380 DATA 2   8  18  1Ø   2,2   8
  18  12   1,2   8  18  13   1,2
   8  18  13   2,2   8  18  15
 1,2   8  18  16   1,2   8  18
18,2   8  18  18   1,2   8  18
18   2,2   8  18  18   3,2   8
18  18   4,2   8  18  18   5
385 DATA 2   8  18  18   6,2   8
  18  18   7,2   8  18  18   8,2
   8  18  18   8   1,2   8  18
18   8   2,2   8  18  18   9   2
,2   8  18  2Ø   8   2,2   8  18
  21   8   2,2   8  18  22   8
 2,2   8  18  23   8   2
390 DATA 2   8  18  24   8   2,2
   8  18  25   8   2,2   8  18
25   9   2,2   8  18  27   8   2
,2   8  18  28   8   2,2   8  18
  29   8   2,2   8  18  3Ø   8
 2,2   8  18  31   8   2,2   8
18  32   8   2,2   8  18  32   9
   2
395 DATA 2   8  18  32  1Ø   2,2
   8  18  32  11   2,2   8  18
32  12   2,2   8  18  32  13   2
,2   8  18  32  14   2,2   8  18
  32  15   2,2   8  18  32  17
 1,2   8  18  32  18   1,2   8
18  32  18   2,2   8  18  32  18
   3
400 DATA 2   8  18  32  18   4,2
   8  18  32  18   5,2   8  18
```

```
32  18   6,2   8  18  32  18   7
,2   8  18  32  18   8,2   8  18
  32  18   8   1,2   8  18  32
18   8   2,2   8  18  32  18   9
   2,2   8  18  32  18  10   2
405 DATA 2   8  18  32  20   9
 2,2   8  18  32  21   9   2,2
 8  18  32  22   9   2,2   8  18
  32  24   8   2,2   8  18  32
25   8   2,2   8  18  32  25   9
   2,2   8  18  32  26   9   2,2
   8  18  32  28   8   2
410 DATA 2   8  18  32  29   8
 2,2   8  18  32  30   8   2,2
 8  18  32  31   8   2,2   8  18
  32  32   8   2,2   8  18  32
32   9   2,2   8  18  32  32  10
   2,2   8  18  32  32  11   2
415 DATA +,2.1,+1(-1),,,,+,1.0,+
1,2+,1.5,+2,,2.0,+3,,2.5,+4,3-,3
.0,+5(-3),2-,3.5,-2,-,4.0,-1,,,,
+,0.9,+1,2+,1.2,+2,3+,1.5,+3,,1.
8,+4,3-,2.1,+5,2-,2.5,+6(-2),-,3
.0,-1,,,
420 DATA +,0.8,+1,2+,1.0,+2,3+,1
.3,+3,4+,1.5,+4,5+,1.6,+5(+4),3+
,1.6,+6(+3),2+,1.5,+7(+2),3+,1.8
,+3(+2),2+,1.8,+3(+2),2+,1.8,+2,
2+,1.9,+2,2+,1.6,+2,3+,1.6,+3,4+
,1.8,+4,3-,2.0,+3,2-,2.4,+6(-2),
-,2.8,-1,,,,+,0.8,+1,2+,1.0,+2,,
1.2,+3,,1.4,+4
425 DATA ,1.6,+5,,1.8,+6,,1.9,+7
,,2.2,+3,,2.2,+3,2+,2.2,+2,+,1.9
,+1,2+,1.7,+2,3+,1.7,+3,4+,1.8,+
4(+2),3+,1.9,+3,2-,2.1,+6(-2),-,
2.5,-1,,,,+,0.7,+1,2+,0.9,+2,,1.
1,+3,3+,1.1,+3,3+,1.1,+3,3+,1.2,
+3,3+,1.1,+3,3+,1.2,+3
430 DATA 3+,1.1,+3,3+,1.1,+3,3+,
1.2,+3,3+,1.1,+3,3+,1.2,+3,3+,1.
2,+3,3+,1.2,+3,3+,1.1,+3,3+,1.2,
+3,,1.3,+4,,1.5,+5,,1.7,+6,,1.9,
+7,,2.2,,,2.2,,4+,2.2,+4(+2),3+,
2.4,+3,2+,1.9,+2,+,1.8,+3,2+,1.8
,+2,3+,1.9,+3,2+,2.0,,-,2.2,-1,,
,,+,0.7,+1
435 DATA 2+,0.9,+2,3+,1.1,+3,4+,
1.3,,5+,1.5,,6+,1.7,,5+,1.3,,4+,
1.3,,3+,1.3,,3+,1.3,,3+,1.3,,3+,
1.3,,,1.3,,,,,,,,,,,,,,,,,,,,,,,
440 REM SET UP STRINGS FOR TITLE
S
445 R$="BR3":Q$="BR8":D$="BD8BL4
":K$="BR10BU8":S$="BR2":I$="BR16
"
450 T$=L$(80)+R$+L$(101)+R$+L$(1
14)+S$+L$(105)+R$+L$(111)+R$+L$(
100)+R$+L$(105)+R$+L$(99)
455 TT$=L$(84)+R$+L$(97)+R$+L$(9
8)+R$+L$(108)+R$+L$(101)
460 TV$=L$(111)+R$+L$(102)+Q$+L$
(116)+R$+L$(104)+R$+L$(101)
465 TR$=L$(69)+R$+L$(108)+R$+L$(
101)+S$+R$+L$(109)+R$+L$(101)+R$
+L$(110)+R$+L$(116)+R$+L$(115)
470 NG$=L$(78)+R$+L$(111)+R$+L$(
98)+R$+L$(108)+R$+L$(101)+I$+L$(
71)+R$+L$(97)+R$+L$(115)+R$+L$(1
01)+R$+L$(115)
475 AM$=L$(65)+R$+L$(108)+R$+L$(
107)+R$+L$(97)+R$+L$(108)+R$+L$(
105)
480 MT$=L$(77)+R$+L$(101)+R$+L$(
116)+R$+L$(97)+R$+L$(108)+R$+L$(
115)
485 AE$=L$(65)+R$+L$(108)+R$+L$(
107)+R$+L$(97)+R$+L$(108)+R$+L$(
105)+R$+L$(110)+R$+L$(101)+I$+L$
(69)+R$+L$(97)+R$+L$(114)+R$+L$(
116)+R$+L$(104)
490 NM$=L$(78)+R$+L$(111)+R$+L$(
110)+R$
495 OM$=L$(79)+R$+L$(116)+R$+L$(
104)+R$+L$(101)+R$+L$(114)
500 TM$=L$(84)+R$+L$(114)+R$+L$(
97)+R$+L$(110)+R$+L$(115)+R$+L$(
105)+R$+L$(116)+R$+L$(105)+R$+L$
(111)+R$+L$(110)
```

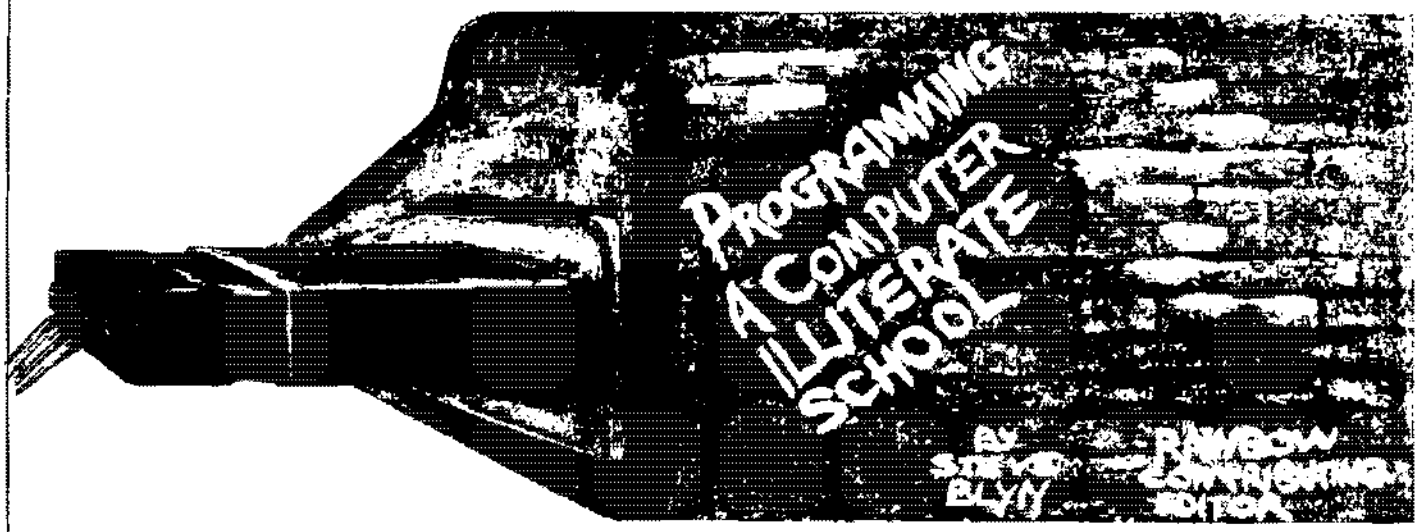
```
505 LA$=L$(76)+R$+L$(97)+R$+L$(1
10)+R$+L$(116)+R$+L$(104)+R$+L$(
97)+R$+L$(110)+R$+L$(105)+R$+L$(
100)+R$+L$(101)+R$+L$(115)
510 AC$=L$(65)+R$+L$(99)+R$+L$(1
16)+R$+L$(105)+R$+L$(110)+R$+L$(
105)+R$+L$(100)+R$+L$(101)+R$+L$
(115):RETURN
515 REM CHECK FOR ELEMENT SYMBOL
520 IFA$="J"ORA$="Q"THEN840
525 IFA$="D"THENX=66ELSEIFA$="U"
THENX=92ELSEIFA$="V"THENX=23ELSE
IFA$="W"THENX=74ELSEIFA$="X"THEN
X=54ELSE535
530 GOTO820
535 IFA$<>"A"THEN550
540 GOSUB845:IFA$="C"THENX=89ELS
EIFA$="G"THENX=47ELSEIFA$="L"THE
NX=13ELSEIFA$="M"THENX=95ELSEIFA
$="R"THENX=18ELSEIFA$="S"THENX=3
3ELSEIFA$="T"THENX=85ELSEIFA$="U
"THENX=79ELSE840
545 GOTO820
550 IFA$<>"E"THEN565
555 GOSUB845:IFA$="R"THENX=68ELS
EIFA$="S"THENX=99ELSEIFA$="U"THE
NX=63ELSE840
560 GOTO820
565 IFA$<>"G"THEN580
570 GOSUB845:IFA$="A"THENX=31ELS
EIFA$="D"THENX=64ELSEIFA$="E"THE
NX=32ELSE840
575 GOTO820
580 IFA$<>"L"THEN595
585 GOSUB845:IFA$="A"THENX=57ELS
EIFA$="I"THENX=3ELSEIFA$="R"THEN
X=103ELSEIFA$="W"THENX=103ELSEIF
A$="U"THENX=71ELSE840
590 GOTO820
595 IFA$<>"M"THEN610
600 GOSUB845:IFA$="D"THENX=101EL
SEIFA$="G"THENX=12ELSEIFA$="N"TH
ENX=25ELSEIFA$="O"THENX=42ELSE84
0
605 GOTO820
610 IFA$<>"R"THEN625
615 GOSUB845:IFA$="A"THENX=88ELS
EIFA$="B"THENX=37ELSEIFA$="E"THE
NX=75ELSEIFA$="H"THENX=45ELSEIFA
$="N"THENX=86ELSEIFA$="U"THENX=4
4ELSEIFA$="F"THENX=104ELSE840
620 GOTO820
625 IFA$<>"T"THEN640
630 GOSUB845:IFA$="A"THENX=73ELS
EIFA$="B"THENX=65ELSEIFA$="C"THE
NX=43ELSEIFA$="E"THENX=52ELSEIFA
$="H"THENX=90ELSEIFA$="I"THENX=2
2ELSEIFA$="L"THENX=81ELSEIFA$="M
```

```
"THENX=69ELSE840
635 GOTO820
640 IFA$<>"Z"THEN655
645 GOSUB845:IFA$="N"THENX=30ELS
EIFA$="R"THENX=40ELSE840
650 GOTO820
655 IFA$<>"B"THEN670
660 GOSUB845:IFA$="B"THENX=5ELSE
IFA$="A"THENX=56ELSEIFA$="E"THEN
X=4ELSEIFA$="I"THENX=83ELSEIFA$=
"K"THENX=97ELSEIFA$="R"THENX=35E
LSE840
665 GOTO820
670 IFA$<>"C"THEN685
675 GOSUB845:IFA$="C"THENX=6ELSE
IFA$="A"THENX=20ELSEIFA$="D"THEN
X=48ELSEIFA$="E"THENX=58ELSEIFA$
="F"THENX=98ELSEIFA$="L"THENX=17
ELSEIFA$="M"THENX=96ELSEIFA$="O"
THENX=27ELSEIFA$="R"THENX=24ELSE
IFA$="S"THENX=55ELSEIFA$="U"THEN
X=29ELSE840
680 GOTO820
685 IFA$<>"F"THEN700
690 GOSUB845:IFA$="F"THENX=9ELSE
IFA$="E"THENX=26ELSEIFA$="M"THEN
X=100ELSEIFA$="R"THENX=87ELSE840
695 GOTO820
700 IFA$<>"H"THEN715
705 GOSUB845:IFA$="H"THENX=1ELSE
IFA$="E"THENX=2ELSEIFA$="F"THENX
=72ELSEIFA$="G"THENX=80ELSEIFA$=
"O"THENX=67ELSEIFA$="A"THENX=105
ELSE840
710 GOTO820
715 IFA$<>"I"THEN730
720 GOSUB845:IFA$="I"THENX=53ELS
EIFA$="N"THENX=49ELSEIFA$="R"THE
NX=77ELSE840
725 GOTO820
730 IFA$<>"K"THEN745
735 GOSUB845:IFA$="K"THENX=19ELS
EIFA$="R"THENX=36ELSE840
740 GOTO820
745 IFA$<>"N"THEN760
750 GOSUB845:IFA$="N"THENX=7ELSE
IFA$="A"THENX=11ELSEIFA$="B"THEN
X=41ELSEIFA$="D"THENX=60ELSEIFA$
="E"THENX=10ELSEIFA$="I"THENX=28
ELSEIFA$="O"THENX=102ELSEIFA$="P
"THENX=93ELSE840
755 GOTO820
760 IFA$<>"O"THEN775
765 GOSUB845:IFA$="O"THENX=8ELSE
IFA$="S"THENX=76ELSE840
770 GOTO820
775 IFA$<>"P"THEN790
780 GOSUB845:IFA$="P"THENX=15ELS
EIFA$="A"THENX=91ELSEIFA$="B"THE
NX=82ELSEIFA$="D"THENX=46ELSEIFA
```

```
$="M"THENX=61ELSEIFA$="O"THENX=8
4ELSEIFA$="R"THENX=59ELSEIFA$="T
"THENX=78ELSEIFA$="U"THENX=94ELS
E840
785 GOTO820
790 IFA$<>"S"THEN805
795 GOSUB845:IFA$="S"THENX=16ELS
EIFA$="B"THENX=51ELSEIFA$="C"THE
NX=21ELSEIFA$="E"THENX=34ELSEIFA
$="I"THENX=14ELSEIFA$="M"THENX=6
2ELSEIFA$="N"THENX=50ELSEIFA$="R
"THENX=38ELSE840
800 GOTO820
805 IFA$<>"Y"THEN840
810 GOSUB845:IFA$="Y"THENX=39ELS
EIFA$="B"THENX=70ELSE840
815 REM PRINT ELEMENT DATA
820 CLS:PRINT@42,NS$(X):PRINT@64
,STRING$(32,"="):PRINT@104,"ATOM
IC NO.: *";X;"*":PRINT@161,"ATOM
.WT.:";AW$(X):PRINT@179,"S.G.:";
SG$(X):PRINT@225,"ION CHARGE:[";
IC$(X);"]":PRINT@242,"ELEC.NEG.:
";EN$(X):PRINT@295,"OXYDATION NO
.: ";OO$(X)
825 PRINT@353,"NO. OF ELECTRONS
IN EACH SHELL":PRINT@387,STRING$
(25,"-"):PRINT@419,"K   L   M
N   O   P   Q":PRINT@451,ES$(X)
830 A$=INKEY$:IFA$=""THEN830ELSE
IFA$=" "THEN835ELSE520
835 SCREEN1,0:ONZ GOTO105,115,12
5,135,145,155,165,175,185
840 CLS0:PRINT@234,"ENTRY ERROR"
;:FORT=1TO400:NEXTT:SCREEN1,0:ON
Z GOTO105,115,125,135,145,155,16
5,175,185
845 A$=INKEY$:IFA$=""THEN845ELSE
RETURN
850 CLS:PRINT"    THE FIRST SCREE
N SHOWS THE     COMPLETE PERIODIC
 TABLE.  TO     SEE THE EIGHT SEP
ARATE CLASSES   OF ELEMENTS, SIMP
LY PRESS        THE spacebar EACH
 TIME."
855 PRINT:PRINT" WHEN VIEWING an
y SCREEN, YOU   MAY RETRIEVE IN
FORMATION ON    any ELEMENT SIM
PLY BY TYPING   THE ELEMENT'S S
YMBOL.":PRINT
860 PRINT" N.B.: WITH SINGLE-LET
TER         ELEMENTS, YOU MAY HAV
E TO TYPE   THE same LETTER TWICE
.":PRINT:PRINT"  PRESS ANY KEY T
O CONTINUE";
865 A$=INKEY$:IFA$=""THEN865
870 CLS:PRINT:PRINT" FOR EXAMPLE
, TO VIEW THE DATA   ON HYDROGEN
, TYPE <HH>. THIS    IS ONLY NEC
ESSARY WHEN OTHER    ELEMENTS SH
ARE THE SAME FIRST   LETTER. IN
OTHER CASES, A       SINGLE LETT
ER WILL SUFFICE."
875 PRINT" FOR EXAMPLE, TO SEE T
HE DATA     ON URANIUM, SIMPLY TY
PE <U> AS   NO OTHER ELEMENT STAR
TS WITH     THAT LETTER. FOR THE
SAME        REASON, SOME 2-LETTER
 ELEMENTS   WILL REQUIRE THAT ONL
Y THE       FIRST LETTER BE TYPED
 IN."
880 PRINT:PRINT"  PRESS ANY KEY
TO CONTINUE";
885 A$=INKEY$:IFA$=""THEN885
890 CLS:PRINT:PRINT" THIS APPLIE
S TO ELEMENTS dy AND xe AS NO OT
HER ELEMENT SHARES   THEIR FIRST
 LETTERS.":PRINT
895 PRINT" WHEN VIEWING DATA ON
AN ELEMENT YOU MAY RETURN TO THE
 TABLE BY   PRESSING THE spacebar
 OR YOU     MAY ENTER ANOTHER SYM
BOL.":PRINT
900 PRINT" NOTE: A NUMBER IN PAR
ENTHESES    (ATOMIC WEIGHT) INDIC
ATES THE    MASS NUMBER OF THE MO
ST STABLE   ISOTOPE."
905 PRINT@484,"PRESS ANY KEY TO
CONTINUE";
910 A$=INKEY$:IFA$=""THEN910
915 CLS:PRINT@41,"ABBREVIATIONS"
:PRINT@73,STRING$(13,"-"):PRINT@
98,"ATOM.WT.= ATOMIC WEIGHT":PRI
NT@162,"S.G.= SPECIFIC GRAVITY":
PRINT@226,"ELEC.NEG.= ELECTRONEG
ATIVITY"
920 PRINT:PRINT"    N.B.: VALUES
GIVEN FOR 'ION      CHARGE' AND '
OXYDATION NO.'      FOR CERTAIN E
LEMENTS ARE         THOSE REPRESE
NTING THE MOST      COMMON OR STA
BLE STATE(S)."
925 PRINT@484,"PRESS ANY KEY TO
START";
930 A$=INKEY$:IFA$=""THEN930ELSE
50
```

EDUCATION NOTES

16K ECB



Many of you readers as well as some of my teaching colleagues have asked what one teacher can do to help introduce computers to his local school. I will share in this month's article some of my own personal experiences in the hope that you will be getting some additional ideas and be inspired to use your resources to help spread the computer revolution.

The junior high school that I teach in is located in the famous Flatbush area of Brooklyn, New York. This is the neighborhood that the movie "Sophie's Choice" was filmed in. Although built to accommodate 1,200 students, the building now houses over 1,700. The students are of diverse ethnic backgrounds: Over 70 percent of the students are minority students. The neighborhood still retains its famous middle-class character.

This story starts in January, 1982. My junior high school was still in the Computer Dark Ages. Not only were there no computers in the school yet, but there were no plans to acquire any. None of the math teachers owned or planned to own home computers. I was the only one interested in this field, but I am a Special Education teacher. What, you ask, could one person do alone?

I am, however, in a unique position in the school. I have been teaching Special Education here for 13 years, am friendly with many of the 100-plus teachers, and have the full cooperation and mutual respect of all of the administrators. I am therefore able to try innovative ideas and introduce new projects with no difficulty. Although my license would not permit me to teach computer literacy full time in the school, I felt that I could influence the actions of others.

It has long been obvious to me that a mathematical equation exists between the amount of experience a teacher or administrator has and his degree of resistance to change. This is not meant to cast aspersions on these professionals. I am part of them and know that it is merely the result of their being witness to many "unique cure-alls" over the years that often amounted to a waste of time only to be replaced by another magic "cure-all."

I realized that it would be necessary to enlist the aid of many of the other experienced professionals on the staff to introduce some sort of computer program to the school. Direct didactic methods would not stand a chance either by me or the administrators. I was certain that computers could not be forced upon this professional staff. Another, more humanistic approach would have to be taken.

An open door policy would have to be established by myself, with the principal's consent, to create an awareness of a computer's potential as a teaching aid in our school.

I decided to start out slowly. Let's not overwhelm or scare anyone off early in the game. I requested permission of the principal to bring my TRS-80 Color Computer to school on a once a week basis. I used some of the educational software that I have developed at home as learning material for my students. Computer programs or software that help to reinforce teaching are known as Computer Assisted Instruction (C.A.I.).

My class is the Special Education Resource Room at this school. I teach small groups of learning disabled students. The students immediately saw the computer to be a novel way of learning new and reviewing old material. They enjoy using the C.A.I. materials and several have even begun to read and write very simple programs. They, of course, love to read each new issue of *the Rainbow*.

Each of the assistant principals, the principal, the special education supervisor of the district, the school's guidance counselors, and the P.T.A. president were invited separately to view these lessons. Teachers were invited to drop in at their convenience any time on any Friday. Friday soon became known as Computer Day in my Resource Room and to a number of the staff in the school.

Two of the school's math teachers took an immediate liking to the computer. They came to visit every Friday and quickly grasped the possibilities for its use with math classes. By coincidence, the school was given some money by the district in March. This money had to be spent immediately

(Steve Blyn teaches both exceptional and gifted children, holds two master's degrees and has won awards for the design of programs to aid the handicapped. He and his wife, Cheryl, own Computer Island.)

by the math department. They decided to use the money to buy computer equipment.

Unfortunately, I could not in good conscience recommend to them the purchase of CoCos at that time. There was a scarcity of educational software on the market, no Network, and a new and unproven disk drive. I therefore suggested that they use their money to begin with four TRS-80 Model III computers, one disk drive and a printer. There was much software available for the Model IIIs then.

The equipment was ordered and arrived quickly. There was, however, no way that the math department could begin a program with the computers and students in the middle of a term. It was decided that the two math teachers who were interested in computers and the principal would take these computers home until September to practice and learn more about them. The fourth computer with the disk drive and printer would be placed in my Resource Room for the rest of the term.

This represented a big opportunity for me to involve more people in computers. I thanked the school for lending my class the computer and volunteered to teach a group of students from the school's math team about computers.

These students were the very brightest math students in our school. We met each day during my preparation period and their lunch period. They quickly learned how to operate the computer, disk drive and printer and how to save and load in programs. This is referred to as computer literacy. I also began to teach this group simple programming commands and routines. This was an introduction for them into real programming. They soon picked up enough elementary programming ability and the skills to begin to write simple programs on their own as a group.

I also volunteered to write some programs which would generate class lists for the school. This would be a large time saver for many people as it is very difficult to keep track of all the kids in the school. The computer could easily print lists of children by official class, subject class, minor groups, shops and many other ways. Just about everyone volunteered to type in the data in order to gain experience with the "new toy." This acted as the school's introduction to data processing programs.

This data program actually served to win over more advocates for computers than all of my teaching efforts. The assistant principals, deans, counselors, attendance teacher and several other people with administrative jobs quickly saw the advantage of such lists to make their jobs easier. After all of the pupils' names were entered into the program, lists were generated on the printer and given to anyone who needed them. Many teachers used these lists to help them enter their final grades.

By the opening of the fall term in September, 1982, the P.T.A. had already decided to contribute the money for four more Model III computers and a Network. This meant that there were now eight computers available for a single classroom. The Network allows the teacher to upload and download programs from the disk computer to the other seven computers. This equipment was clearly enough to really begin a computer literacy course for entire classes. One of the math teachers who had borrowed a computer over the summer became the teacher of this class. I willingly helped out with some demonstration lessons. Since September, five classes a day have been successfully receiving computer literacy and programming instruction.

Just before the summer, our Gates program received a sum of $2,000 to spend on instructional materials. The Gates program has remedial classes for children who are below grade level in reading or math. The assistant principal in charge of this program decided to spend that money on computers. More software and peripherals had been put on the market by then and I recommended that they purchase three TRS-80 Color Computers, three black and white monitors and one printer. I recommended TRS-80 Color Computers for them because of the inexpensive price, and the musical and graphic capabilities of this computer. Also, I had developed much educational software which I would donate to their program. The computers arrived just before summer vacation. They were also taken home over the summer by several teachers to practice and get a head start on learning the possibilities of their new machines.

In the fall, the Gates program began to use their computers. They concentrated mainly on C.A.I. reading programs and word processing. The reading programs were modifications of the ones that I was using in the Resource Room.

They were also very interested in using the word processor and printer as a means of putting out better class magazines. Since the students are obtaining much more professional looking results, their stories and articles are becoming both longer and better.

Since the Gates classes were not using their computers on a regular basis, they were kind enough to let one reside in my room. This, of course, again saved me the trouble of bringing in my own computer. The assistant principal of this program was so impressed with the Color Computer that he bought one for his own personal use at home.

All proceeded on this course until March. The school received some unexpected funds and the principal decided to let the Resource Room have its own computer. A TRS-80 64K Color Computer and disk drive were purchased. My mother always says, "when it rains, it pours." As an added bonus to my class's program, one of the original math teachers was kind enough to donate a used color television to my class for use as a monitor. After demonstrating and using my Color Computer in school for over one year with a black and white television, I could finally utilitize the color capabilities. Color, of course, adds a pleasant extra motivational factor to all of the programs.

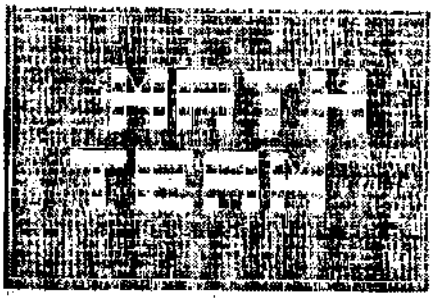
The school's principal and the program office chairman were both very impressed throughout this entire period by the data processing abilities of computers. The program office is the place that figures out teacher, class, room, lunchtime, shop and many other teacher and student schedules. They asked many questions about the possibilities of a computer becoming an aid to programming and printing many of the repetitive and administrative tasks in the school. The principal told the local school board of his desire to purchase a computer to help alleviate and quicken much of the office work.

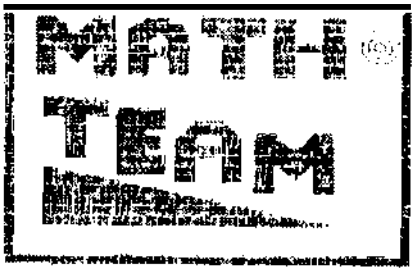
The school board was so impressed with this idea that they appropriated enough money to purchase a very sophisticated computer system. An IBM computer with one hard and two floppy disk drives and a printer have recently been ordered for administrative purposes. Several special software packages to do advanced word processing and to create data bases have also been ordered. We expect to have this equipment up and running by September, 1983.

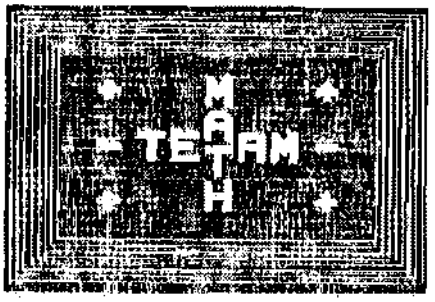
I feel that the method of approaching the original problem was correct. If I, or anyone else, had tried to force the introduction of computers into our school in a dogmatic or threatening fashion, there would have been much resistance. The "let's do it together" approach that I took and the willingness of the principal to let many faculty members borrow the equipment over the summer vacation were both very important and successful ideas.

We now have successful computer programs on several levels. Our school is no longer computer illiterate. I feel much personal satisfaction in the part I played and continue to play in this situation.

I have included three short programs written by several members of the eighth grade math team. We had been working together for almost a year and decided to have a contest to create a logo for their team. Although very bright, these students are not too advanced in the use of high resolution graphics because of their limited access time to the computer. I taught them the uses of the line command and let them use their abilities to create the programs that follow. The results are especially pleasing to me as none of these kids have a home computer of their own on which to practice.







240.... 023D
430.....0446
END... 069E

Listing 1:

```
10 REM"TORI"
20 PCLS:SCREEN 1,1:PMODE 4,1
30 COLOR 7
40 FOR Y= 1 TO 8
50 LINE(4+T,4+T)-(253-T,188-T),P
SET,B
60 T= T+4
70 NEXT Y
80 LINE (60,52)-(64,64),PSET,BF
90 LINE(56,56)-(68,60),PSET,BF
100 LINE(56,94)-(68,98),PSET,BF
110 LINE(60,128)-(64,140),PSET,B
F
120 LINE (56,132)-(68,136),PSET
,BF
130 LINE(192,52)-(196,64),PSET,B
F
140 LINE (188,56)-(200,60),PSET,
BF
150 LINE (188,94)-(200,98),PSET,
BF
160 LINE(192,128)-(196,140),PSET
,BF
170 LINE(188,132)-(200,136),PSET
,BF
180 LINE(120,48)-(124,68),PSET,B
F
```

```
190 LINE (132,48)-(136,68),PSET,
BF
200 LINE(124,48)-(128,56),PSET
210 LINE(124,60)-(124,64),PSET
220 LINE(128,56)-(132,48),PSET
230 LINE (128,64)-(132,60),PSET
240 LINE(124,60)-(128,64),PSET
250 PAINT (128,60),7,7
260 LINE(120,76)-(124,92),PSET,B
F
270 LINE(132,76)-(136,92),PSET,B
F
280 LINE(124,80)-(132,84),PSET,B
F
290 LINE(124,72)-(132,76),PSET ,
BF
300 LINE (120,76)-(124,72),PSET
310 LINE(124,78)-(126,76),PSET
320 LINE (130,76)-(132,78),PSET
330 LINE(132,72)-(136,76),PSET
340 LINE(120,96)-(136,100),PSET,
BF
350 LINE(126,100)-(130,116),PSET
,BF
360 LINE(120,120)-(124,140),PSET
,BF
370 LINE(120,128)-(136,132),PSET
,BF
380 LINE(132,120)-(136,140),PSET
,BF
390 LINE(80,86)-(96,90),PSET,BF
400 LINE(86,86)-(90,106),PSET,BF
410 LINE(100,86)-(104,106),PSET,
BF
420 LINE(100,86)-(116,90),PSET,B
F
430 LINE(100,94)-(112,98),PSET,B
F
440 LINE(100,102)-(116,106),PSET
,BF
450 LINE(144,86)-(150,90),PSET,B
F
460 LINE(152,88)-(156,106),PSET,
BF
470 LINE(140,88)-(144,106),PSET,
BF
480 LINE(140,94)-(156,98),PSET,B
F
490 LINE (152,86)-(156,88),PSET
500 LINE(140,88)-(144,86),PSET
510 LINE(144,92)-(146,90),PSET
520 LINE(150,90)-(152,92),PSET
530 LINE(160,86)-(164,106),PSET,
BF
540 LINE(172,86)-(176,106),PSET,
BF
550 LINE(164,86)-(168,92),PSET
560 LINE(168,92)-(172,86),PSET
570 LINE(154,98)-(168,102),PSET
580 LINE(168,102)-(172,98),PSET
590 PAINT(167,95),7,7
600 LINE(122,74)-(134,74),PSET
610 LINE(122,75)-(134,75),PSET
620 LINE(142,87)-(152,87),PSET
630 LINE(144,86)-(152,86),PSET
640 LINE(144,88)-(152,88),PSET:L
INE(144,89)-(152,89),PSET:LINE(1
44,90)-(152,90),PSET
650 GOTO650
```

Listing 2:

```
170.....0282
END... 052E
```

```
10 REM"BOB"
20 PCLS:SCREEN1,1:PMODE4,1
30 COLOR 7
40 LINE(24,16)-(24,56),PSET:LINE
-(36,56),PSET
50 LINE(36,56)-(36,36),PSET:LINE
-(48,48),PSET:LINE-(60,36),PSET
60 LINE(60,36)-(60,56),PSET:LINE
-(72,56),PSET:LINE-(72,16),PSET:
LINE-(60,16),PSET
70 LINE(60,16)-(48,28),PSET:LINE
-(36,16),PSET:LINE-(24,16),PSET
80 PAINT(64,28),3,7
90 LINE(80,24)-(80,56),PSET:LINE
-(92,56),PSET:LINE-(92,44),PSET
100 LINE(92,44)-(104,44),PSET:LI
NE-(104,56),PSET:LINE-(116,56),P
SET
110 LINE(116,56)-(116,24),PSET:L
INE-(104,16),PSET:LINE-(92,16),P
SET:LINE-(80,25),PSET
120 LINE(92,36)-(104,36),PSET:LI
NE-(104,32),PSET:LINE-(99,27),PS
ET:LINE-(92,32),PSET:LINE-(92,36
),PSET
130 PAINT(93,37),3,7
140 DRAW "BM124,16;R39;D13;L13;D
29;L13;U29;L13;U13"
150 PAINT(130,20),3,7
160 DRAW"BM168,16;R13;D17;R17;U1
7;R13;D41;L  13;U17;L17;D17;L13;
U41"
170 PAINT(176,28),3,7
180 DRAW"BM28,72;R39;D13;L13;D29
;L13;U29;L13;U13"
190 PAINT (48,80),3,7
200 DRAW"BM72,80;R30;D13;L13;D5;
R9;D9;L9;D5;R13;D13;L29;U45"
210 PAINT (84,90),3,7
220 DRAW"BM108,100;D30;R12;U17;R
16;D17;R12;U30"
230 LINE(148,100)-(136,88),PSET:
LINE-(120,88),PSET:LINE-(108,100
),PSET
240 DRAW"BM120,100;D5;R16;U5"
250 LINE(136,100)-(132,96),PSET:
```

```
LINE-(124,96),PSET:LINE-(120,100
),PSET
260 PAINT(128,92),3,7
270 DRAW"BM168,96;L12;D41;R12;U2
1"
280 LINE(168,116)-(180,128),PSET
:LINE-(192,116),PSET
290 LINE(168,96)-(180,108),PSET:
LINE-(192,96),PSET
300 DRAW"BM192,96;R12;D41;L12;U2
1"
310 PAINT(196,116),3,7
320 LINE(28,120)-(28,160),PSET:L
INE-(204,160),PSET:LINE-(28,120)
,PSET
330 PAINT(56,144),3,7
340 LINE(5,5)-(250,186),PSET,B
350 LINE(10,10)-(245,181),PSET,B
360 PAINT (7,7),3,7
370 FOR T= 1 TO 4:CIRCLE(232,36)
,Y:Y=Y+4:NEXT
380 GOTO 380
```

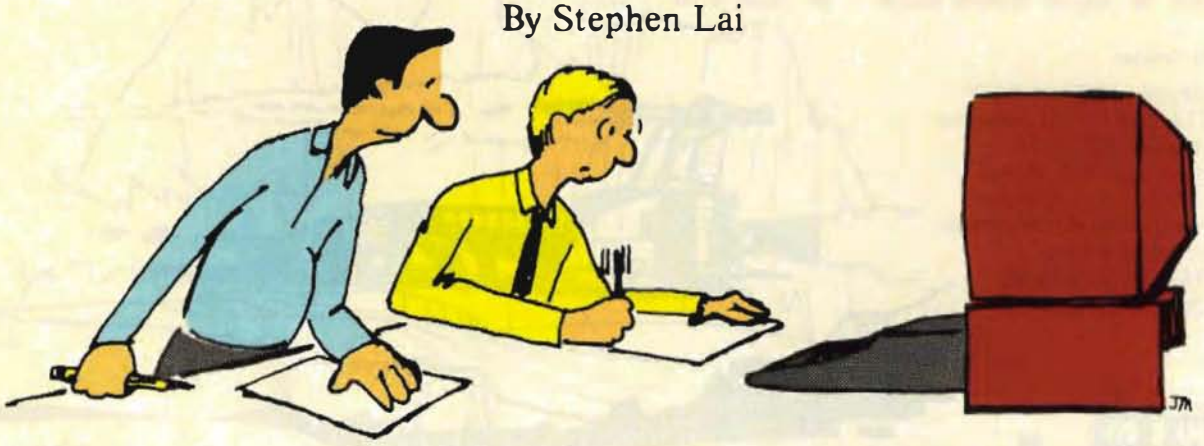
240.....0247
430.....0481
END... 06D2

Listing 3:

```
10 REM "J&E"
20 PCLS:SCREEN 1,1:PMODE 4,1
30 CLS
40 LINE (64,44)-(72,52),PSET,BF
50 LINE(88,44)-(96,52),PSET,BF
60 LINE(104,44)-(136,52),PSET,BF
70 LINE(144,44)-(176,52),PSET,BF
80 LINE(184,44)-(192,52),PSET,BF
90 LINE(208,44)-(216,52),PSET,BF
100 LINE(64,60)-(96,68),PSET,BF
110 LINE(104,60)-(136,68),PSET,B
F
120 LINE(156,60)-(164,68),PSET,B
F
130 LINE(184,60)-(216,68),PSET,B
F
140 LINE(64,76)-(72,84),PSET,BF
150 LINE(88,76)-(96,84),PSET,BF
160 LINE(104,76)-(112,84),PSET,B
F
170 LINE(128,76)-(136,84),PSET,B
F
180 LINE(156,76)-(164,84),PSET,B
F
190 LINE(184,76)-(192,84),PSET,B
F
200 LINE(208,76)-(216,84),PSET,B
F
210 LINE(64,52)-(72,60),PSET,B
220 LINE(88,52)-(96,60),PSET,B
230 LINE (104,52)-(112,60),PSET,
B
240 LINE(128,52)-(136,60),PSET,B
250 LINE(156,52)-(164,60),PSET,B
260 LINE(184,52)-(192,60),PSET,B
270 LINE(208,52)-(216,60),PSET,B
280 LINE(64,68)-(72,76),PSET,B
290 LINE(88,68)-(96,76      ),PSE
T,B
300 LINE (104,68)-(112,76),PSET,
B
310 LINE(128,68)-(136,76),PSET,B
320 LINE(156,68)-(164,76),PSET,B
330 LINE(184,68)-(192,76),PSET,B
340 LINE(208,68)-(216,76),PSET,B
350 LINE(72,44)-(80,60),PSET:LIN
E -(88,44),PSET
360 LINE(40,100)-(72,108),PSET,B
F
370 LINE(80,100)-(112,108),PSET,
BF
380 LINE (120,100)-(152,108),PSE
T,BF
390 LINE(160,100)-(168,108),PSET
,BF
400 LINE(184,100)-(192,108),PSET
,BF
410 LINE(52,116)-(60,124),PSET,B
F
420 LINE(80,116)-(108,124),PSET,
BF
430 LINE(120,116)-(152,124),PSET
,BF
440 LINE(160,116)-(192,124),PSET
,BF
450 LINE(52,132)-(60,140),PSET,B
F
460 LINE(80,132)-(112,140),PSET,
BF
470 LINE(120,132)-(128,140),PSET
,BF
480 LINE(144,132)-(152,140),PSET
,BF
490 LINE(160,132)-(168,140),PSET
,BF
500 LINE(184,132)-(192,140),PSET
,BF
510 LINE(52,108)-(60,116),PSET,B
520 LINE(80,108)-(88,116),PSET,B
530 LINE(120,108)-(128,116),PSET
,B:LINE(144,108)-(152,116),PSET,
B:LINE(160,108)-(168,116),PSET,B
:LINE(184,108)-(192,116),PSET,B
540 LINE(52,124)-(60,132),PSET,B
:LINE(80,124)-(88,132),PSET,B:LI
NE(120,124)-(128,132),PSET,B:LIN
E(144,124)-(152,132),PSET,B:LINE
(160,124)-(168,132),PSET,B:LINE(
184,124)-(192,132),PSET,B
550 LINE(168,100)-(176,116),PSET
:LINE-(184,100),PSET
560 LINE (4,4)-(252,188),PSET,B
570 GOTO 570
```

Boggle Clone Demands Concentration

By Stephen Lai



After playing several rounds of Boggle, a popular word game made by Parker Brothers, with my friends and family, I decided to write a program patterned after it. In the proeess I learned much about arrays, bubble sorting and randomizing.

Each player will need a pencil and paper to play this game. Three minutes are given for one to eight players to find as many hidden words as possible in a 4 x 4 layout of randomly selected letters. Words are formed by linking horizontally, vertically, and diagonally adjacent letters together. No single letter may be used more than once, but if two identical letters are both located in the layout, both may be used. Any word that can be found in a standard English dictionary may be used, with the exception of proper nouns and words of less than three letters.

More thorough instructions are provided in the program.

Instead of using *DRAW* statements to produce the 26 letters of the alphabet on the graphics screen, I used data which is *POKEd* into the appropriate memory locations on the screen. I picked the black/green mode over the black/buff mode to make the undocumented colors more inconspicuous.

Here is a detailed description of the program, line by line:

LINES **DESCRIPTION**

10: Sets up arrays for graphic representation of the 26 letters of the alphabet (AZ), the 16 six-lettered cubes (CB$) and the 16 cells in which the cubes will be randomly placed.

20: Sets graphic screen to highest resolution; clears the screen black; skips over lines 30—80 to get to line 90.

30—40: Subroutine that waits for the user to depress the "ENTER" key.

50—70: DATA for the AZ array.

80: DATA for the CB$ array.

90: READs the data from lines 50—80.

100—110: Sets up DRAW statements for "1 MINUTE," "30 SECONDS," "10 SECONDS," and "ADD UP POINTS."

120—190: Introduction: asks five questions important to game play.

200—210: Initial screen set up.

220—230: Randomly places the 16 cubes into the 4 x 4 layout (I used the method discussed by Steve Blyn in the February 1983 issue of *the Rainbow*, pages 14—16); displays graphics screen, initiates the three minute timer.

240—320: Timer runs; subroutines appropriately executed to display and erase signals that warn players that one minute, 30 seconds, and 10 seconds are left for finding words.

330—340: Buzzer SOUNDs to signal that the three minute time limit has expired; players are told to ADD UP POINTS; the computer waits for the "ENTER" key to be pressed.

350—370: Asks each player the amount of points earned in the round; adds this to the total from all previous rounds; the computer checks to see if anyone has equalled or surpassed the specified score or if the specified number of rounds has been played, depending on the INPUTted choice.

(Stephen Lai is a high school freshman in Palatine, Illinois. His primary interests are the color computer, jogging, tennis, and carnivorous plants.)

380—410: Displays everyone's updated scores.
420: Players are informed that another round is to be played; a GOTO 220 starts the formation of a new random layout.
430—490: Final scores are displayed, from highest to lowest using a bubblesort; players are asked whether or not they wish to play another game.
500—630: Instructions; uses the subroutine located at lines 30—40 to flip seven pages of text onto the screen.

```
90..... 03B2
190.... 077F
290.... 0AF5
470.... 0E83
550.... 134B
END... 177A
```

The listing:

```
10 DIMAZ(25,13),CB$(15),N(16):TC
=119:PMODE4
20 PMODE4:CLS0:GOTO90
30 PRINT@481,STRING$(30,128);:PR
INT@485,"HIT <ENTER> WHEN READY"
;
40 A$=INKEY$:IFA$<>CHR$(13)THEN4
0ELSECLS:RETURN
50 DATA8,20,34,65,127,65,65,60,1
8,18,28,18,18,60,28,34,32,32,32,
34,28,60,18,18,18,18,18,60,62,32
,32,60,32,32,62,63,32,32,62,32,3
2,32,30,32,32,38,34,34,28,65,65,
65,127,65,65,65,28,8,8,8,8,8,28,
2,2,2,2,2,34,28,67,68,88,96,80,7
6,67
60 DATA32,32,32,32,32,32,63,65,9
9,85,73,65,65,65,65,97,81,73,69,
67,65,28,34,65,65,65,34,28,60,34
,34,60,32,32,32,28,34,34,34,42,3
6,26,60,34,34,60,40,36,34,28,34,
16,8,4,34,28,62,8,8,8,8,8,8,65,6
5,65,65,65,65,62,65,65,65,65,34,
20,8
70 DATA65,65,65,73,85,99,65,65,3
4,20,8,20,34,65,65,34,20,8,8,8,8
,62,2,4,8,16,32,62,124,130,130,1
30,130,138,134,126,1,1,1,72,72,4
8
80 DATAAACIOT,ABILTY,ABJOMQ,ACDE
MP,ACELRS,ADENVZ,AHMORS,BFIORX,D
ENOSW,DKNOTU,EEFHIY,EGINTV,EGKLU
Y,EHNIPS,ELPSTU,GILRUWY
90 FORF1=0TO25:FORF2=0TO12STEP2:
READAZ(F1,F2):AZ(F1,F2+1)=AZ(F1,
F2):NEXTF2,F1:FORF1=0TO13:READAZ
(16,F1):NEXT:FORF1=0TO15:READCB$
(F1):NEXT
100 SE$="R2EU4HL2GD4BR12FR2EUHL2
HUER2NFBR5NR4D3NR3D3R4BR4R2EUBU2
UHL2GD4BR6RDLU":M1$="BM103,162E2
D6NL2R2BR6U6F2E2D6BR4U6BR4ND6F4D
2U6BR4D5FR2EU5BR4R2ND6R2BR4NR4D3
NR3D3NR4":S3$="BM109,160R4D3NL3D
3NL4BR3XSE$;":S1$="BM109,162E2D6
NL2R2BR3XSE$;"
110 AP$="BM88,137U4E2F2DNL4D3BR3
;U6R2F2D2G2NL2BR5;U6R2F2D2G2NL2B
R9NU6R4U6BR3;D3ND3R3EUHNL3BR7;D3
ND3R3EUHNL3BR4D6R4U6L4BR7D6BR3U6
F4D2U6BR3R2ND6R2BR6NFL2GDFR2FDGL
2H"
120 CLS:PRINT@76,"BOGGEL":PRINT@
174,"BY":PRINT@233,"STEPHEN LAI
":PRINT@386,"DO YOU NEED INSTRUC
TIONS";:INPUTA$:IFA$="Y"THENGOSU
B500130 CLS:PRINT@66,"HOW MANY P
LAYERS (1-8)";:INPUTPL:IFPL<1ORP
L>8THEN130ELSEDIMSC(PL),NA$(PL),
FS(PL):FORF1=1TOPL:FS(F1)=F1:NEX
T
140 FORF1=1TOPL:CLS:PRINT:PRINT"
WHAT IS YOUR NAME, PLAYER"F1
150 INPUTA$:IFLEN(A$)>9THENPRINT
"PLEASE, UNDER 10 LETTERS";:GOTO
150ELSENA$(F1)=A$:NEXT
160 CLS:PRINT@69,"DO YOU WANT TO
 PLAY BY";:PRINT@133,"(S)CORE OR
 (R)OUND";:INPUTC$:IFC$="R"THEN
C=2:GOTO180ELSEIFC$<>"S"THEN160
170 C=1:PRINT@225,"WHAT SCORE DO
 YOU WISH TO PLAY UP
```

```
 TO";:INPUTHS:GOTO190
180 PRINT@225,"HOW MANY ROUNDS D
O YOU WISH TO                PLAY";
:INPUTRO
190 CLS0
200 PCLS:FORF1=0TO64STEP4:LINE(F
1,0)-(191+F1,192),PSET:LINE(F1,1
92)-(191+F1,0),PSET:NEXT:FORF1=6
8TO235STEP4:A=255-F1:A1=F1-64:LI
NE(F1,0)-(256,A),PSET:LINE(A,0)-
(0,A),PSET:LINE(0,A1)-(A,192),PS
ET:LINE(256,A1)-(F1,192),PSET:NE
XT
210 LINE(100,156)-(156,170),PRES
ET,BF:LINE(36,123)-(220,144),PRE
SET,BF:LINE(37,124)-(219,143),PS
ET,B
220 A=RND(-TIMER):LINE(65,9)-(18
4,109),PRESET,BF:L=16:FORF1=1TO1
6:N(F1)=F1:NEXT:FORF1=1TO16:A=RN
D(L):T=N(A):N(A)=N(L):L=L-1:POKE
999+F1,T:NEXT:FORF1=1TO16:N(F1)=
PEEK(999+F1):NEXT
230 FORF1=0TO3:FORF2=0TO3:A=N(F1
*4+F2+1)-1:Q=ASC(MID$(CB$(A),RND
(6),1))-65:FORF3=0TO13:POKE2025+
F1*768+F2*4+F3*32,AZ(Q,F3):NEXTF
3,F2,F1:LINE(38,125)-(218,143),P
SET,BF:PLAY"T25L25BAGFEDCDEFGAB"
:SCREEN1,0:TIMER=0
240 TI=INT(TIMER/60):IFTI>TC GOS
UB260
250 TT=INT(TI/30):T2=TI-TT*30:LI
NE(38+T2*6,125+TT*3)-(44+T2*6,12
7+TT*3),PRESET,BF:GOTO240
260 ONCH+1 GOTO270,280,290,300,3
10,320,330
270 DRAWM1$:FORF1=1TO2:FORF2=0TO
1:SCREEN1,F2:PLAY"O5T255L255BFA"
:NEXTF2,F1:SCREEN1,0:CH=1:TC=124
:RETURN
280 DRAW"C0XM1$;C5":CH=2:TC=149:
RETURN
290 DRAWS3$:FORF1=1TO2:FORF2=0TO
1:SCREEN1,F2:PLAY"O3FCG":NEXTF2,
F1:SCREEN1,0:CH=3:TC=154:RETURN
300 DRAW"C0XS3$;C5":CH=4:TC=169:
RETURN
310 DRAWS1$:FORF1=1TO2:FORF2=0TO
1:SCREEN1,F2:PLAY"O2EDA":NEXTF2,
F1:SCREEN1,0:CH=5:TC=172:RETURN
320 DRAW"C0XS1$;C5":CH=6:TC=179:
RETURN
330 SOUND1,20:CH=0:TC=119:DRAWAP
$:RP=RP+1
340 A$=INKEY$:IFA$<>CHR$(13)THEN
340
350 CLS:PRINT@75,"ROUND #"RP;:FO
RF1=1TOPL:PRINT@160,NA$(F1)",":P
RINT"HOW MANY POINTS DID YOU GET
";:INPUTA:SC(F1)=SC(F1)+A:NEXT:O
NC GOTO360,370
360 FORF1=1TOPL:IFSC(F1)=>HS THE
N430ELSENEXT:GOTO380
370 IFRO=<RP THEN430
380 A=1
390 CLS:FORF1=1024TO1119:POKEF1,
128:NEXT:PRINT@10,"ROUND #"RP;:P
RINT@65,"PLAYER";:PRINT@90,"SCOR
E";:PRINT@96,STRING$(32,"-");
400 FORF1=A TOA+5:IFPL<F1 THEN42
0ELSEPRINT@64+(F1-A+1)*64,NA$(F1
)+STRING$(26-LEN(NA$(F1)),"-");:
PRINT@92+(F1-A+1)*64,SC(F1);:NEX
T
410 IFPL>A+5 THENA=A+6:GOTO390
420 GOSUB30:PRINT"THE CUBES ARE
BEING SHUFFLED FOR           THE N
EXT ROUND":GOTO220
430 CLS7:PRINT@234,"FINAL SCORE"
;:FORF1=1TO3000:NEXT:IFPL=1THENC
LS:PRINT@64,NA$(1)",":PRINT"YOUR
 FINAL SCORE WAS"SC(1)"POINTS IN
"RP"ROUNDS.":PRINT:PRINT:GOTO490
440 A=0
450 A=0:FORF1=PL TO2STEP-1:IFSC(
FS(F1))>SC(FS(F1-1))THENA=1:A1=F
S(F1):FS(F1)=FS(F1-1):FS(F1-1)=A
1
```

```
460 NEXT:IFA=1THEN450
470 A=1
480 CLS:PRINT"   PLAYER";:PRINT@
26,"SCORE":PRINT@32,STRING$(32,"
-"):FORF1=A TOA+5:IFPL<F1 THENGO
SUB30:GOTO490ELSEPRINT@(F1-A+1)*
64,NA$(FS(F1)):PRINT@26+(F1-A+1)
*64,SC(FS(F1)):NEXT:GOSUB30
490 IFPL>A+5THENA=A+6:GOTO480ELS
EPRINT"DO YOU WANT TO PLAY AGAIN
 (Y/N)":INPUTA$:IFA$="Y"THENRUNE
LSEEND
500 CLS:PRINT@73,"INSTRUCTIONS":
PRINT:PRINT"  EACH PLAYER WILL N
EED A PAPER AND PENCIL.":PRINT:P
RINT"  THE OBJECT OF THIS GAME I
S TO LIST AS MANY WORDS AS POSSI
BLE  WITHIN A GIVEN TIME LIMIT.
WHEN A WORD A FOUND, IT IS WRITT
EN   DOWN."
510 GOSUB30
520 PRINT:PRINT"  WHEN THE TIMER
 STARTS, EACH   PLAYER SEARCHES
FOR WORDS OF    THREE LETTERS OR
 MORE. THE TIME LIMIT FOR FINDIN
G AS MANY WORDS AS POSSIBLE IS T
HREE MINUTES.   SIGNALS ARE SOUN
DED AND FLASHED WHEN ONE MINUTE,
 30 SECONDS, AND";
530 PRINT"10 SECONDS ARE LEFT.":
PRINT:PRINT"  THE 'Q' IS FOLLOWE
D BY A 'U'  FOR THE REASON THAT
'Q' IS AL-  WAYS FOLLOWED BY A '
U' IN THE   ENGLISH LANGUAGE. 'Q
U' COUNTS ASTWO SEPERATE LETTERS
.";:GOSUB30
540 PRINT:PRINT"  WORDS ARE FORM
ED BY LINKING   ADJACENT LETTERS
 TOGETHER. LET- TERS MUST JOIN I
N PROPER SE-    QUENCE TO SPELL
A WORD. THEY MAYJOIN HORIZONTALL
Y, VERTICALLY,  OR HORIZONTALLY.
"
550 PRINT:PRINT"  NO SINGLE LETT
ER MAY BE USED  MORE THAN ONCE A
 WORD, BUT IF   THERE ARE TWO LE
TTERS OF THE    SAME TYPE IN DIF
FERENT POSITIONSOF THE GRID, BOT
H MAY BE USED.":GOSUB30
560 PRINT:PRINT"  ANY WORD, WITH
 THE EXCEPTION  OF PROPER NOUNS,
 IS ACCEPTABLE, AS LONG AS IT CA
N BE FOUND IN A STANDARD ENGLISH
 DICTIONARY.":PRINT:PRINT"  PLUR
AL AND SINGULAR NOUNS CAN BE WRI
TTEN DOWN AS SEPERATE     WORDS,
 AND BOTH WILL RECIEVE    ";
570 PRINT"FULL CREDIT.":GOSUB30
580 PRINT:PRINT"  WHEN THE TIMER
 STOPS, EVERYONESTOPS SEARCHING
FOR WORDS. ONE  AT A TIME, EACH
PLAYER READS HIS LIST OF WORDS.
IF ONE OR MORE  OTHER PEOPLE CON
TAIN THAT EXACT WORD ON THEIR LI
ST, THE WORD IS CROSSED OUT FROM
 ANY LIST THAT  ";
590 PRINT"CONTAINS THAT WORD.":G
OSUB30
600 PRINT:PRINT"  THE COMPUTER W
ILL SIGNAL YOU  TO 'ADD UP POINT
S' . THE SCORING GOES AS FOLLOWS
:":PRINT@161,"WORD LENGTH":PRINT
@192,"  3  4  5  6  7  8 OR MORE
":PRINT@257,"POINTS":PRINT@288,"
   1  1  2  3  5  11"
610 PRINT:PRINT"  ONLY COUNT WOR
DS ON YOUR LIST THAT HAVEN'T BEE
N CROSSED OFF.  AFTER YOU ADD UP
 YOUR POINTS FORTHE ROUND, PRESS
 <ENTER>":GOSUB30
620 PRINT:PRINT"  A GAME IS WON
AFTER A PLAYER  HAS EITHER EQUAL
LED OR TOPPED A CHOSEN HIGH SCOR
E OR BY THE PER-SON WITH THE HIG
HEST SCORE AFTERA CHOSEN NUBER O
F ROUNDS.":PRINT:PRINT@224,"----
--END-OF-INSTRUCTIONS-------";
630 GOSUB30:RETURN
```

SUPERPATCH for EDTASM+

By Roger Schrag

Roger Schrag updates his popular Patch to disk for Radio Shack's Editor Assembler Program.

In the December issue of *the Rainbow*, I presented a program that would patch Radio Shack's EDTASM+ cartridge to store programs on disk instead of on tape. Since then I have added a whole host of new features to my patch program. Now I would like to present *Super Patch*, an entirely new patch program. It is shown in Listing 1.

What does *Super Patch* do, you ask? Lots of things. It patches the Editor so that you may load or write source code to disk. It also allows you to assemble your programs directly onto disk, as well as verify any disk file.

Super Patch modifies ZBUG so that you may load a block of memory from disk, either normally or offset. It also allows you to write any block of memory to disk, as well as display a file directory of any drive on the screen or the printer.

Super Patch also provides many minor features and conveniences, namely a repeating keyboard, custom cursor and automatic selection of the default disk drive and printer baud rate.

To add all these features to EDTASM+, we must make it reside in low memory. Transferring it from ROM to RAM allows us to alter sections of code, and to insert whole new routines. Unfortunately, moving EDTASM+ to RAM eats up a lot of our memory that could otherwise be used for program storage. On a normal 32K Color Computer you will only have about 17K of storage space for your source code.

However, this brings us up to *Super Patch's* most powerful feature. It supports 64K! If you have installed the 64K modification in your computer, or if you purchased your machine after approximately October, 1982, *Super Patch* will automatically use all of your system's memory, giving you about 49K of space for your programs. But don't fret if your computer doesn't have the 64K ability—*Super Patch* will still work fine.

Here's how to use *Super Patch* to create your enhanced version of EDTASM+: First, insert your EDTASM+ cartridge and turn on your computer. When the sign-on message appears, go into ZBUG and enter U C000 1000 27FF. This will copy EDTASM+ into low memory, where the final patched program will reside. Dump a copy of it onto tape by entering P EDTASM 1000 37FF 1000.

Next, return to the Editor and enter the source code shown in Listing 1. Notice the section labeled Program Customization. There are several decisions you will have to make based on your own personal preferences and the equipment you have. Let's look them over:

Your first option is the cursor format. The program as it appears in the listing will give you a red blinking cursor. However, the comment lines show three other types of cursor you may use. Let's suppose that you would rather have a solid black cursor than a blinking red one. You would insert asterisks to make lines 20 and 21 into comment lines, and remove the asterisks from lines 26 and 27.

The next option is the default drive. Any time you are asked for a filename and you don't specify a drive number, the default drive will be used. Since I only have one drive, zero must be my default. However, if you have two drives, you might find it more convenient to have drive one as the default.

The next option is the status of the verification system. As listed, the program will verify every sector that is ever written on the diskette. This is a good precautionary measure, but it slows down write operations by about 50 percent. The choice between speed and safety is all yours. This won't

---

(Roger Schrag, a high school senior, enjoys working with the CoCo and writing for ***the Rainbow****. He also designs and translates programs for Adventure International.)*

affect the speed of read operations, such as loading source code.

The next option is the printer's Baud rate. If you have a Radio Shack printer, you will probably want to select 600 Baud. But if you have a printer that doesn't operate at the standard 600 Baud, you may use this option so that you won't have to *POKE* the Baud rate constant each time you want to use your printer.

The remaining options relate to the repeating keyboard. You may want to try various combinations for RATE1 and RATE2 until you find what is most comfortable for you. Some people may not want certain keys to repeat, so the final option allows you to keep the *ENTER*, *BREAK*, and *CLEAR* keys from repeating.

After you have selected all of your options, enter the rest of the source code shown in Listing 1 exactly as it appears. Assemble it onto tape directly after the copy of EDTASM+ you made from ZBUG. Save the source code right after that. You may want to look it over or modify it at some future time.

Now you are ready to put everything on disk, so you will need to connect your disk controller. Please turn off your computer when you remove the cartridge and insert the controller. You should always play it safe and shut off the computer when you exchange cartridges.

Rewind your cassette tape all the way and turn on the computer. When the Disk BASIC sign-on message appears, enter *CLOADM* to load the RAM version of EDTASM+ saved from ZBUG, but don't execute it yet. When OK appears, type *CLOADM* again to load in the assembled patch program. Don't execute it, either. The patch program will overwrite various sections of EDTASM+ as it loads, and a block of new code will be added on.

When OK reappears, you are ready to save the final product onto disk by entering SAVEM"EDTASM",&HE00,&H37FF,&HE00. Save as many copies on different diskettes as you like.

Now enter the program in Listing 2 from Disk BASIC and save it on your diskettes right along with EDTASM+. This is a utility to transfer source code files from tape to diskette. You will probably want to transfer the source code from Listing 1 right away, before you forget which tape you put it on.

Your *Super Patched* EDTASM+ is now ready to use. Simply type LOADM"EDTASM":EXEC and press *ENTER* to load it. It will reside in memory from $E00 to $37FF, and use memory from $3800 to $39FF for internal use. All memory from $3A00 on up to $7FFF, or $FEFF on a computer with the 64K capability, will be available for your program storage.

Super Patched EDTASM+ is completely position independent, so you may load it offset. For example, the command LOADM"EDTASM",&H1000:EXEC will load EDTASM+ into memory from $1E00 to $47FF, and memory from $E00 to $1DFF will be left untouched. You could use this memory for graphics pages, assembling directly into memory, or whatever you wish.

The L, W, V, and A commands of the Editor have all been modified. To load source code from diskette, press L and *ENTER*. You will be prompted to enter the filename. Any filename valid in BASIC is valid here, but don't enclose it in quotes. Also, if you don't include an extension, none will be used. An extension isn't mandatory, but I would recommend that you use the extension /SRC for all source code files you create.

Finally, if you don't specify a drive number, the default drive will be used. Once you enter the filename, the source code will be loaded.

The W and V commands work in much the same way. All the V command does is merely open the file and immediately close it. This verifies that a file does indeed exist, and that there aren't any serious problems in the diskette's directory.

To assemble a program, type A and any assembly switches you would like to use. Then press *ENTER*. Unless you specify the IM or NO switches, you will be prompted for the filename. You should give any program you assemble the extension /BIN, but, as always, no extension will be assumed if you don't specify one.

As lines of assembled code scroll across the screen, the computer will pause every now and then to write a sector to the diskette. You may press *BREAK* to abort the assembly, but you will have to wait a few seconds for the computer to first close the disk file.

A word of caution is in order. *Never* press the Reset button while the red light on any of your drives is on. If you were to press Reset while the computer was writing a sector on the diskette, you would probably have to reinitialize the entire diskette with *DSKINI*. This applies to all situations, not just when using EDTASM+.

The L, P, and V commands of ZBUG have also been modified. About the last one, the V command, a word is in order. The unpatched version of EDTASM+ on cartridge has a V command in the Editor and a V command in ZBUG. Both do the exact same thing: verify a tape file. As we have already discussed, the Editor's V command has been patched to verify a disk file. ZBUG's V command, however,

has been transformed into a whole new command for displaying file directories. More on that later.

To load a machine language program or block of memory from ZBUG, press L and *ENTER*. You will be asked for the filename and the program will be loaded. You may specify an offset directly after the L, if you wish. For example, L 100 will load the program offset by $100, or 256 bytes.

To put a block of memory onto diskette, press P followed by the starting address, the ending address, and the execution address. You will be asked for the filename and the block of memory will be written. This command may be used to duplicate programs (no pirating please), save blocks of data, and so forth. As an example, the command P 5000 6000 5200 will write everything in memory from $5000 through $6000 onto diskette. The program will begin execution at $5200.

The F command will display a directory of all files on a particular diskette. To display the directory on the screen, type in FS and number of the drive containing the diskette. To print up the directory on your printer, type in FP and the drive number. In either case, you may leave out the drive number and the default will be used.

You may press "shift @" to freeze the display just as you can with the *DIR* statement in BASIC, but *don't* press *BREAK!* Doing so will return you to BASIC. If you forget and do press BASIC, press Reset to reenter ZBUG.

From ZBUG, you are not able to examine the BASIC and disk system ROM's. Instead, you are able to examine the upper block of 32K RAM that usually isn't accessable from BASIC. If your computer doesn't have the 64K capability, looking at this upper block of memory may give strange results.

We've now covered just about all of the features provided by *Super Patch*. To summarize, only the L, W, V, and A commands of the Editor, and the L, P, and F commands of ZBUG have been modified. Everything else should operate as it always has, except for the new cursor and repeating keyboard.

Now I have a puzzler question for you. If you look at addresses $E25C through $E269 of the unpatched EDTASM+ cartridge, here is what you will find:

```
LEAX  0BE33,PCR
LDA   18E
STA   ,X
LDU   18F
STU   1,X
```

This routine is executed by ZBUG just before printing the BRK @ message whenever a breakpoint is encountered. Does anyone have any ideas as to why Microsoft would make ZBUG attempt to write to the Color BASIC ROM?

This article represents the start of a new series of articles about somewhat more advanced applications of assembly language. If you have any topic suggestions, or if you have any questions about *Super Patch*, feel free to drop me a line. My address appears at the top of the first listing. Please include a self addressed, stamped envelope, and I will respond as promptly as possible.

Listing 1:

```
00001 ****************************
00002 *  SUPER PATCHED EDTASM+   *
00003 ****************************
00004 *
00005 *By: Roger Schrag
00006 *    2054 Manning Avenue
00007 *    Los Angeles, CA 90025
00008 *
00009 *Final revision: 5/05/83
00010 *
00011 *** Program Customization ***
00012 *For each option, alternatives are
00013 *shown below in comment lines. Feel
00014 *free to substitute these for your
00015 *own) alternatives according to your
00016 *personal preferences
00017 *
00018 *
00019 *Cursor format
00020 CURSOR  EQU     $60     Red cursor
00021 BLINK   EQU     $9F     Blinking
00022 *CURSOR EQU     $FF     Red cursor
00023 *BLINK  EQU     $00     Non blinking
00024 *CURSOR EQU     $60     Black cursor
00025 *BLINK  EQU     $20     Blinking
00026 *CURSOR EQU     $80     Black cursor
00027 *BLINK  EQU     $00     Non blinking
00028 *
```

```
00029 *
00030 *Which drive is the default drive
00031 DEFALT  EQU     $0      Drive zero
00032 *DEFALT EQU     $1      Drive one
00033 *
00034 *
00035 *Whether or not all write operations
00036 *should automatically be verified
00037 VERIF   EQU     $FF     Verify everything
00038 *VERIF  EQU     $00     Don't verify
00039 *
00040 *
00041 *Printer's baud rate
00042 BAUD    EQU     $1      9600 baud
00043 *BAUD   EQU     $57     600 baud
00044 *BAUD   EQU     $BE     300 baud
00045 *
00046 *
00047 *How long you must hold down a key
00048 *before it starts repeating
00049 RATE1   EQU     $1E     Half a second
00050 *RATE1  EQU     $0F     Quarter of a second
00051 *RATE1  EQU     $3C     One whole second
00052 *
00053 *
00054 *Rate at which the keys repeat
00055 RATE2   EQU     $03     Medium
00056 *RATE2  EQU     $02     Fast
00057 *RATE2  EQU     $06     Slow
```



```
00058 *
00059 *
00060 *Which keys have the repeating feature
00061 REPRES  EQU     $7F     Every key repeats
00062 *REPRES EQU     $3F     Every key repeats
00063 *       EXCEPT: Enter, Clear, and Break
00064 *
00065 *
00066 *Change the sign-on message
00067         ORG     $1058
00068         FCC     /SUPER PATCHED EDTASM+ 1.0/
00069         FCB     $0D     Carraige return
00070         FCC     /(C) BY MICROSOFT/
00071         FCB     $A0     Terminator
00072 *
00073 *
00074 *Fix an apparent bug in Zbug
00075         ORB     $2D4E
00076         LEAX    $326A,PCR  Load SWI vector
00077 *
00078 *
00079 *Change the filename handling system to
00080 *allow the user to enter full disk file
00081 *names (in the FILENAME/EXT:D format)
00082         ORG     $15E0
00083         LBRA    FNAME
00084 *
00085 *
00086 *Change references to device #-1 (tape)
00087 *to device #1 (disk)
00088         ORG     $1547
00089         LDB     #1
00090         ORG     $1574
00091         LDB     #1
00092         ORG     $15AD
00093         LDA     #1
00094         ORG     $15CD
00095         LDA     #1
00096 *
00097 *
00098 *Make the Editor close the disk file
00099 *when access to it is complete
00100         ORG     $1B52
00101         LBSR    ECLOSE
00102 *
00103 *
00104 *Make Zbug close the disk file
00105 *when access to it is complete
00106         ORG     $2D88
00107         LBSR    ZCLOSE
00108 *
00109 *
00110 *At the right time, call a routine
00111 *to open the disk file, instead of
00112 *a routine to open the tape file
00113         ORG     $15B7   Patches into
00114         LEAU    ASSEM,PCR
00115         BRA     $15D7   the "A" command
00116         ORG     $15BF   Patches into
00117         LDU     #LOAD   the "L" command
```

```
00118          ORG     $15C4   Patches into
00119          LDU     #WRITE  the "W" command
00120          ORG     $1637   Patches into
00121          LEAU    LOAD,PCR
00122          BRA     $15D7   the "V" command
00123 *
00124 *
00125 *Alter the error handling routine to
00126 *account for new error messages
00127          ORG     $1721
00128          LBSR    ERROR   Compute error code
00129          CLR     $1      Clear some flags
00130          CLR     $4
00131          LEAX    ERRS,PCR List of messages
00132          LBSR    $121E   Print error message
00133 *
00134 *
00135 *Fix input routine so that if you
00136 *respond to "FILENAME?" prompt by
00137 *hitting Break, you will be returned
00138 *to the Editor or Zbug properly
00139          ORG     $1EF0
00140          LBEQ    $172F
00141 *
00142 *
00143 *Make Zbug's "P" command write
00144 *to disk, by using the ROM routine
00145 *that normally processes the Basic
00146 *statement "SAVEM"
00147          ORG     $16C9
00148          CLR     $FFDE   Select ROMs
00149          JMP     $CEA2   Use ROM routine
00150          ORG     $31AD   Eliminate a tape
00151          NOP             Filename check
00152          NOP
00153          NOP
00154          NOP
00155          NOP
00156 *
00157 *
00158 *Make Zbug's "L" command load
00159 *from disk, by using the ROM routine
00160 *that normally processes the Basic
00161 *statement "LOADM"
00162          ORG     $163F   The offset is in U
00163          PSHS    U       Put offset on stack
00164          LDU     #LOAD   Get the filename and
00165          LBSR    $15C9   Open file for input
00166          PULS    X       Retrieve the offset
00167          PSHS    DP      Save Direct Page
00168          CLRA            Clear the Direct
00169          TFR     A,DP    Page register
00170          STX     $D3     Store the offset
00171          CLR     $FFDE   Select ROMs
00172          JSR     $CF07   Use ROM routine
00173          CLR     $FFDF   Turn on 64K
00174          PULS    DP,PC   Restore DP & return
00175          ORG     $31A4   Eliminate a tape
```

```
00176         RTS             Filename check
00177 *
00178 *
00179 *Change Zbug's "V" command into an "F"
00180 *command which does a disk directory
00181         ORG     $2EAB   Zbug's command table
00182         FDB     DIR-$2D4E  Address of command
00183         FCC     *F*     Change "V" to "F"
00184 *
00185 *
00186 *Wipe out some calls to tape related
00187 *routines in ROM that are no longer needed
00188         ORG     $14FD
00189         NOP
00190         NOP
00191         NOP
00192         ORG     $1628
00193         RTS
00194 *
00195 *
00196 *Alter the keyboard input routine so
00197 *that we may define a custom cursor
00198         ORG     $1519
00199         LDB     #CURSOR Get cursor character
00200         EORB    #BLINK  Blink it if necessary
00201 *
00202 *
00203 *Make EDTASM+ use our routines instead
00204 *of those contained in the ROM
00205         ORG     $154C
00206         LBSR    INPUT   Input from a device
00207         ORG     $169C
00208         LBSR    INPUT
00209         ORG     $155D
00210         LBSR    CLS     Clear the screen
00211         ORG     $1524
00212         LBSR    KBSCAN  Scan the keyboard
00213         NOP
00214         ORG     $1589
00215         LBSR    KBSCAN
00216         NOP
00217         ORG     $157E
00218         LBSR    OUTPUT  Output to a device
00219         NOP
00220 *
00221 *
00222 *Modify "Q" command to reenable ROMs
00223 *before attempting to return to Basic
00224         ORG     $1505
00225         CLR     $FFDE   Select ROMs
00226         CLR     $FF40   Turn off disk drive
00227         JMP     $A027   Jump into Basic
00228 *
00229 *
00230         ORG     $E00
00231 *
00232 *
00233 *New entry point to EDTASM+
00234 START   LEAX    $38FF,PCR  Find bottom of mem
00235         TFR     X,D     Round off to even
00236         CLRB            Increment of $100
00237         STD     $FF     Set memory pointer
00238         LEAX    NMI,PCR Use our own interrupt
00239         STX     $10A    Service routines
00240         LEAX    IRQ,PCR Instead of those
00241         STX     $10D    In the ROM
00242         LBSR    $1558   Clear the screen
00243         LDB     #VERIF  Set up disk system's
00244         STB     $9B7    Verification system
00245         LDB     #DEFALT Select which drive
00246         STB     $95A    Will be the default
00247         LDX     #BAUD   Select the printer's
00248         STX     $95     Serial baud rate
00249         LBRA    $1005   Enter EDTASM+
00250 *
00251 *
00252 *Subroutines called by the program
00253 *patches to enable the new features
00254 *
00255 *
00256 *Close any open disk files
00257 CLOSE   CLR     $FFDE   Select ROMs
00258         JSR     $CA3B   Use ROM routine
00259         CLR     $FFDF   Turn on 64K
00260         RTS             Return
00261 *
00262 *
00263 *Patches for the Editor and Zbug
00264 EDCLOSE BSR     CLOSE   Close files
```

```
00265          LBRA    $1377   Return to the Editor
00266 ZCLOSE   BSR     CLOSE   Close files
00267          LBRA    $1387   Return to Zbug
00268 *
00269 *
00270 *Routine to allow you to enter a name
00271 *for the disk file. Also, the address
00272 *in U is offset so that this program
00273 *will remain entirely position independent.
00274 FNAME    TFR     U,D     Put address in D
00275          LEAU    $0,PCR  Calculate the offset
00276          LEAU    D,U     Add it to the address
00277          PSHS    U       Save the address
00278          LEAX    PROMPT,PCR "FILENAME?"
00279          LBSR    $1221   Print the prompt
00280          CLR     $16     Clear input counter
00281          LBSR    $1EC6   Get user's input
00282          LDB     $12     Get length of input
00283          LDX     $13     Get address of input
00284          PSHS    DP      Save the Direct Page
00285          CLRA            Clear the Direct
00286          TFR     A,DP    Page register
00287          LEAY    BACK,PCR Save a return
00288          PSHS    Y       Address on the stack
00289          CLR     ,-S     Lower the stack
00290          LDA     $95A    Select the default
00291          STA     $E8     Drive number
00292          LDY     #$94C   Name storage area
00293          LDA     #$20    ASCII space
00294 ERASE    STA     ,Y+     Clear out the
00295          CMPY    #$957   Filename storage
00296          BNE     ERASE   Area
00297          CLR     $FFDE   Select ROMs
00298          JMP     $C8A4   Use ROM routine
00299 BACK     CLR     $FFDF   Turn on 64K
00300          PULS    DP,U,PC Restore and return
00301 PROMPT   FCC     *FILENAME?*
00302          FCB     $A0     Terminator
00303 *
00304 *
00305 *Routines to open a disk file.
00306 *Note the slight differences for
00307 *Load, Write, and Assemble.
00308 LOAD     LDA     #$49    (I)nput mode
00309          LDX     #$1FF   ASCII format
00310          BRA     OPEN    Open the file
00311 WRITE    LDA     #$4F    (O)utput mode
00312          LDX     #$1FF   ASCII format
00313          BRA     OPEN    Open the file
00314 ASSEM    LDA     #$4F    (O)utput mode
00315          LDX     #$200   Binary format
00316 OPEN     STX     $957    Store the format
00317          LDX     #$100   The record length
00318          STX     $97C    is 256 bytes
00319          LDB     #$1     Use device #1
00320          CLR     $FFDE   Select ROMs
00321          JSR     $C468   Use ROM Routine
00322          CLR     $FFDF   Turn on 64K
00323          RTS             Return
00324 *
00325 *
00326 *Alter error handling routine to
00327 *handle new disk related errors:
00328 *  DF  VF  NE  WP  FN  FM
00329 *If the error isn't one of those,
00330 *it is assumed to be an IO error.
00331 ERROR    CLR     $FFDF   Turn on 64K
00332          EXG     A,B     Put the code in A
00333          CLRB            Clear B
00334          LEAX    CODES,PCR  List of errors
00335 GETERR   CMPA    B,X     Compare the codes
00336          BEQ     RET     Return if they match
00337          TST     B,X     Check for end of list
00338          BEQ     RET     Return if end of list
00339          INCB            Move to next code
00340          BRA     GETERR  Compare next on list
00341 RET      RTS             Return
00342 *
00343 *
00344 *List of error codes
00345 CODES    FCB     $3B     Disk full
00346          FCB     $48     Verification error
00347          FCB     $34     File not found
00348          FCB     $3C     Write protect
00349          FCB     $3E     Bad filename
00350          FCB     $2A     Bad file mode
00351          FCB     $00     Signal end of list
```

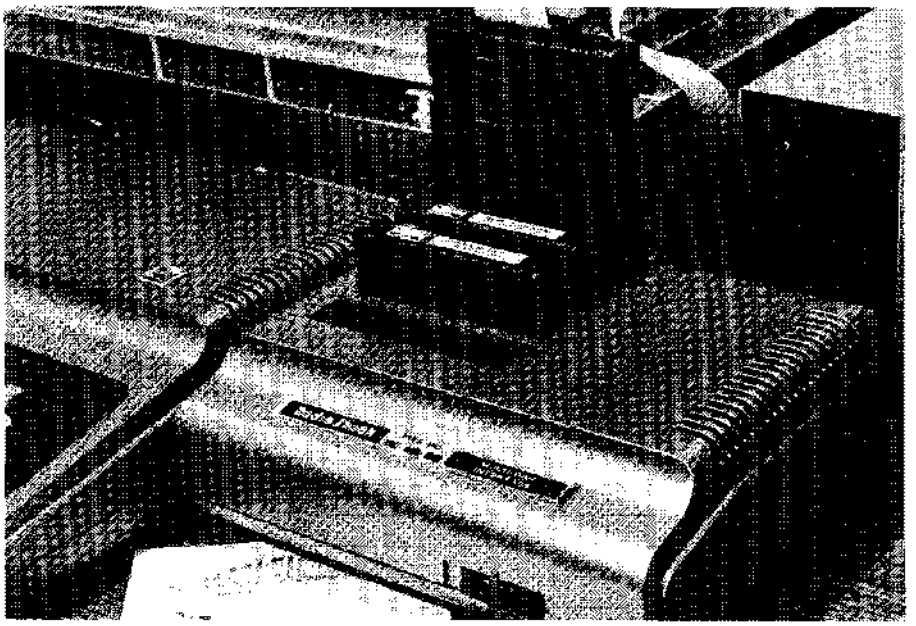
```
00352 *
00353 *
00354 *List of error messages
00355 ERRS    FCC     *DISK FULL*
00356           FCB     $A0     Terminator
00357           FCC     *VERIF. ERROR*
00358           FCB     $A0     Terminator
00359           FCC     *FILE NOT FOUND*
00360           FCB     $A0     Terminator
00361           FCC     *WRITE PROTECT*
00362           FCB     $A0     Terminator
00363           FCC     *BAD FILENAME*
00364           FCB     $A0     Terminator
00365           FCC     *BAD FILE MODE*
00366           FCB     $A0     Terminator
00367           FCC     *I/O ERROR*
00368           FCB     $A0     Terminator
00369 *
00370 *
00371 *Routine to do a disk directory
00372 DIR     LDX     $13     Get user's input
00373           CLRA            Device #0 (screen)
00374           DEC     $12     Was device specified?
00375           BMI     DR1     Skip ahead if not
00376           LDB     ,X+     Examine specification
00377           CMPB    #$50    Was it a "P"?
00378           BNE     DR1     Skip ahead if not
00379           LDA     #$FE    Device #-2 (printer)
00380 DR1     STA     >$6F    Select the device
00381 *The ">" in the above line is crucial!
00382           LDA     $95A    Get the default drive
00383           DEC     $12     Was drive specified?
00384           BMI     DR2     Skip ahead if not
00385           LDA     ,X      Get specification
00386           ANDA    #$3     Convert from ASCII
00387 DR2     PSHS    DP      Save the Direct Page
00388           CLRB            Clear the Direct
00389           TFR     B,DP    Page register
00390           STA     $EB     Select the drive
00391           CLR     $FFDE   Select ROMs
00392           JSR     $CBD2   Do a directory
00393           CLR     $FFDF   Turn on 64K
00394           CLR     $6F     Reselect screen
00395           PULS    DP,PC   Restore and return
00396 *
00397 *
00398 *Routine to input from a device
00399 INPUT   CLR     $FFDE   Select ROMs
00400           JSR     $A176   Use ROM routine
00401           CLR     $FFDF   Turn on 64K
00402           RTS             Return
00403 *
00404 *
00405 *Routine to output to a device
00406 OUTPUT  CLR     $FFDE   Select ROMs
00407           JSR     [$A002] Use ROM routine
00408           CLR     $FFDF   Turn on 64K
00409           RTS             Return
00410 *
00411 *
00412 *Routine to scan the keyboard
00413 KBSCAN  CLR     $FFDE   Select ROMs
00414           ANDCC   #$AF    Enable interrupts
00415           LDA     #$35    Enable the control
00416           STA     $FF03   Register in the SAM
00417           JSR     [$A000] Use ROM routine
00418           PSHS    CC      Save the status flags
00419           CLR     $FFDF   Turn on 64K
00420           PULS    CC,PC   Restore and return
00421 *
00422 *
00423 *Routine to clear the screen
00424 CLS     CLR     $FFDE   Select ROMs
00425           JSR     $A928   Use ROM routine
00426           CLR     $FFDF   Turn on 64K
00427           RTS             Return
00428 *
00429 *
00430 *Non-maskable interrupt service routine
00431 NMI     LDA     $982    Check status flags
00432           BEQ     EXIT    Exit if invalid
00433           LDX     $983    Get transfer address
00434           STX     $0A,S   Put it on the stack
00435           CLR     $982    Clear the flag
00436           RTI             Return from interrupt
00437 *
00438 *
00439 *60 cycle interrupt service routine
00440 IRQ     LDA     $FF03   Check status flags
```

```
00441          BPL     EXIT    Exit if invalid
00442          LDA     $FF02   Reset the latch
00443          LDA     $985    Check drive status
00444          BEQ     REPEAT  Skip if motor is off
00445          DEC     $985    Decrease motor timer
00446          BNE     REPEAT  Skip if not done
00447          LDA     $986    Time to shut off
00448          ANDA    #$B0    Disk drive motor
00449          STA     $986    Update motor status
00450          STA     $FF40   Shut off drive
00451 REPEAT   LDX     #$152   Start of key data
00452 RP1      LDA     ,X+     Check a row of keys
00453          CMPA    #$FF    Are any pressed?
00454          BNE     RP2     Go if key is pressed
00455          CMPX    #$15A   Was that the last row?
00456          BNE     RP1     Loop back if not
00457          INC     KCLEAR,PCR  Increment counter
00458          LDA     KCLEAR,PCR  Has keyboard been
00459          CMPA    #6      Clear for .1 second?
00460          BNE     EXIT    If not, exit routine
00461          CLR     KCLEAR,PCR  Clear the counters
00462          CLR     KHOLD,PCR
00463          BRA     EXIT    Exit the routine
00464 RP2      INC     KHOLD,PCR  Increment counter
00465          LDA     KHOLD,PCR  Has key been held
00466          CMPA    #RATE1  Long enough to repeat?
00467          BNE     EXIT    If not, exit routine
00468          SUBA    #RATE2  Prepare counter for
00469          STA     KHOLD,PCR  The next repeat
00470          LDX     #$152   Start of key data
00471 RP3      LDA     ,X      Check one row
00472          ORA     #REPRES Make all keys seem to
00473          STA     ,X+     Have been re-pressed
00474          CMPX    #$15A   Was that the last row?
00475          BNE     RP3     Loop back if not
00476 EXIT     RTI             Return from interrupt
00477 *
00478 *
00479 *Counters for repeating keys
00480 KCLEAR  FCB     0
00481 KHOLD   FCB     0
00482 *
00483 *
00484 *The key repeat routine is an adaptation
00485 *of a program by Charles Roslund which
00486 *appeared in the Sept. 1982 RAINBOW.
00487 *
00488 *
00489          END     START
```


Listing 2:

```
1 'Utility to transfer EDTASM+
2 'text files from tape to disk
3 CLS
4 PRINT"EDTASM+ SOURCE CODE"
5 PRINT"FILE TRANSFER UTILITY"
6 PRINT STRING$(32,"=")
7 CLEAR500,16308:DEFUSR0=16310
8 FOR X=16310 TO 16318
9 READ Y:POKE X,Y
10 NEXT X
11 DATA 173,159,160,4
12 DATA 173,159,160,6
13 DATA 57
14 PRINT"ENTER THE NAME OF THE"
15 LINEINPUT"TAPE FILE: ";I$
16 PRINT
17 PRINT"ENTER THE NAME OF THE"
18 LINEINPUT"DISK FILE: ";O$
19 AUDIO ON:OPEN"I",#-1,I$
20 OPEN"O",#1,O$:GOTO 22
21 X=USR(0):MOTOR OFF
22 IF PEEK(129)>0 THEN 29
23 IF PEEK(124)=255 THEN 30
24 X$=CHR$(0):X=VARPTR(X$):
25 POKE X,PEEK(125)
26 POKE X+2,1:POKE X+3,218
27 PRINT X$;:PRINT#1,X$;
28 GOTO 21
29 PRINT"?IO ERROR":GOTO 31
30 PRINT"TRANSFER COMPLETE"
31 CLOSE:END
```

This Peripheral Acquaintance Is A Real PAL



By Paul S. Hoffman

There's a new piece of hardware on the block, a real PAL for your TRS-80 Color or TDP-100 computer. The new addition from Tandy Towers in Fort Worth is the Color Computer Multi-Pak Interface (CCMI). It's a PAL in several ways. For one, it'll allow you to switch between four different Program-Pak cartridges without turning off the computer; for another, you can select a slot either through hardware (a four-position switch on the front panel) or software (four different *POKEs* to a single memory location); another reason to call CCMI a PAL is that it contains a chip called a PAL ("Programmable Array Logic"device#PAL 1414, a "semi-custom IC which may be programmed for specific functions" — quote from the CCMI Service Manual).

The unit plugs into the cartridge slot on the right side of the computer, and passes all the signals in parallel to all four slots, *except* for CCS (Cartridge Select Signal) and SCS (Spare Chip Select). These two signals are routed through the select latch, hardware switch and PAL to determine which slot will be operational. Power is provided through a separate cord—one more socket needed in the computer corner!—and, again, we can't tell by a quick look whether the baby is on or off! No power on/off indicator light, I'm sorry to say. The recommended power-up procedure is to turn on *all* peripherals (certainly the Multi-Pak Interface) before turning on the computer. When powering down, turn the computer off first, then all peripheral devices.

Certainly a major advantage to having a device such as this is the ability to have a Disk Controller permanently attached, not needing to constantly "pull the plug" on it to use a ROM Pak or the X-Pad. (I can't tell you how many times I've had a disk crash due to a faulty connection!) But it gives you a bit more flexibility than that, since you have four slots available. With the flip of a switch, I can have:

SLOT 4: Disk BASIC with X-Pad (Disk Controller plugged in here)

SLOT 3: Extended BASIC with X-Pad (X-Pad plugged in here)

SLOT 2: Micropainter

SLOT 1: Anything else—maybe *Poltergeist* for those breaks between work sessions.

Tandy suggests that the Disk Controller (if you have it) go in slot 4 (toward the rear of the Interface). I suspect that this is simply because of the controller's size—it would be quite awkward to reach around it to plug or unplug another Pak. The same thing goes, as far as I'm concerned, with the X-Pad Controller, which is the same size. Put bigger cartridges in back, smaller in front.

Notice that in the configuration I listed above, the X-Pad works both in Disk BASIC and in the slot it's plugged into. It actually functions regardless of which slot has been selected, because it doesn't depend on the two select lines used to pick a slot, and functions in a specially allocated area of memory. At this date, this is the only peripheral sold by Tandy which functions like this. Similar items would be real-time clock ROM Packs currently offered by other companies.

After some experimentation, I've cataloged four major potential uses for the Multi-Pak Interface:

1) Switching to and from Extended BASIC, Disk BASIC and ROM Cartridges.

2) Copying ROM Cartridges to RAM or tape for study/modification.

3) Saving Graphic screens from graphically oriented ROM carts or games.

4) Operating fully-decoded "special I/O" devices (such as X-Pad) simultaneously with Disk.

The software *POKE* for selecting a slot is to location 65407 ($FF7F).

```
POKE 65407,0    for slot 1
POKE 65407,17   for slot 2
POKE 65407,34   for slot 3
POKE 65407,51   for slot 4
```

If you look at how those numbers are represented in hex ($00, $11, $22, $33), you'll see that the high-order nybble (bits 4 through 7) duplicates the low-order nybble (bits 0 through 3). This is because the CCMI uses bits 0 and 1 to determine which slot gets the CTS signal, and bits 4 and 5 to route the SCS signal. Both signals have to go to the same slot for proper selection. The actual address used for this selection process (65407) is determined by the programming of the PAL chip. There seems to have been a last-minute change in that programming, because the Owner's Manual for the Interface lists the address as 65439 ($FF9F). An addenda sheet corrects the address. The strange thing about it is the most of the time 65439 seems to function exactly the same as 65407—writing to it will change the slot, and 65407 will mirror the change. Also, reading either location can tell you which slot has been selected by the switch. If you haven't done a software *POKE*, the address will contain 204 ($CC), 221 ($DD), 238 ($EE) or 255 ($FF) for slots 1, 2, 3 or 4. In other words, the front panel switch fills in bits 2, 3, 6 and 7 as well as creating the proper bit-pattern in 0, 1, 4 and 5.

In general, the hardware switch is the easiest way to switch slots, but frequently it is necessary to press the RESET button on the computer to start a program. Some programs (such as Disk BASIC and the new Deluxe RS-232C Program-Pak) don't automatically send the computer a CART signal and need to be nudged with "EXEC &HC000."

Software selection (by doing the *POKE* to 65407) overrides the switch. Once you've done a *POKE*, you can slide the switch back and forth at will, to no effect. RESET will return control to the switch (unless programming has altered the reset vector).

To look at (for the purpose of study or modification) the program in a ROM Pak, put the cartridge in one slot (WHILE THE COMPUTER AND CCMI ARE OFF), then place the switch in position to select an empty slot (Extended BASIC). After turning the CCMI on and then the computer, do the ROM Pak disable *POKE* (POKE &hff23,36). Then software-select the ROM Pak by *POKE-ing* the appropriate value into 65407. At this point, the ROM Pak program can be accessed starting at address $C000. If you have a RAM-based monitor program or disassembler, you can now load into the computer and examine/disassemble the addresses above $C000. If you have 32K or better, you can move the program down in memory to an unused area of RAM, and then save it to tape for further study or modification.

This is certainly the safest, best way to examine ROM-based programs, since it does *not* involve plugging ROM Paks in while the computer is on. (While testing what the CCMI will and won't do, I stupidly unplugged the X-Pad with power on and zapped my 6809 CPU—I also had the Disk attached to the same slot through a Y-connector and managed to zap the Disk ROM. DON'T PLUG OR UNPLUG ANY CARTRIDGE WITH THE POWER ON!)

The matter of exploring graphics created by ROM Pak programs can be quite interesting. If you play a ROM Pak game or use a graphically oriented ROM Pak program (*Logo, Micropainter, Art Gallery, Graphic Pak*, etc.), you can switch out of the program into an empty slot (using the front-panel switch) and the graphics will be retained in memory. If you do a lot of graphic programming, you might already have been able to identify the mode used in the ROM Pak you've selected. Chances are it'll be either *PMODE 4*, *PMODE 3* or *PMODE 1*. A simple program like Listing 1 will display part of the computer's graphics memory and help you find where the ROM Pak has stored its pictures. The program will run continuously, showing you various sections of Extended BASIC's graphics memory. When you see what looks like the right graphic configuration, press any key. The computer will print on the screen all the pertinent information. If you have a screen print program, you could dump the picture to a printer, or you can save the picture to tape by using the indicated starting and ending addresses in a *CSAVEM* command.

CSAVEM "TITLE," start address, end address, 413 (413 contains a decimal 57—"return from subroutine" code—so that if you accidentally try to *EXEC* a saved picture as though it were a machine language program, the execute address—413—will return the computer to where it was without crashing).

Many ROM Paks will have their graphics area offset slightly from Extended BASIC's. In these cases, you might have to do a "block move" of the memory forward or backward to coincide with one of Extended BASIC's graphic pages. For example, *Art Gallery* pictures end up being in *PMODE 1*, but they start *before* page 1. In fact, they start at $400, the beginning of the text screen. The program in Listing 2 will move the picture down on the screen, but you will not be able to recover the top 512 bytes (the text screen has clobbered them).

Since Extended BASIC's "pages" are 1536 bytes long, you might not get a graphic from a ROM Pak to line up properly with the start of a page. While running Listing 1, look for the

most complete picture, then press any key. If the picture was too high on the screen, key in and run Listing 3. If the picture is still too high, press BREAK and then *RUN* it again.

If the picture is too *low* on the screen, type in and *RUN* Listing 4, repeating as many times as necessary.

All of this business of looking at, moving and copying graphic screens leads naturally into the subject of using the Multi-Pak Interface to link the X-Pad with Disk. An addendum to the CCM1 Owner's Manual says, "Graphic displays created while using the X-Pad . . . can be saved on disk . . . Read the three corresponding values per the X-Pad manual to load the display. Once completed, the finished display can be saved using the *SAVEM* command." The middle sentence of what I've quoted is absolute nonsense—I suppose the reference is to the three addresses the X-Pad uses for X coordinate, Y coordinate and pen status—and the last sentence, while correct, leaves much to be said. Both the *SAVEM* and the *CSAVEM* command need three addresses as parameters: start address, end address and execute address. The X-Pad manual, unfortunately, gives an absolute value for the start of graphics memory ($600) and calculates addresses from that. The problem is that Disk BASIC reorganizes the graphics area, putting it initially 2K higher in memory (starting at $E00). It also could get shoved even higher if you've used the *FILES* command to allocate more disk buffer space. Goodness knows *where* the start of graphics could end up! So how do you find out where it is? The same way Disk BASIC does: You look at the pointers to graphics memory that BASIC sets up.

$BA = start of current graphics page
$B7 = 1 byte past end of current graphics page
$BC = start of graphics memory

This means that the format for saving to disk should be:

SAVEM "TITLE",PEEK(&HBA)*256, *PEEK*(&HB7)*256-1,&13

A picture saved from Extended BASIC (using *CSAVEM*) will most likely (if it is started on graphics page 1) be coded to load back in starting at $600. If you try to load this picture in while Disk BASIC is operational, it will write over Disk BASIC's I/O buffers and only be partially visible on the screen (or not at all if it was in a low-resolution mode). Such a picture must be OFFSET LOADED. The format for the command is:

CLOADM "TITLE", PEEK(&HBC)*256-&H600

The formula after the comma is there to calculate the number of bytes of offset (subtracting the original starting address from the new starting address to get the difference).

On the other hand, pictures saved to tape while running Disk BASIC will need to be offset loaded in a different way if you want to use them *without* the disk. Their addresses are too high (normally by 2K bytes), and the offset has to "wrap-around" the top of memory. The correct format for loading such a picture from tape is:

CLOADM "TITLE", &HC000

To summarize, the CCM1 is a fine buy for the price ($179.95). It has the additional advantage of buffering all of the ROM port connections, leaving your SAM and CPU a bit more protected (though that didn't keep my stupidity from destroying a CPU). Possible disadvantages are that the buffering chips (74LS367s) and the PAL chip are highly susceptible to damage from static electricity. Just make sure you follow the rules of good computer maintenance, and you'll enjoy additional flexibility—and fewer disk crashes due to poor contact!

Listing 1:

```
5 GOTO170
10 FORM=0TO4:'MODE #
20 FORP=1TO5:'PAGE #
30 FORS=0TO1:'SCREEN COLOR SET
40 PMODEM,P:SCREEN1,S
50 FORX=0TO100:'DELAY
60 IFINKEY$<>""THEN120:'LOOK FOR
 KEYPRESS
70 NEXTX
80 NEXTS:'NEXT COLORSET
90 NEXTP:'NEXT PAGE
100 NEXTM:'NEXT MODE
110 GOTO10:'REPEAT
120 CLS:PRINT"PMODE";M;", PAGE";
P
125 PRINT"SCREEN 1,";S
130 PRINT"SCREEN STARTS AT &H";H
EX$(PEEK(&HBA)*256)
140 PRINT"SCREEN ENDS AT &H";HEX
$(PEEK(&HB7)*256-1)
150 END
170 PCLEAR8:GOTO10
```

Listing 2:

```
5 PMODE1,1:SCREEN1,0
10 FORM=&HFFF TO&H600 STEP-1
20 POKE M+&H200,PEEK(M)
30 NEXTM
40 FORM=&H7FF TO&H600 STEP-1
50 POKEM,0:'PAINTS TOP PORTION S
AME AS BORDER COLOR
60 NEXTM
70 GOTO70
```

Listing 3:

```
5 PMODEM,P:SCREEN1,S:'REPLACE TH
E VARIABLES IN THIS LINE WITH TH
E CORRECT MODE #, PAGE # AND COL
ORSET # FOUND BY USING LISTING 1
10 FORX=PEEK(&HB7)*256-1 TO PEEK
(&HBA)*256 STEP-1
20 POKEX,PEEK(X-&H200)
30 NEXTX
40 GOTO40
```

Listing 4:

```
5 PMODEM,P:SCREEN1,S:'REPLACE TH
E VARIABLES IN THIS LINE WITH TH
E CORRECT MODE #, PAGE # AND COL
ORSET # FOUND BY USING LISTING 1
10 FORX=PEEK(&HBA)*256 TO PEEK(&
HB7)*256-1
20 POKEX,PEEK(X+&H200)
30 NEXTX
40 GOTO40
```

Networking For Better Software Evaluation

By Michael Plog, Ph.D.

One major problem with computer use in education (especially the Color Computer) is the lack of knowledge about quality software.

Now, let's get it correct. I am not talking about the lack of *existence* of quality software, just the lack of *knowledge* about quality software.

Software purchase is not a minor concern, especially for schools with severe budget constraints. How do school people make decisions about purchasing software? Several ways.

The first is blind faith. Armed with a limited checkbook, the brave purchaser reads advertisements, looking for small details which might give a clue to the actual operation of a program. A selection is made, purchase order typed, signed through the bureaucratic structure, and in a few weeks, the product is delivered. Sometimes the selection is ideal; it fits the needs of the person ordering it, and works without a single problem. Sometimes, however, the selection is less than perfect. If it works, the stipulated definitions by the author are much different than the concepts used by the person ordering it. Yes, there are snake oil salesmen out there, ready to take your money for a product that might be totally useless.

The second way of ordering software—usually employed after a few negative encounters—is the informal sharing of experience. Other owners of the Color Computer are called; notices are given at club meetings. Sometimes the seeker gets lucky using this method, and someone is found who has already purchased the software in question. Now, the informal sharing provides a single review, but at least by a known source. Since theft is not uncommon among end-users of computers, some purchasers do not have to buy anything, but obtain a copy of a program, often with inadequate instructions.

Occasionally, a third (far superior) method of software selection is used—the *review*. The potential purchaser gets a chance to look at the product, try it out, maybe even get a chance to spend enough time to work completely through the program. The problem with software review is that it is so limited. We can all do it at our local Radio Shack store, for the programs in stock. The problem is that this is a very small slice of available products for the Color Computer. Even if some of you are lucky enough to have a third party software dealer nearby (I am not), selection is still limited. In addition, I have seen teachers and principals reviewing educational software, but I have never seen a teacher bring in students to sit at the keyboard and try the products.

All three of these selection methods—reviews, informal experience sharing, and blind faith—can work well and will continue to be part of the human decision process. There is another way, however. I am recommending the creation of a national network for educational software evaluation to be conducted via *the Rainbow*. It is only coincidence that this notion is given birth with an issue devoted to education. This coincidence may be a good omen for the success of the project.

Say the words aloud once or twice: a national network for educational software evaluation. No cute acronym for the project, but a sincere request for cooperation from readers of *the Rainbow*.

Please notice the evaluation form. Simply put, the idea is for any of you (hopefully, many of you) to complete the form on educational software you have. Completed forms will be collected at a central location, and periodically, results will be provided in *the Rainbow*. Then, the fourth method of software selection (the national evaluation) can be added to the other three methods.

(Michael Plog received his Ph.D. degree from the University of Illinois, the M.S. from Memphis State University, and the B.S. from the University of Tennessee. For his foreign/research language option required for the doctorate, he naturally selected computer language. Michael currently works for the Illinois State Board of Education as a research and evaluation specialist.)

EDUCATIONAL SOFTWARE
EVALUATION FORM
for the COLOR COMPUTER

Name of Software: _____ Price: _____

Marketed From: _____

Minimum System Needed:
- Amount of RAM _____
- Number of Disk Drives _____
- Color Set Needed Y N
- Printer Y N
- Joysticks Y N

Supplied on:
- _____ ROM Pack
- _____ Tape
- _____ Diskette

Written in:
- _____ BASIC
- _____ ML
- _____ Other _____

Purpose (check all that apply)
- ☐ Interactive Instruction
- ☐ Teaching About Computers
- ☐ Creation of Materials
- ☐ Management of Instruction
- ☐ Administrative Management of Information
- ☐ Data Exchange
- ☐ Other _____

Subject Matter (check all that apply)
- ☐ Computer Learning
- ☐ Foreign Language
- ☐ Game/Simulation
- ☐ Language Arts
- ☐ Math
- ☐ Science
- ☐ Social Studies
- ☐ Other _____

Circle All Most Appropriate Classes for use of Software:
Pre-School K 1 2 3 4 5 6 7 8 9 10 11 12 Adult

Rate the Software on each of the following items.
("1" = poorest; "5" = best; "N/A" = Not Applicable)

	Very Poor					Very Good
How Easy to Operate	N/A	1	2	3	4	5
Use of Color	N/A	1	2	3	4	5
Use of Graphics	N/A	1	2	3	4	5
Clarity of Instructions	N/A	1	2	3	4	5
Completeness of Instructions	N/A	1	2	3	4	5
How Well Does It do the Job It is Supposed To	N/A	1	2	3	4	5
Interest Level for Appropriate Classes	N/A	1	2	3	4	5

Major Strengths:

Major Weaknesses:

Your Overall Grade For This Software:
A B C D F

Your relationship to schools:
- ☐ Student
- ☐ Teacher
- ☐ Administrator
- ☐ Parent
- ☐ Other
- ☐ Private Citizen

_____
_____
(Signature/Address)

Mail Completed Form To:
Michael Plog, Ph.D., 829 Evergreen, Chatham, IL 62629

Think of the advantage of this national evaluation method for people trying to buy educational software. Instead of one or two reviews for educational software, many can be available. Instead of trying to outguess a developer's definition of "screen controlled directions easily understood by a first grade student," actual experiences can be examined. For the isolated Color Computer user (and I imagine most of us are), knowledge of people from all parts of the country, and possibly other countries.

I do not want to spend too much time on the benefits of such a system. Let's look for a minute at what evaluation means in this context. An evaluation is a description and *judgment* of a particular software package. Without the judgment, there is no evaluation. (Evaluation's middle name is "value.") Take a look at the evaluation form.

First, the description of the software package. We have to identify what software is being evaluated, thus the blanks for the name and who markets the package, as well as other items such as price, necessary system, and how supplied.

Notice the portion of the form called "Purpose." These are categories of uses of programs. For a more complete explanation of each, dig out your last month's *Rainbow* and check out the article there. In order to get a full description of the software, we need to know the purpose. *Logo*, for example, would have the purpose of "Interactive Instruction" and "Teaching About Computers"; students operate the keyboard and command the computer to do certain things. A word processing package, as another example, could be used for "Administrative Management of Information," or even for "Interactive Instruction" in a typing class. A file management package would most probably have the purpose of "Management of Instruction" (to keep grades, test records, etc.). A program creating a word search would be "Creation of Materials." You get the idea, right?

The portion of the form titled "Subject Matter" needs little explanation. You may not want to check any of these, depending on the purpose of the software. For example, "Creation of Materials" could be for any subject matter. Likewise, "Administrative Management of Information" might not be related to any single subject matter. If the software is related to a particular subject matter, please note it.

To complete our description of the software (and to start reporting your judgments about the package), supply your best estimate for the appropriate classes to use the software. For a word processing package, you might want to circle 7 through adult, or even start with a lower age/class.

The other items on the form ask for your judgments about the software. Rate the software on the seven criteria listed. We are asking for your judgments; do not be afraid to be honest here. This is an evaluation of educational software, so we want you to "grade" the package.

Finally, identify your relationship to schools. Perspectives of different groups may vary, which is valuable information to potential buyers.

So much for the description and judgment of the national network. Now for the network part. The network, dear reader, means *you!* This idea will not work unless you complete and mail in those forms. I will collect the forms and provide an analysis after a substantial number come to me. (The analysis, naturally, will be conducted using my Color Computer.) The data will be entered and stored using a File System, and the analysis provided with a BASIC program.

Please mail your completed form to: Michael Plog, 829 Evergreen, Chatham, Ill., 62629. Unfortunately, I cannot provide a reward for completed forms, not even the stamps necessary to mail the information to me. We will have to rely on your interest and involvement as a concerned Color Computer user. I happen to think most of you readers are interested and involved.

One other note. I do not know how many times we will print the evaluation form in *the Rainbow*, so please make copies of the form. Feel free to make as many copies as you wish. I do not like to destroy my magazines (I save all my *Rainbows*, don't you?), so I highly encourage making copies. In fact, if you want to pass out the form to school people, feel free to make several copies.

By way of a final remark and quick update, Radio Shack has completed an agreement with the Minnesota Educational Computing Consortium (MECC) to convert selected programs from the MECC educational courseware library to the Model III and Color Computer. Some of you may know about MECC; they have a national reputation for leadership in educational software. We look forwared to the results of this agreement.

Until next month, think evaluation—think education!

Drawing On This Blackboard Chalks Up Programming Ease.

By Joseph Kolar
Rainbow Contributing Editor

"This is such a simple program that even the person not too familiar or enthusiastic about programming could use it to create his/her own displays."

The object here is to allow anybody, within the constraints of this program, to create their own programs. It appears to have some benefit when used as a blackboard for individualized instruction. There is no need to get into the arcane depths of computing to produce creditable results. It allows a person to create a display that they visualize in their mind's eye without the attention to programming conventions that are usually required. It is unencumbered with excess impedimenta and elegant touches. In a word, it is simple.

The program is for use on the hi-res screen of the 16K ECB Color Computer. All variables are double letters. The first letter is the code for the option and the second letter is the name of the letter or number called.

A Code—Capital letters and numerals of the script alphabet.
B Code—Small letters of script.
C Code—Alternate small script letters. (optional)
K Code—Upper case and numerals of the print alphabet.
N Code—Lower case of the print alphabet.
Varied Codes—Spacers and punctuation marks.

Both alphabets may be used on the screen at the same time. A maximum of 20 printed letters and 25 script letters fill an entire row, provided you use "S8" size option, in the *DRAW* statement. S8 is recommended; S12 is good and S4 may be used, but not recommended.

Nine rows will fit comfortably on the screen, providing you use S8, begin at a vertical coordinate of 20 and increment +20 for each row to provide adequate spacing. On the horizontal, if indenting, use 15 or 20, otherwise, 5 leaves plenty of space at the left margin.

The *DRAW* command is used in every program line. If size is not changed during the program, it need only be put into the initial *DRAW* statement. The first line would begin: *DRAW* "S8BM15,20."

To concatenate the name "Joe" in print, you would add to the above *DRAW* line: +KJ$+NO$+NE$. This means concatenate (+), capital printed letter, (K), name of letter, (J), concatenate, (+), small printed letter, (N), name of letter, (O), concatenate, (+), small printed letter, (N), name of letter, (E).

Due to the large overhead, only three screen pages of print/text are possible. The two sets of numerals are somewhat similar and one set could be deleted to conserve memory. If I remember correctly, some of the punctuation marks are duplicated. A few could be deleted. The spacer works well with any option.

This is such a simple program that even the person not too familiar or enthusiastic about programming could use it to create his/her own displays. Concatenation serves the bill very nicely. The program lines, beginning at line 2000 would consist entirely of *DRAW* statements, except for the last line on each display page. The variables are so coded that the programmer can easily follow and locate his position when looking at the listing. All she/he has to do is read the second letter of the variable. Reading the second letters will indicate exactly what is being written or printed to the screen. Thus, it is easy to locate and correct a mistake. In the interest of saving memory *CLEAR* 450 is good, so far, but *CLEAR* 500 appears to be good all the time.

When *RUN*, the program will display the first page. To proceed to the next page, press the G key. Do the same to get to the third page. Another press of the G key and it will recycle to the first page. It may be held at the end of the third page in an infinite loop.

Everything is printed or written via concatenation. Begin a new program line after two lines of concatenated variables are added to the *DRAW* statement. (Sixty-four spaces or two rows on the screen.) Much more than that will get an LS

"Everything is printed or written via concatenation. Begin a new program line after two lines of concatenated variables are added to the DRAW *statement."*

error message. It is better to be reasonable and not try to cram as many variables into a program line as possible. You will only have to backtrack and lop off a few variables and start a new program line anyway.

Reaching the end of a row, (right-hand margin) requires a new program line. The "follower" line must begin with *DRAW* and the next variable need not have + prefixed, but all the following concatenated variables must have the + in front of the variable.

There is a third case when you are likely to need an additional *DRAW* program line. When creating two columns of text on the same screen line, it is easier and neater to use a new *DRAW* statement with the coordinates to locate the first letter of the second column. If alignment is not necessary, you can insert +"BRN" where N is a number, to move over N spaces to the right. Try any number, N, check it out and adjust left or right by increasing or decreasing N. Then continue to concatenate.

Briefly, to use this program, delete lines 2000-2999. Locate and size the starting program line at 2000 inside the quote marks of a *DRAW* statement. Then concatenate about 10 variables, not forgetting any spacer variables. *RUN* it and see how it looks. Then, begin a new program line with *DRAW*. No quote marks! Continue to concatenate. If you concatenate about 10 more variables, you will be approaching the right-hand margin. Check it out to make sure you don't run off the screen. Reposition the second row using the same system as line 2000. Make sure you increment the vertical by +20 to insure adequate spacing between rows.

If you use 20 as an increment, you can easily locate a program line by counting by 20s and check the *DRAW* statement in the listing that has a vertical element the same as the one you just counted out. It is also a good idea to increment each second and possible third program line in the same row, by an odd number. This gives a good clue as the program's lines will indicate if it is a "starter," 2000, 2010, even number or a "follower" line, 2005, 2007, 2015, 2017, odd numbers.

At the end of the last *DRAW* program line on a screen display page, add *:GOSUB* 3020. This will hold the display until you are ready to proceed. Press the G key.

When you get to the very last line in your program (2000 block)you have the choice of holding the last screen display or returning to the starting page.

If you want to hold at the last screen display, at the end of your final line, add *:GOTO* 3010. If you want to return to the first page, add *:GOSUB* 3020. Then press the G key.

That's it! Pretty easy, huh?

After using this system a few times, it is almost a joy, and certainly satisfying to key in any desired information. There is so little to remember. Since the variables are coded, practically the entire system can be committed to memory. Feel free to redesignate the variables that govern the punctuation marks and spacers to suit yourself. For that matter, you can change the code letters, if you prefer some other codes.

Note that the C coded letters should be examined and if not desired, deleted from the program to save memory. The B coded letters begin at the baseline and the C coded letters begin at the top of the small letters. Key in *DRAW* "S8BM100,80"+CO$+CN$+BO$+BN$ and see the difference.

```
150____01D3
450____0556
790____0903
1240___0D2F
1610___10FF
2070___1446
2340___1825
END___1B9B
```

The listing:

```
10 CLEAR 450
20 PMODE4,1:PCLS:SCREEN 1,0
30 AA$="BRNR2HU3ER2FNUD3NGFBR"
40 AB$="BRNR2HNGU3NHER2FGNL3FDGB
R2"
50 AC$="BU5F2M+3,-1HL2G2D2FR3NEB
R2"
60 AD$="BU4BRD4LURFR2EU3HL3GBD4B
R3"
70 AE$="BRNR2HUENRHER2FNEBD3GBR2
"
80 AF$="BU2NF2NRBU2ER3NRGDNLNRD2
GBR3"
90 AG$="BU5FR4HL2G2D2FR3EUL2BD2B
R3"
100 AH$="EU2NR3UHBR5GD3FBR2"
110 AI$="BUURBD2NHE3UHGD2F2RGR"
120 AJ$="BUNEFREU4LNGR2BD5"
130 AK$="EU3HBR5G3LBURF2DRBR"
140 AL$="BU2R2E2ULGD3GLBR2R2EBRB
D"
150 AM$="BU5FND4EFND4EFD3FBR"
160 AN$="BU5FND4ERFD3FBR"
170 AO$="BRHU3ER2FNED3GNL2BR2"
180 AP$="EU2UNHNUER2FDGL2HBD3BR4
"
190 AQ$="BRNR2HU3ER2FD3GUHBD2BRD
FBU2BR"
200 AR$="EU3NHER2FDGL3EF3BR"
210 AS$="E4UHGDF3DL3H2BR6BD2"
220 AT$="BU5FER2NRGD3GLHURBD2BR4
"
230 AU$="BU4ED4FRENU4FBR"
240 AV$="BU5FD3FREU3EBRBD5"
250 AW$="BU5FD3FENU4FEU3EBD5"
260 AX$="BU5RF2DF2BU5G5BR6"
270 AY$="BU4ED2FRND2REU2BD5BR"
280 AZ$="BU5FER2FGLG2FREFBR"
290 BA$="RU2ERFDNFGLHBDBR4"
300 BB$="RE2U3HGD4RFREUNLRBD2"
310 BC$="RU2ERFBD2L2HBDBR3"
320 CC$="BRBU3FER2BL3BDDFREF"
330 BD$="RU2ERFDGLHBR3U4HGDFBDBF
F"
340 CD$="BU3FERBG2NUFREU4HGDF2DF
"
350 BE$="RE2HGF2R"
360 CE$="BU3F2R2EHLGF2R"
370 BF$="RE3UHGD6FEUHNLEBD"
380 BG$="ENFUERFDGLHBR3D3GHE3BD"
390 CG$="BU3R2NRGDFREUNHD4GHE3BD
"
400 BH$="RE3UHGD3NDBEFDR"
410 CH$="BRBU3FBEUHGD2ND3EFD3R"
420 BI$="REU2BUNBD3DRBR"
430 CI$="BU3FREBUNEBDD3RBR"
440 BJ$="RE2UBUEBGBD2D4GHE3BD"
450 BK$="RE3UHGD4BUBER2DL2NUFR2B
R"
460 BL$="BRRE2U3HGD4FR"
470 CL$="BU3F2EU3HGD4FR2"
480 BM$="RENDUNUEFND2EFD2R"
490 CM$="BU3FND2EFND2EFD2R"
500 BN$="RENDUNUEFD2R"
510 CN$="BU3FND2NUERD3R"
520 BO$="BEUERFDGLHBDBR4"
```

```
530 CO$="BEUERNR3FDGLHBDBR4"
540 BP$="EUERFDGLHD4BU3BR3"
550 CP$="BRBU3FND4ERFDGLHBDBR2RB
R"
560 BQ$="BEUERFDGLHBDBR2D2FEUHNL
RBDBU"
570 BR$="REU2FRD2R"
580 CR$="BU2EFRD2R"
590 BS$="REU2F2GNHR2"
600 BT$="REU3NL2NR2NUBD3FR"
610 CT$="REU2NL2NR2NU2BD2FR"
620 BU$="RENU2FENU2F"
630 CU$="BENU2NDFENU2NDF"
640 BV$="RENU2FEU2RBD3"
650 CV$="BU3RD2FEU2RBD3"
660 BW$="RENU2FENU2FEU2RBD3"
670 CW$="BRBUNU2FENU2FEU2RBD3"
680 BX$="RE3BL3F3"
690 BY$="RENU2FENU2D3GHE3BD"
700 BZ$="BUBRUERFDGLBRD2GHE2R"
710 OM$="BR2BUDGBEBR2"'COMMA
720 QU$="BU3BRUERFDGBDDBR2"'? MA
RK
730 DT$="BRRBR2"'PERIOD
740 OL$="BU4BR2DBDDBR2BD"'COLON
750 SS$="BR5"'SPACE
760 XC$="BU5BR2D2BD2DBR2"'!
770 HP$="BE2R2BRBD2"'HYPHEN
780 UP$="BU5BR2D2BDBF2"'APOSTROP
HE
790 QT$="BU5BR2D2BRU2BD5BR2"'QUO
TE
800 A0$="BRHU3ER2FD3GNL2BR3"
810 A1$="BU4BR2ED5BR2"
820 A2$="BU4ER2FDGL3D2R4BR2"
830 A3$="BU4ER2FGNL2FDGL2NHBR6"
840 A4$="BR3U5G3R4BD2BR3"
850 A5$="BU5NR4D2R3FDGNL3BR4"
860 A6$="BU5BRNR3GD3FR2EUNL2GBD2
BR7"
870 A7$="BU5R4D2G3BR6"
880 A8$="BRNR2HUEHER2FGNL2FDGBR4
"
890 A9$="R3EU3HL2GDFR2EBF3"
1000 NA$="BU4R3FDHL2GDFR2ENU2FBR
2"
1010 NB$="BU6RD6NLBUFR2EU2HL2GBF
3BR3"
1020 NC$="BRNR2HU2ER2FBD2GBR3"
1030 ND$="BRNR2HU2ER2FD2NGDU6NLB
D6NLBR3"
1040 NE$="BRHU2ER2FDL3BD2R2NEBR3
"
1050 NF$="BRU3NLNRU2ERFDBD4BR2"
1060 NG$="BRNR2HU2ER2FNUD2NGD2GL
2HBEBR5"
1070 NH$="NRBU6RD3ND3ER2FD3BR2"
1080 NI$="RNRU4LBURBD5BR3"
1090 NJ$="BDFREU5LBURBD5BR2"
1100 NK$="RU6NLBD4R2N[illegible]F2BR2"
```

```
1110 NL$="BU6RD6NLRBR2"
1120 NM$="BU4FND3EFND3EFD3BR2"
1130 NN$="BU4FND3ERFD3BR2"
1140 NO$="BRHU2ER2FD2BNL2BR3"
1150 NP$="BU4FND2ER2FD2BL2HD3LBU
2BR7"
1160 NQ$="BRNR2HU2ER2FNED2NBD3RB
U2BR2"
1170 NR$="BRBU4FND3ERFBD3BR2"
1180 NS$="BUFR2EHL2HER2FBD3BR2"
1190 NT$="BU4RNUNRD3FREBDBR3"
1200 NU$="BUNU3FR2ENU3FBR2"
1210 NV$="BU2NU2F2E2NU2BD2BR2"
1220 NW$="BUNU3FENU3FENU3BR2BD"
1230 NX$="E2NH2NE2F2BR2"
1240 NY$="BUNU3FR2ENU3D2BL2HBUBR
6"
1250 NZ$="BU4R4G4R4BR2"
1260 SS$="BR5"
1270 XM$="NUNGBR2"
1280 DT$="BR2R"
1290 XC$="UBUU3BD4BR2"
1300 QM$="BU5ER2FDBLDBDDBR4"
1310 KA$="U5ER2FD3NL4D2BR3"
1320 KB$="U6R3FDBNL3FDBNL3BR5"
1330 KC$="NRHU4ER2FBD4BNL2BR4"
1340 KD$="U6R3FD4BNL3BR4"
1350 KE$="NR4U3NR3U3R4BD6BR3"
1360 KF$="U3NR3U3R4BD6BR3"
1370 KG$="BRHU4ER2FBD3NLDBNL2BR4
"
1380 KH$="U3NU3R4NU3D3BR3"
1390 KI$="NU6BR3"
1400 KJ$="BUNUFR2ENU5BDBR3"
1410 KK$="U3NU3R2E2UBD3BL2F2DBR3
"
1420 KL$="NU6R4BR3"
1430 KM$="U6F3E3D6BR3"
1440 KN$="U3NUF4NU5DBR3"
1450 KO$="BRNR2HU4ER2FD4BBR4"
1460 KP$="U6R3FDBNL3BF3BR"
1470 KQ$="BRHU4ER2FD4BLNUNDNLBR5
"
1480 KR$="U6R3FDBL2NLF3BR3"
1490 KS$="BUFR2EUHL2HUER2FBD5BR3
"
1500 KT$="BR2U6NL2R2BD6BR3"
1510 KU$="BRHU5BR4D5BNL2BR5"
1520 KV$="BR2H2U4BR4D4B2BR5"
1530 KW$="BUNU5FE2F2ENU5BR3BD"
1540 KX$="UE2H2UBR4DB2F2DBR3"
1550 KY$="BR2U3H2UBR4DB2BF3BR2"
1560 KZ$="NR4UE4UNL4BF3BD3"
1570 K1$="BRU6NBBD6BR3"
1580 K2$="BU6R3FD2BL3D2R4BR3"
1590 K3$="BU6R3FDBNL2FDBNL3BR4"
1600 K4$="BU6D4R3NU4NRD2BR4"
1610 K5$="R3EU2HL3U2R4BD6BR3"
1620 K6$="BUNUFR2EUHL2BU3ER2FBD5
BR3"
1630 K7$="BU6R4D3BD2BR4"
1640 K8$="BRHUEHUER2FDBNL2FDBNL2
BR4"
1650 K9$="BUFR2EU4HL2BDFR2EBD4BR
3"
1660 K@$=KO$
1670 HF$="BU2R2BD2BR2"
1990 '
2000 DRAW "S8BM20,20"+AW$+BH$+BA
$+CT$+SS$+BI$+BS$+SS$+BY$+CO$+CU
$+BR$
2005 DRAW SS$+BN$+BA$+BM$+BE$+QU
$
2010 DRAW "SBBM15,40"+KW$+NH$+NA
$+NT$+SS$+NI$+NS$+S8$+NY$+NO$+NU
$+NR$+S8$
2015 DRAW NN$+NA$+NM$+NE$+QM$
2020 DRAW "BM20,60"+AM$+BY$+SS$+
BN$+BA$+BM$+BE$+SS$+BI$+BS$+SS$
2025 DRAW AC$+CH$+BE$+BR$+BY$+CL
$+SS$+BA$+BN$+BD$+SS$
2030 DRAW "BM5,80"+BH$+BI$+BS$+S
S$+BN$+BA$+BM$+BE$+SS$
2035 DRAW BI$+BS$+SS$+AS$+BT$+BE
$+BV$+BE$+DT$
2040 DRAW "BM15,100"+KM$+NY$+SS$
+NN$+NA$+NM$+NE$+SS$+NI$+NS$+SS$
2045 DRAW KC$+NH$+NE$+NR$+NY$+NL
$
2050 DRAW "BM5,120"+NA$+NN$+ND$+
```

```
SS$+NH$+NI$+NS$+SS$+NN$+NA$+NM$+
NE$+SS$
2055 DRAW NI$+NS$+SS$
2060 DRAW "BM5,140"+K8$+NT$+NE$+
NV$+NE$+DT$
2070 DRAW "BM5,160"+BC$+BA$+BT$+
SS$+SS$+NC$+NA$+NT$+SS$+SS$
2075 DRAW BD$+BO$+BG$+SS$+SS$+ND
$+NO$+NG$
2080 DRAW "BM5,180"+A2$+DT$+SS$+
AD$+BA$+BD$+SS$+KD$+NA$+ND$+SS$
2085 DRAW K2$+DT$+SS$+AM$+CO$+CM
$+SS$+KM$+NO$+NM$:GOSUB 3020
2170 DRAW "BM5,20"+A3$+SS$+AP$+B
A$+BU$+CL$
2180 DRAW "BM 120,20"+K3$+SS$+KP
$+NA$+NU$+NL$
2190 DRAW"BM5,40"+A4$+SS$+AA$+BR
$+BT$+BH$+BU$+BR$
2200 DRAW"BM120,40"+K4$+SS$+KA$+
NR$+NT$+NH$+NU$+NR$
2210 DRAW "BM5,60"+A5$+SS$+AF$+B
R$+BA$+BN$+BK$
2220 DRAW "BM120,60"+K5$+SS$+KF$
+NR$+NA$+NN$+NK$
2230 DRAW "BM5,80"+A6$+SS$+AN$+B
A$+BN$+BC$+BY$
2240 DRAW "BM120,80"+K6$+SS$+KN$
+NA$+NN$+NC$+NY$
```



```
2250 DRAW "BM5,100"+A7$+SS$+AS$+
BU$+BS$+BA$+BN$
2260 DRAW "BM120,100"+K7$+SS$+K8
$+NU$+NS$+NA$+NN$
2270 DRAW "BM5,120"+A8$+SS$+AH$+
BE$+BN$+BR$+BY$
2280 DRAW "BM120,120"+K8$+SS$+KH
$+NE$+NN$+NR$+NY$
2290 DRAW "BM5,140"+A9$+SS$+AB$+
BE$+BT$+BT$+BY$
2300 DRAW "BM120,140"+K9$+SS$+KB
$+NE$+NT$+NT$+NY$
2310 DRAW "BM0,160"+A1$+A0$+SS$+
AG$+BE$+BO$+CR$+BG$+BE$
2320 DRAW "BM112,160"+K1$+K0$+SS
$+KG$+NE$+NO$+NR$+NG$+NE$
2330 DRAW "BM0,180"+A1$+A1$+SS$+
"BR2"+AD$+BA$+BV$+BI$+BD$
2340 DRAW "BM120,180"+K1$+K1$+SS
$+KD$+NA$+NV$+NI$+ND$:GOSUB 3020
2350 DRAW "BM15,20"+AT$+BH$+BE$+
SS$+BQ$+SU$+BI$+BC$+BK$+SS$
2355 DRAW BB$+BR$+CO$+CW$+CN$+SS
$+BF$+CO$+BX$+SS$
2370 DRAW "BM5,40"+BJ$+BU$+BM$+B
P$+BS$+SS$+CO$+CV$+BE$+BR$+SS$
2375 DRAW CT$+CH$+BE$+SS$+BL$+BA
$+BZ$+BY$+SS$+BD$+CO$+CG$+DT$
2390 DRAW "BM15,60"+KT$+NH$+NE$+
SS$+NQ$+NU$+NI$+NC$+NK$+SS$+NB$+
NR$
2395 DRAW NO$+NW$+NN$+SS$+NF$+NO
$+NX$
2410 DRAW "BM5,80"+NJ$+NU$+NM$+N
P$+NS$+SS$+NO$+NV$+NE$+NR$+SS$
2415 DRAW NT$+NH$+NE$+SS$+NL$+NA
$+NZ$+NY$
2430 DRAW "BM5,100"+ND$+NO$+NS$+
DT$
2440 DRAW "S4BM15,120"+KT$+NH$+N
E$+SS$+NQ$+NU$+NI$+NC$+NK$+SS$
2445 DRAW "R20"+SS$+NF$+NO$+NX$+
SS$+NJ$+NU$+NM$+NP$+NS$
2447 DRAW SS$+NO$+NV$+NE$+NR$+SS
$+NT$+NH$+NE$+SS$+"R16"
2470 DRAW"BM5,130"+ SS$+ND$+NO$+
NS$+DT$
2480 DRAW "BM25,150"+K1$+DT$+SS$
+NR$+NE$+ND$+SS$+K2$+DT$+SS$
2485 DRAW NL$+NA$+NZ$+NY$+SS$+K3
$+DT$+SS$+NB$+NR$+NO$+NW$+NN$
2487 DRAW SS$+K4$+DT$+SS$+NS$+NM
$+NA$+NL$+NL$:GOTO 3010
3000 GOTO 1990
3010 GOTO 3010
3020 A$=INKEY$
3030 IF A$="B" THEN GOTO 3040 EL
SE GOTO 3020
3040 PCLS:RETURN
```

Peeking Into The Hidden Commands Of The MC-10

By Tom Szlucha

At first glance, the MicroColor Computer (MIC) recently introduced by Radio Shack appears to be well-suited for beginning programmers, but may turn away many advanced programmers because it lacks several commands needed to manipulate machine code. After some detailed examination into the BASIC interpreter, I have found that this is not entirely true. Besides the 68 documented commands and functions, there are at least five undocumented commands that, with some study, should allow an unleashing of the machine language capabilities of this Micro-box.

Tandy has built several undocumented commands into MIC apparently for use with cassette-based software. These commands support various machine language functions which should allow them to market more sophisticated software than is allowed with BASIC. The quest for these "hidden" commands came from knowing that Radio Shack has a reputation for not always disclosing everything that they know (or should know) about the operation of their machines. It was also driven by a desire to understand some of the differences between the BASIC used in MIC and that used in the CoCo.

There are several techniques which can be used to establish the existence of hidden commands. One of these techniques is somewhat hit and miss but can be effective. Suspected commands are typed into the computer and the subsequent response is studied. For example, examine Table 1 which contains MIC's response to several commands.

Table 1 — Response to Various Keyboard Input

Command	Response	Comment
NEW	OK prompt	known command
LIST	OK prompt	known command
XYZ	syntax error	known garbage
EXEC	FC error	suspected command
CLOADM	S (searching tape)	suspected command
CLOADZ	TM error	verification

Typing *NEW* and *LIST* give the expected OK prompt, and XYZ is clearly recognized as a non-command resulting in a syntax error. *EXEC* gives an FC error rather than syntax error indicating that MIC recognizes the existence of the command but needs more information to process it. The response to *CLOADM* is a familiar S in the upper lefthand corner of the screen as BASIC waits for a program to appear at the cassette port. If *CLOAD* were typed instead of *CLOADM*, the response would be the same, but note the response to *CLOADZ*, a TM error occurs. This indicates BASIC treats the "M" with special respect. In fact, BASIC only likes to see an M, space or quote after *CLOAD*.

With several hundred commands used in the various dialects of TRS-80 language, not including countless possible new variations, the trial and error method of searching is obviously limited. Another technique which can be used to discover "new" commands is to examine the BASIC Command Table. The Command Table is a list of key words used by the computer to distinguish commands and functions from variable names. MIC's BASIC directory is located in memory location 57413 to 57670 (decimal). The short program (Listing 1) shown below can be used to "peek" at this list of commands.

Listing 1 — Peeking At MIC's Command Table

```
10 FOR A=57413 TO 57670
20 P=PEEK(A)
30 IF P>127 THEN LPRINT CHR$(P-128),:LPRINT
   A:GOTO 50
40 LPRINT CHR$(P);
50 NEXT A
```

This program prints a list of the commands and functions recognized by BASIC along with the address of the last character in each command. These commands are stored in memory using the ASCII code for each letter with 128 added to the last character of each command to indicate to the computer that it is the end of a word. Undocumented commands: *EXEC, CLOADM, USR, VARPTR,* and *OFF*. Of these, three have been determined to work, the other two will have to await further detective work.

The hidden commands work essentially like their CoCo counterparts. The following is a brief description of each.

CLOADM "filename",S,E,T — Loads a machine language program named "filename." S is the address to start loading the code, E the ending address, and T, the transfer address; all of these are in decimal.

EXEC X Executes a machine language program. If the program was previously loaded with *CLOADM*, X, the transfer address is handled automatically

VARPTR This is a function used to determine the address where a specific variable is stored. It also can be used to determine the size of a string variable. This is a very powerful command, allowing string packing for fast animation and allows

fast sorting routines—a real gem to have on MIC. See Ron Mummaw's recent article "VARPTR—BASIC's Hidden Command" in the June 1983 issue of *the Rainbow*.

The fourth command, *USR*, more than likely works in Color BASIC, but the address which holds the location of the user-supplied subroutine has eluded detection. Until this location can be found, *EXEC* can be used in many instances as a substitute. The fifth command, *OFF*, is in the key word table and is recognized by the computer, but there is nothing to turn off. In Color BASIC, *OFF* works with *MOTOR* and *AUDIO*, but these are not present as functions on the MC-10. It could be that this command is either a leftover or is present for a future enhancement. A call to Tandy's technical hotline was of little use in obtaining more detail on these commands. It appeared that they were learning about these hidden commands for the first time from the phone inquiry. Their response as expected was that the commands were for "future applications" and they were not aware of any further intended documentation.

It is certainly exciting to discover this hidden power in the MC-10, but somewhat disappointing that a *CSAVEM* command was not also discovered. The lack of a machine language save feature appears to be the only major stumbling block to writing machine code on this computer. To test the *EXEC* and *CLOADM* commands, I resorted to writing a short test program on the CoCo, saving it to tape, then inputting it to MIC. Once a commented disassembly of the BASIC ROM is available, a save routine can be developed. Also, it may be possible to trick the machine into saving machine code by changing the BASIC pointers. Looks like more detective work ahead.

By the way, BASIC code in MC-10 is stored almost identical to the method used in the Color Computer. Bytes 1 and 2 of a command line point to the next command line. Bytes 3 and 4 contain the line number. The next bytes (up to 124 more) contain the coded commands, variables, constants, etc. The commands are tokenized, the rest in ASCII code. As pointed out by Dan Downard in "Technical Review of the MC-10" in the July 1983 *Rainbow*, the tokens are not the same as those used in Color BASIC. In a future article, I plan to show a software routine to convert programs written in Color BASIC on the CoCo to load and run on the MC-10.

In a related topic, I have noticed that the MC-10 seems to run BASIC *faster* than the Color Computer. To verify this, I ran the simple Benchmark program shown in Listing 2 to compare them.

Listing 2 — Timing Program

```
10 FOR I=1 TO 1000
20 PRINT I
30 NEXT I
```

Sure enough, the MC-10 took 17.5 seconds, whereas the CoCo took 20.7 seconds. It takes the speed-up *POKE* for the CoCo to beat its little brother (13.9 seconds). Since the BASIC clock speed is the same for the two computers at .895 mhz, I attribute the faster execution speed to more efficient interpreter code in the MC-10.

(I would like to acknowledge the technical contributions of my 15-year-old son, David Szlucha and his 16-year-old collaborator, Dean Swain. These microteens made many of the discoveries discussed in this article within hours of unpacking our new MC-10.)

MC-10 vs. CoCo
A Command Summary

By Frank Smith

While poking around inside my PoCo I made some interesting discoveries. One thing I noticed was that Microsoft managed to pack plenty of power in that 8K ROM. I kept running into so many commands which belonged to Extended BASIC on the CoCo that I decided to make a comparison chart between both levels of BASIC on the CoCo and Micro Color BASIC. The Extended BASIC commands are marked with a #. The only real disappointments are the lack of *EDIT* and *ELSE*.

Radio Shack has made provision for an easy upgrade to 20K by using a socket which can hold a 16K RAM module (Cat. No. 26-3013, $49.95).

As mentioned by Thomas Szlucha in articles last month and this month, there are some BASIC words which are not mentioned in the documentation that comes with the PoCo. Could there be more?

CoCo Functions	MC-10 Functions
ABS(numeric)	ABS(x)
ASC(str)	ASC(x$)
# ATN(numeric)	****
CHR$(code)	CHR$(x)
# COS(numeric)	COS(numeric)
EOF(f)	****
# EXP(numeric)	EXP(numeric)
# FIX(numeric)	****
# HEX$	****
INKEY$	INKEY$
INT(numeric)	INT(x)
JOYSTK(j)	****
LEFT$(str,length)	LEFT$(str,c)
LEN(str)	LEN(str)
LOG(numeric)	LOG(numeric)
MEM	MEM
MID$(str,pos,length)	MID$(string,pos,len)
PEEK(location)	PEEK(n)
POINT(x,y)	POINT(x,y)
# POS(device)	****

CoCo Functions	MC-10 Functions
# PPOINT(x,y)	****
RIGHT$(str,c)	RIGHT$(str,length)
RND(n)	RND(n)
SGN(x)	SGN(x)
SIN(numeric)	SIN(numeric)
# STRING$(length,code)	****
STR$(numeric)	STR$(x)
# SQR(numeric)	SQR(numeric)
# TAN(numeric)	TAN(numeric)
# TIMER	****
# USR(x)	>>USR(x)<<
VAL(str)	VAL(str)
VARPTR(var)	>>VARPTR(var)<<

CoCo Statements	MC-10 Statements
AUDIOON / AUDIOOFF	****
# CIRCLE(x,y),r,c,hw,s,e	****
CLEAR n,h	CLEAR n,h / CLEAR n,h
CLOAD	CLOAD
CLOADM	****
****	CLOAD*
CLOSE d	****
CLS c	CLS(c)
# COLOR(foregnd,backgnd)	****
CONT	CONT
CSAVE	CSAVE
****	CSAVE*
DATA	DATA
# DEF FN	****
# DEFUSR n	****
DEL	****
DIM	DIM
# DLOAD	****
# DRAW	****
# EDIT	****
END	END

CoCo Statements	MC-10 Statements
EXEC a	>>>EXEC a<<<
FOR...TO...STEP/NEXT	FOR...TO...STEP/NEXT
# GET(start)-(end),des,G	****
GOSUB	GOSUB
GOTO	GOTO
IFtestTHEN...ELSE...	IF...THEN ****
INPUT	INPUT
INPUT#-1	****
# INSTR (pos,search,target)	****
# LET	LET
LIST	LIST
LLIST	LLIST
****	LPRINT
# LINE(x1,y1)-(x2,y2),BF	****
# LINE INPUT	****
MOTORON / MOTOROFF	****
NEW	NEW
ON...GOSUB	ON...GOSUB
ON...GOTO	ON...GOTO
OPEN m,#d,f	****
# PAINT (x,y),c,b	****
# PCLEAR n	****
# PCLS	****
# PCOPY	****
# PLAY	****

CoCo Statements	MC-10 Statements
# PMODE mode,start-page	****
POKE (location,value)	POKE
# PRESET	****
PRINT	PRINT
PRINT@n	PRINT@n
PRINT#-1	****
PRINT#-2	****
PRINTTAB	PRINTTAB
# PRINT USING	****
# PSET (x,y,c)	****
# PUT (start)-(end),scrc,act	****
READ	READ
REM	REM
# RENUM	****
RESET (x,y)	RESET(x,y)
RESTORE	RESTORE
RETURN	RETURN
RUN	RETURN
# SCREEN screen-type,clrset	****
SET (x,y,c)	SET(x,y,c)
SKIPF	SKIPF
SOUND tone, duration	SOUND(t,d)
STOP	STOP
# TROFF	****
# TRON	****

Lower Case Interpreter Could Be Time Saver

Wish you were able to type program listings in lowercase mode instead of having to toggle back and forth between lower- and uppercase with the CoCo's cumbersome Shift-0? Wish no more. The folks at JARB Software have come up with a nice, short machine language program that allows you to type in lowercase mode not only the messages within quotes that you want in lowercase, but also the BASIC commands that otherwise require an uppercase input.

LCINT, which stands for *Lower Case Interpreter*, is a short machine language on tape that loads in less time than what it takes to insert the cassette in the tape recorder. Two versions of the program are included on each side of the tape, *LCINT16* and *LCINT32*, for a 16K . . . Oh, but you've guessed what the two versions are for. Once you've loaded the appropriate version on your machine, you type in *EXEC*. Nothing noticeable happens just yet. But press Shift-0 and type the following line:

10 print "Show us your lower case!"

Now *LIST* it. The line now appears as

10 PRINT "Show us your lower case!"

and when you *RUN* the program your message appears just as you typed it, with reverse video characters indicating lowercase, unless, of course, you have installed one of the lowercase boards on the market. And, if you have a printer, true upper- and lowercase would be displayed. The important thing, however, is that the lowercase BASIC command *print* was accepted as typed, permitting easier input.

LINCT also provides a single key PAUSE function, implemented with the CLEAR key, which would not only perform the same functions as SHIFT@, but gives you the convenience of single step through program listings or through disk directories, leaving the normal functions of the CLEAR key untouched when not listing a program or a directory.

If you have a need for printing messages in upper- and lowercase in your BASIC programs, this machine language utility can be a timesaver. The program performs as advertised. The 16K version resides in memory locations &H3F60 through &H3FFA, and the 32K version resides between memory locations &H7F60 and &H7FDF. The program comes on cassette tape, but is disk compatible. Although the documentation consists of a single sheet, it proves adequate for this simple application program.

(JARB Software, 1636 D Avenue, Suite C, National City, CA 92050, $10.95)

—Dr. Carlos Calle

Save Time With *Tape Utility*

Tape Utility from Spectrum Projects is actually a set of nine operations to make living with a Color Computer a lot easier or, at least, a lot less complicated. To begin with, *Tape Utility* can be used with either a tape or disk system, but using with a tape-only system will yield only four of the nine operations available on the program.

The program is written in machine language and comes with a four-page documentation which was found to be fairly complete. After a *CLOADM* and *EXEC* command a menu appears for the user to select one of the many options. While the program is running, no BASIC or Disk BASIC commands are available. A command is present to return to BASIC and typing *EXEC* will start up the program again.

I would categorize this utility as one of the "make things easier" variety—that is, a program that will, for the most part, do things that are possible to do without the utility, but take a lot more time to do. With this in mind, I jumped in, as many users would, and used the program without a complete review of the documentation. The results yielded only a few surprises and, upon closer scrutiny of the instructions, I discovered most of the surprises are covered in the documentation.

What does this program do to make life easier? To begin with, tape users can print a directory, either on screen or on a line printer of a given tape. The directory will include the name of the file, type (BASIC, machine language or Data), file mode, start, end and execute addresses of each program.

Surprise number one came when I tried to terminate the reading of data before the tape was completed. *Tape Utility* will only terminate while reading data and not during a gap or blank section of the tape. Yes, this is all covered in the documentation, but it does create some problems if you read an entire cassette and want to get back to the *Tape Utility* menu. You must either restart the program or read some dummy cassette data to enable the BREAK key stop option.

Tape users may also easily make tape-to-tape copies of BASIC, machine language or Data programs. *Tape Utility* will not copy some auto-start and most copy-protected disks and tapes.

Disk users are given six more options on *Tape Utility*. The program will easily copy a file from disk to tape, or tape to disk and do an automatic copy of an entire disk of programs to tape. Those of you who have used the *CSAVE* and *SAVEM* command will understand how much easier it is to use *Tape Utility* than continuous *PEEK*ing for start, end, and execute addresses of machine language programs. *Tape Utility* also includes some disk-only functions such as displaying a two-column compressed directory on screen or line printer and setting default drive number other than drive #0. Unfortunately, the disk directory given does not include start, end and execute addresses as the tape directory does. *Tape Utility* also does not include a simplified disk-to-disk copy routine. The user must return to Disk BASIC to use the *COPY* command if this needs to be done. Although *COPY* is easy to use it would be nice to have something like it available while *Tape Utility* is up and running. *Tape Utility* also does not "fix" tape programs written in the disk operating area. Attempts to use such programs may result in a crashed disk and a very frustrated user. This surprise (number two!) is *not* covered in the documentation.

Tape Utility will copy programs up to 10K in length in a 16K machine and 26K in a 32K machine. Users are cautioned in the documentation that *Tape Utility* will not copy segmented files produced by some assemblers. *Tape Utility* will recover from errors that are encountered during a read of a tape file, however. This unique ability is especially valuable if you are doing long tape-to-tape or tape-to-disk copies.

The decision on whether to buy *Tape Utility* or not is really a matter of the personal preferences of the user. The program is straightforward and performs as documented. It is a matter of time saving and convenience. Decide whether you want your CoCo to do the majority of the busy work involved in copying and cataloging tapes or if you have a few extra hours to do it yourself the hard way. I prefer to have the computer do the busy work and consider this program well worth the price.

(Spectrum Projects, 93-15 86th Drive, Woodhaven, NY 11421, disk or cassette $24.95 plus $3 s/h)

—Brian James

Humbug For the MC-10

Shortly after reviewing the new MC-10 "PoCo CoCo," I received a telephone call from Pete Stark of Star-Kits. Yes, in the same issue in which I stated that machine language was apparently not supported by the MC-10, an ad appeared for *Humbug*. Anyone familiar with *Humbug* knows it's one of the nicest monitor programs for the CoCo. Thanks to Pete, we now know that there is an *EXEC* and even a *CLOADM* command that are undocumented.

Taking advantage of the hidden commands, *Humbug* appears to be the first piece of commercial software for the PoCo. As I am machine language oriented anyway, I cannot think of a more useful tool for exploring the new computer. There is only one minus, the availability of memory. With only 4K of on-board memory, *Humbug* occupies a little less than 3K. After housekeeping and screen memory only around 400 bytes remain for user programs. Radio Shack is already advertising a 16/20K upgrade module, so, by the time you read this, the memory limitation should not be a problem. To combat this problem three different versions are available at different memory locations as the 6803 Processor is not capable of position independent code.

Before covering the different commands available, let me say something about the manual. The 25 pages of excellent instructions are not only complete with source code, but also contain a wealth of information on the MC-10, a memory map along with several addresses of key ROM calls.

The manual can be considered a standard of comparison and is alone worth the cost of the software. For example, one of the addresses points out the fact that Radio Shack didn't use the on-board serial interface for the RS-232 port. Addresses are given with proper values to control the RS-232 Baud rate from 110 to 2400 Baud. There is even a command to change the Baud rate.

Commands

Following is a summary of the commands implemented by *Humbug*. A detailed description of the operation of each command is given in the operation manual. A few are worth noting. The AT or analyze tape command will document the contents of an unknown tape, giving you the program name, format, number of bytes and the start, end and execution addresses. The FI command will allow you to search any section of memory for a hex string. The SA commands allow you to make machine language tapes. Full breakpoint control and single-stepping of programs is supported.

AD — Formatted ASCII Dump
AI — ASCII Input
AO — ASCII Output
AT — Analyze Tape
BA — Change Baud Rate
BP — Print Breakpoints
BR — Set/Reset Breakpoints
CO — Continue
CS — Checksum Memory
DE — Desemble
EX — Exit to BASIC
FI — Find 1, 2 or 3 bytes
FM — Fill memory with constant
HD — Hex dump
HE — Print command codes
JU — Jump to program
MC — Compare memory
ME — Memory examine and change
MM — Move memory
MT — Memory test
RC — Register change
RE — Register examine
SA — Save ML to cassette!
SS — Single-step
ST — Start single-step
!! — Monitor reset

I/O Control

Humbug has a unique feature using the SHIFT-@ combination that is used in BASIC to pause operation. When a SHIFT-@ is encountered one of four commands can follow. A "P" will toggle the pause mode which pauses screen output after every 16 lines for examination. The letter "O" turns the RS-232 port on and off for printer output. Hitting BREAK will cancel any program and return to the monitor prompt. Any other keys will resume the program.

Summary

As *Humbug* has been around on many other 68XX systems for quite a while, you will have no problems with operation. It is time- and user-tested and this reviewer played with the commands for quite a while and found no problems. You will find commands not available on most computer monitors in *Humbug*. I recommend it to anyone who wants to run machine language programs on their new MC-10.

(Star-Kits, P.O. Box 209-R, Mt. Kisco, NY 10549, $29.95 on tape)

—Dan Downard

De-Mystifying *PCLEAR 0*

—Ron Krebs

Several articles and hints have been published outlining the techniques and desirability of performing *PCLEAR 0* on the CoCo to provide maximum available memory. There is some confusion however, regarding the possibility and/or proper method of doing a *PCLEAR 0* when using the disk system. Please review your Extended BASIC manual if you are not completely familiar with the *PCLEAR* statement.

Let's review for a moment the method of doing a *PCLEAR 0* without the disk system:

Type: P●KE&H19,6:NEW (ENTER)

The *POKE* instruction causes a value of 6 to be poked into location $19. Memory locations $19 and $1A (Decimal 25 and 26), combine to form a 16 bit pointer register which tells BASIC where the main source buffer is located. Memory location $1A is '0' so the *P●KE* operation sets the two byte pointer to $0600. After the pointer is initialized, the *NEW* statement causes other BASIC registers to adjust accordingly.

But here is the important point:

When power is first turned on, BASIC causes the memory cell at location $0600 to be cleared because it requires the first byte of the source buffer to be cleared for proper operation. Now, let's observe what happens to memory when power is first applied to the CoCo. If you have a means of sequentially *PEEK*ing through memory you will generally find that the dynamic RAM used in your CoCo will have alternating bytes set to 0 and $FF. Depending on the memory chips used, your CoCo may have all the even bytes set to zero or it may be the odd ones.

Next, let's look at a possible means of performing a *PCLEAR 0* with a disk system. After reserving some RAM for the D●S, we decide to locate the source buffer at $0E00:

Type: P●KE&H19,&H●E:NEW (ENTER)

This should do it, but we find that it works with some computers and doesn't with others! Why? Because we don't know if the byte at location $0E00 is cleared or not! ●n some computers it will be and on others it won't. We have now identified the problem and the solution is easy. To *PCLEAR 0* with a disk system installed:

Type: P●KE&H19, &H0E:POKE&H0E00, 0:NEW

The second *POKE* statement does the trick! BASIC has the required zer● byte at the beginning of the buffer and will be perfectly happy. Your CoCo has more memory to work with and I hope that makes you happy too!

(Ron Krebs has many years of 6800/6809 programming experience along with a background in technical sales and management. Ron and his wife Mona own Mark Data Products.)

The Dragon Is Coming!

By Bob Rosen

Just what is this Dragon 32 that you see on the upper left hand corner of the cover of *Rainbow* Magazine? The Dragon 32 is England's answer to the Color Computer. (And a most impressive answer it is!) To start with, the first thing you notice when you see it, is that it has an improved keyboard. Upon closer inspection, it is similar to the original Model I keyboard manufactured by Hi-Tek. It is a standard 53 key Qwerty keyboard with a very niee smooth feel to it. Upon powering up, you get the following logo:

(C) 1982 DRAG●N DATA LTD,16K BASIC INTERPRE TER 1.0,
(C) 1982 by MICR●S●FT

As you can guess, it is the same Microsoft who wrote the BASIC for the Color Computer. To my surprise when comparing the Quick Reference Card of the Dragon32 to that of the CoCo, they have the exaet error eodes and statements except for *DLOAD*. The reason *DLOAD* is missing is that the Dragon 32 has no serial port. Instead it has a parallel port. ●ne interesting command I noticed on the Dragon Quick Reference Card is the presence of a *PRINT#-2* command instead of the usual *LPRINT* commands associated with parallel ports. This eliminates changing all the *PRINT #-2* commands of a Color Computer program to *LPRINT* commands. Even though the commands are the same, the actual BASIC is different from the Color Computer. This is because TANDY has exclusive rights to their version of Microsoft BASIC for the Color computer. While not all CoCo BASIC programs will run as is on the Dragon 32, I tried several RS Program Paks and they all worked correctly. I even tried a long machine language program *(Donkey King)* and had no problems. Included with my Dragon 32 was a pamphlet of programs for it. This list has programs from Spectral Associates and Mark Data that have been previously sold for the CoCo. This leads to the assumption that even though CoCo programs may not run "as is" on the Dragon, they can be converted to do so.

The Dragon 32 is aptly named as it comes stock with 32K of RAM. Unlike the 32K CoCo from Radio Shack, Dragon uses two sets of 16K chips instead of eight 64K chips. There are various hardware differences in the Dragon. The Dragon uses the PAL TV system instead of the U.S. NSTC system. Since there are no VHF TV stations in England, the RF output is on channel 33 and the bandwidth is eight mhz instead of six. The video mixer chip is a LM1889 instead of a MC1372. The power supply (220 volts, 50 hz) is outside the computer and is connected via a DB-9 socket on the baek next to the power switch. The two joystick, cassette and parallel printer jacks are on the left side along with the RF output and reset button. The Program Pak port is on the familiar right hand side but there is no door. The Dragon 32 has one jack that the stock CoCo doesn't have. There is a five pin din jack on the back for video out. This is a decided advantage for those who want to do word processing to its fullest.

To summarize, the Dragon 32 is a fine computer with very close similarities to the Color Computer. The extra features of a better keyboard, parallel printer port, video output jack and white case are a definite advantage. But the best news is that a Dragon will soon be coming to your neighborhood! Starting this summer, a U.S. version of the Dragon will begin production in Louisiana. It will be a NSTC version with a serial port. Price will be very competitive with the Color Computer and will give potential Color Computer buyers another choice.

A Good Program With One Sour Note

As a teacher who works with computers every day, I'm always interested in any new software that has the possibility for use in a classroom setting. As a lover of music, I was especially pleased when *the Rainbow* sent me *Music Drill* from Computer Island to evaluate. This program gives the user a chance to practice sightreading of individual notes, as well as giving an audible feedback of the correct sound of that note. It requires 16K and Extended BASIC.

Every profession has its own buzz words, and this program is what computer using teachers call a "drill and practice" program—that is, it provides drill and practice on skills already taught. This is *not* a "tutorial" type of program that would teach a new skill, so the user must already be somewhat familiar with the musical scale and the placement of notes on the staff. Computers excel at the endless repetition that effective drills need, one reason such programs are so popular with teachers. Good tutorial programs are few and far between. They require careful planning and field testing, and generally require more memory than drill and practice programs. The essential difference between tutorial programs (the teaching of new skills) and drill and practice programs (reinforcement of previously taught skills) should be kept in mind by anyone scleeting software for use in the home or school.

Well, now. You have selected *Music Drill* because you or your kids want a way to practice sightreading of notes, and the association of the sound to its position on the staff. What will you see when you run *Music Drill?* First, you are asked if you want to practice the bass or treble clefs. Then, you are presented with a menu of the keys available, including a "mixed key" mode that will generate notes from randomly selected keys. When you select the key you wish to work in, the computer draws the staff and plays the scale in that key.

Next, a note is placed on the staff but not played. You then indicate your guess by pressing the appropriate key (a, g, f, d, etc.). If the guess is correct, the computer plays the note and places another one on the screen. If the guess is incorrect, the computer gives an obnoxious sound and another chance to guess. After the third incorrect guess, the computer displays the correct answer, the screen clears, and the same note is placed on the screen. This routine is repeated until you press the correct key. (The "stalling" of the program until the user makes the correct response is a sound educational practice, and one I wish would be incorporated into more software intended for classroom or instructional use.)

After 10 notes, the computer displays the time you took to complete the round, and the number of notes you guessed correctly.

The graphics in this program are nicely done. The clefs are pleasingly and correctly shaped, and the notes appear where they should. The sounds generated are as close as the computer can get to the true note (which, in nearly all cases, is pretty close). The documentation, though, is skimpy, both in size (a 3.5" by 4.5" sheet of paper stapled to the inside of a paper folder) and content, but does give the essentials to run the program. This program comes on cassette (which loaded correctly every time I used it) and is written in BASIC.

My one complaint has to do with the scoring. After the third incorrect guess, the computer displays the correct answer, clears the screen, and places the same note on the screen. If the user presses the correct key (remember, they have just seen the answer), that is counted as a correct guess of a new note. Thus, in a round of 10 notes, the user could miss all the notes the first time they are presented (after three guesses), and still receive a score of five out of 10 "correct" guesses, since each round of guesses is considered as a new note.

I feel sure the reason behind this is that the people at Computer Island did not want someone to feel a complete failure, even if they were just beginning to learn music, but the lowest possible score is five out of ten, even if all the notes are missed on the first try. The score report is worded "In ____ minute(s) of drill, you guessed ____ notes correctly on the first try." Now, the user *may* have guessed some of those correctly on the first try, or all of them; or all of those "correct" guesses may have been on the second round (that is, after three incorrect guesses and a look at the correct answer). There is no way to know from the score report.

As it stands, this program can be used for effective drill and practice in the home, but an improved score report (perhaps indicating the number of correct guesses on the first, second, and third tries) would make this program a good bet for classroom use as well.

With the one reservation about scoring, I would recommend this program.

(Computer Island, Dept. R, 227 Hampton Green, Staten Island, NY 10312, $19.95 on tape)

—Mark Williams

Ruler Reader was intended not to teach but to reinforce the skill of measuring to the nearest eighth of an inch. The program, using low-res graphics, draws a ruler and a colored line to be measured, then asks the student to input the number of whole inches and lowest-terms fraction of an inch. There is a graphics diversion after every three questions. This interim reward encourages students to continue, while successful ruler readers get a more extensive graphics "reward" at the end of the program.

Don't plunk your children or students in front of the computer and expect great things, though. The program assumes previous experience with both linear measurement and fractions. After reviewing these concepts, I introduce the program on a classroom monitor and explain its use. I pretend that the TV screen is a "window" over a yardstick. The yardstick extends to the left and right; the line to be measured also extends to the left, out of sight. If necessary, I demonstrate this with a 12-inch ruler and a piece of construction paper with a 2 x 3 inch horizontal rectangle cut out. After they have seen a number of examples on the screen, most students catch on. There is a sample question in the program for review.

Once the instruction is finished, the computer handles the individual drill and practice essential to measurement skills. The computer is no substitute for hands-on activity, but it does help, and the ruler program gives me good results in terms of student interest and performance.

The basic format and fraction manipulation in *Ruler Reader* are adapted from a program by D. B. Allison (80 Micro, April 1982).

Outline

Line	Function
30-70	Introduction
80	Sets values for sample
90	Begin main program
100-160	Set values for each question
180-220	Whole inch routine
230-340	Fraction routine
350-390	Report on line length
410	Increment score
440-460	Report score, offer replay
470-540	Sample question
700	Delay for student response
800	Graphics diversion after three questions
1000-1070	Draw ruler
5000	Graphics reward at end

Variable List

n$	name of student
AS	choose sample
K	random length of line to be measured
K1	adjusts line to ruler
Q	question number being worked
C2	ruler color
C1	line color
N	first number on ruler
ZZ	total number of questions
GH	marker for fraction reduction bypass (avoids division by zero if fraction happens to be EG three whole inches)
F	fractional remainder of line
T	number of tries at inches
V	number of tries at fraction
CA	correct answer in inches
A	student input of whole inches; also variable in fraction reduction
W	delay counter
A$	student input of fraction
L1,L,L$	
R,R$	variables in manipulation of fractions
F	reduced numerator
F1	reduced denominator
Z	exponent counter in fraction reduction
NR	number right
AV	percent correct
Y$	response to replay demand
M	memory location in graphics diversion
C,B	value of character poked to memory
S,T	randomization factors in graphics diversion

```
80..... 01E3
230.... 044A
430.....0763
550.... 0B79
END... 0E0C
```

The listing:

```
10 'RULER READER BY T. GRAY
20 'BASED ON "KING KOMPUTER" BY
D.B. ALLISON, 80 MICRO, APR. 82
30 CLS:PRINT@66,"CAN YOU READ AN
 INCH RULER?              LET'S FIN
D OUT!"
40 PRINT:PRINT"THE RULER WE WILL
 USE IS LARGER THAN NORMAL.  THI
S MAKES IT EASYTO READ, BUT I CA
N ONLY SHOW A  BIT OF IT AT A TI
ME.
50 PRINT:INPUT"TYPE YOUR NAME AN
D PRESS <ENTER>";N$
60 CLS:PRINT@128, "OKAY, "N$", D
O YOU WANT               A SAMPLE QU
ESTION?"
70 INPUT"TYPE Y OR N AND <ENTER>
";A$:IFA$<>"Y"THEN90
80 K1=21:Q=0:C1=8:C2=4:N=1:GOTO1
40
90 CLS:INPUT"OKAY, LET'S START N
OW.              HOW MANY QUESTIONS
DO YOU WANT? (PICK A NUMBER BETW
EEN 5 AND 20)";ZZ:IFZZ<5ORZZ>20O
RZZ<>INT(ZZ)THEN90
100 FOR Q=1TOZZ:N=RND(10)
110 C2=4:C1=RND(7)+1:IFC1=4THEN1
10
120 K=RND(24):IFN>1ANDK<8THEN120
130 K1=2*K+1
140 CLS0:PRINT"QUESTION "Q;:GOSU
B1000:GOSUB2020
150 BH=0
160 IN=INT(K/8):F=K-IN*8:T=0:V=0
:CA=N-1+IN
170 IF Q=0THEN470
180 PRINT@320,"HOW MANY WHOLE IN
CHES":INPUTA
190 IF A=CA THEN PRINT@352,"THAT
'S RIGHT, ";N$:FORW=1TO600:NEXTW
:GOTO 230
200 PRINT@384,"NOPE--TRY AGAIN."
:FORW=1TO600:NEXT:PRINT@384,"
                "
210 T=T+1:IFT<3THEN180
220 PRINT@352, "THE ANSWER IS "C
A" INCHES."
230 PRINT@352:INPUT"WHAT FRACTIO
N OF AN INCH IS LEFT";A$:IFVAL(A
$)=0THENA=0:GOTO270
240 L1=LEN(A$):IFL1<>3THEN270ELS
EL=1:R=1
250 L$=LEFT$(A$,L):R$=RIGHT$(A$,
R):A3=VAL(L$):A4=VAL(R$):IFA4=0
THENA=3ELSEA=A3/A4:IFA4>8ORA4<2T
HEN A=3
260 IF MID$(A$,L+1,1)<>"/"THENA=
3
270 IFBH=1THEN 320
280 FORZ=3TO1STEP-1:IFF/2<>INT(F
/2)THEN300
290 F=F/2:NEXT Z
300 IF Z=0ANDA=0THEN390
310 F1=1:FORP=1TOZ:F1=F1*2:NEXTP
:A1=F/F1
320 IF A1=A AND F1=A4 THEN 350
330 PRINT@416,"NOPE--TRY AGAIN":
V=V+1:BH=1:FORW=1TO700:NEXT
340 IFV=3THEN 370ELSEPRINT@416:G
OTO230
350 IF IN=0 THEN 380ELSEPRINT@41
6,"THAT'S CORRECT, "N$:SOUND120,
3:PRINT@448,"LINE "Q" IS "CA"-"F
"/"F1" INCHES LONG."
360 GOSUB700:GOTO410
370 PRINT@448,"THE CORRECT ANSWE
R IS "CA"-"F"/"F1"INCHES.":GOSUB
700:GOTO 410
380 PRINT@416,"RIGHT, "N$", LINE
 "Q" IS ":PRINT F"/"F1" INCHES L
ONG.":SOUND120,3:GOSUB700:GOTO41
0
```

```
390 PRINT@416, "RIGHT, "N$".":IF
CA=1THENPRINT"LINE "Q" IS ONE IN
CH LONG" ELSE PRINT"LINE "Q" IS
"CA"INCHES LONG.":SOUND120,3
400 GOSUB700
410 IFV=0ANDT=0THEN NR=NR+1
420 IF Q/3=INT(Q/3)THENGOSUB800
430 NEXT Q
440 CLS:AV=INT((NR/ZZ)*100):PRIN
T@160,"YOU GOT ";NR;"RIGHT OUT O
F ";ZZ" FOR A SCORE OF ";AV;" PE
RCENT.":FORW=1TO800:NEXTW:IFAV=1
00THENPRINT"YOU DID VERY WELL. W
ATCH THIS!":FORW=1TO900:NEXTW:GO
SUB5000
450 CLS:PRINT@128,"WANT TO GO AG
AIN? TYPE Y OR N  AND <ENTER>":
INPUTY$:IFY$="Y"THEN NR=0:GOTO60
460 CLS:PRINT"OKAY,"N$". I HOPE
YOU  ENJOYED THE GAME
.":END
470 PRINT@64,"EACH QUESTION WILL
 BE LIKE THIS";:PRINT@320,"HOW M
ANY WHOLE INCHES?":INPUTA:IFA=1T
HENPRINT"THAT'S RIGHT, "N$", "
480 PRINT"IN THIS QUESTION THE A
NSWER IS 1":FORS=64 TO 94:PRINT@
S,CHR$(128):NEXTS:GOSUB700
```



```
490 FORX=320TO415:PRINT@X," ":NE
XT:PRINT@320,"WHAT FRACTION  OF
 AN  INCH  IS LEFT?  (USE LOWEST
 TERMS).":INPUTA$
500 FORX=320TO415:PRINT@X," ":NE
XT:PRINT@320,"REMEMBER, 2/8 AND
4/8 ARE NOT  IN LOWEST TERMS.":
PRINT@384,"THE ANSWER IS 1/4 INC
H.":IFA$="1/4"THENPRINT@416,"SO
YOU WERE RIGHT."
510 PRINT"THE LINE IS 1-1/4 INCH
ES LONG."
520 GOSUB700
530 GOSUB 800:PRINT@64,"IF YOU G
IVE A WRONG ANSWER, I  WILL TEL
L YOU.  YOU WILL HAVE 3 CHANCES
AT EACH PART.  AFTER THETHIRD TR
Y I WILL TELL YOU THE  CORRECT
ANSWER."
540 PRINT:INPUT"DO YOU WANT TO S
EE THE EXAMPLE  AGAIN? TYPE Y OR
 N AND <ENTER>";A$:IFA$="Y"THEN8
0ELSE90
550 END
700 PRINT@15,"PRESS <ENTER>":PRI
NT@47,"TO CONTINUE":A$=INKEY$
710 IF INKEY$=""THEN710
720 RETURN
800 GR=RND(128)+127:FORW=1024TO1
535:POKEW,GR:NEXT:CLS0:RETURN
1000 'DRAW RULER
1010 FOR X=17TO63STEP16:A=5:GOSU
B2000:NEXT:'***INCH MARKERS
1020 FORX=9TO63STEP16:A=3:GOSUB2
000:NEXT:'*** 1/2 INCH MARKERS
1030 FORX=5TO63STEP8:A=2:GOSUB20
00:NEXT:'***1/4 INCH MARKERS
1040 FORX=3TO63STEP4:A=1:GOSUB20
00:NEXT:'***1/8 INCH MARKERS
1050 Y=9:GOSUB2010:Y=17:GOSUB201
0
1060 PRINT@229,N;:PRINT@237,N+1;
:PRINT@245,N+2;
1070 IF N=1THENX=1:A=7:GOSUB2000
1080 RETURN
2000 FORY=10TO10+A:SET(X,Y,C2):N
EXTY:RETURN:
2010 FORX=1TO63:SET(X,Y,C2):NEXT
:RETURN
2020 FORX=1TOK1:SET(X,7,C1):NEXT
:RETURN
5000 'GRAPHICS REWARD
5010 CLS0:FOR T=1TORND(7)+3:C=RN
D(128)+127:B=RND(128)+127:S=RND(
3)
5020 FOR M=1024TO1535STEPS:POKEM
,C:POKEM+1,B:NEXTM,T
5030 C=128:FORM=1024TO1535:POKEM
,C:NEXT
5040 RETURN
```

The Tooth Of Time Byte By Byte

It's a real pleasure to see more educational programs becoming available for the CoCo. Unfortunately, many of these are either poorly written programs or are simply poor learning experiences. However, Crystal Software has put together a program that succeeds on both counts. With five modes, six skill levels, hi-res graphics and sound, *Time Teacher* is designed to teach young children how to tell time.

A menu gives a choice of five modes. The first is a display mode showing both a traditional clock face and a digital clock. All the hours and then all the minutes are shown on the two different clocks. This is basically a count-along-with-the-computer mode.

The next two options are practice modes. In the second mode, the child enters a time into the digital clock using the keyboard. The computer then displays the time on the traditional clock. In the third mode, the child sets the hands of the traditional clock with a joystick. The joystick is positioned as if it were a clock hand, the fire button setting first the hour hand and then the minute hand. The computer then displays the same time on the digital clock.

The last two options are quiz modes with skill levels. The fourth mode prompts the child to select the digits that correspond to a randomly set time on the traditional clock face. In this mode, you do not have to enter leading zeros for single-digit hours (example--04:30). Children can forget or become confused by those leading zeros, and this program wisely avoids them in the quiz mode.

In the fifth mode, the child sets the hands of the traditional clock with a joystick so that it matches the time randomly set on the digital clock.

The quiz modes have six skill levels which are incremented automatically after three correct answers are given in a row. The skill levels are: exact hours; half hours; 15-minute, 10-minute, 5-minute intervals; and all minutes. As students complete a skill level, they are rewarded with a flashing display and a star is placed on the screen.

In both quiz modes, if students answer correctly, they are rewarded with sound and a smiling face on the clock. A wrong answer in a low buzz while the correct answer is displayed. To continue the quiz after a wrong answer, simply press any key.

After or during a quiz, a detailed progress report can be printed on screen or on a printer. The report includes the child's name, the skill level reached, the number of times answered correctly, the total number of questions, a percent score and a weighted score.

Of course, nothing is perfect. The program doesn't provide an easy way to jump to higher skill levels for accomplished students. The random times used in the quiz modes sometimes cause easy times to appear at the higher skill levels or the same time to appear repeatedly. So, a child struggling at one skill level can get a couple of easy or repeated times. The child will then be incremented to the next skill level and be lost there with no way to get back. A possible solution would be to add an automatic decrementing of skill levels after a set number of wrong answers.

All in all, this is an excellent program for the home or classroom. Sound is used effectively throughout, neither overpowering the material, nor making wrong answers entertaining. Graphics are clear and the program is easy to use. Some adult guidance may be needed at first, but kids will quickly learn to use this program without help.

(Crystal Software, 6591 Dawsey Road, Rock Creek, OH 44084, $15.95 on tape)

—James Ventling

Hint . . .

Get The Most Memory

You can get the most memory available on your CoCo by entering the command POKE 25,6:NEW. This, in effect, is a PCLEAR0 on your system.

This command will *not* work with a disk installed. It will, instead, clobber the disk operating system. If that happens, simply RESET CoCo.

IF 'Program Control' *THEN* Fine *ELSE GOSUB* Confusion

By Richard A. White
Rainbow Contributing Editor

Program control is nothing more than decision making and then taking appropriate actions after the decision. To make a decision, the program must make some sort of test. The result of any single test a computer makes is either a true or a false answer. Remember, it's a binary beast even when talking BASIC. If you had only *IF THEN* in your BASIC you still could write any program you wanted. Other control statements like *FOR TO NEXT, ELSE, ON GOSUB* and *ON GOTO* provide for simplicity, speed and efficient use of memory. We have used and commented on each of these statements at one time or another in previous articles. Now is a good time to compare them and suggest ways to use them that you may not have thought about.

IF A=B THEN 100. This is perhaps the simpliest of statements—it means just about what it says. IF A=B GOTO 100 is also acceptable BASIC and means the same thing. The 100 refers to line 100 in either case. If A does not equal B then the program goes to the next line in the program. If you need to go to a subroutine, then write IF A=B GOSUB 1000. IF A=B THEN GOSUB 1000 works also, but why keep the unneeded *THEN* around to take up memory space?

Any group of statements and functions may follow a *THEN*. For example, IF A=B THEN CLS : PRINT A$ A : SOUND 100,10 : GOTO 100. For the actions following the *THEN* to take place it must be *true* that A=B. When it is *false* that A=B, you will want some other action. With *ELSE*, this action can be placed in the same line of code. In most cases this leads to program clarity and simplicity. Our example is IF A=B THEN CLS : PRINT A$ A : SOUND 100,10 : GOTO 100 ELSE CLS : PRINT "NOT EQUAL" : SOUND 50,10. Note that colons are *not* used on either side of *THEN* and *ELSE*. If you want to waste memory you can put them in, they will do nothing else for you. The ability that *ELSE* gives you to choose one piece of code or another is extremely powerful. Without *ELSE* you would be reduced to using *IF THEN* mainly to route the program to lines where the needed code would be placed rather than executing the code directly. This of necessity would locate the code remote from the text calling it. Program clarity is significantly reduced, speed is reduced and memory usage is increased. Instances when you will want to branch to remote lines are when you are going to major program routines or are going to subroutines used by other portions of your program.

> *"The ability that **ELSE** gives you to choose one piece of code or another is extremely powerful. . ."*

IF THEN's can be nested; that is, one following another in the same line with or without intervening code. Following are some lines I used a couple of columns ago, but reformatted to help show the relationships of the keywords to one another.

```
IF (Test 1) THEN (if Test 1 is true)
        IF (Test 2) THEN (if Test 2 is true ACTION A)
        ELSE (if Test 2 is false ACTION B)
ELSE (if Test 1 is false ACTION C).
```

The second *IF* is inset under the first *THEN* that it follows. Each *ELSE* is placed under the *IF* that refers to it. An *ELSE* will be related to the closest IF that does not have an *ELSE*. If the *ELSE* under the second *IF* were removed, then the remaining *ELSE* would automatically be associated with that *IF* even though it is followed by code relating to the first *IF*. If you had no ACTION B in the event that Test 2 is false, put the *ELSE* in anyway so that ACTION C will occur when Test 1 is false.

```
IF (Test 1) THEN (if Test 1 is true)
        IF (Test 2) THEN (if Test 2 is true ACTION A)
        ELSE
ELSE (If Test 1 is false ACTION C).
```

You can test for many things after an *IF*. These include A less than B- A < B, A less than or equal B- A <= B, A not equal B- A <> B, A greater than B-A > B, or A greater than or equal B- A >=B. Of course both sides need not he

(Richard White has a long background with microcomputers and specializes in BASIC programming. With Don Dollberg, he is the author of the TIMS data base management program.)

variables. A > 0 or B = 5 are quite acceptable. You can do the same tests on strings like A$ = "GOLIATH" or A$ > "DAVID." Let's assume that A$ = "GOLIATH." Others might point out that David defeated Goliath and in that sense A$ > "DAVID" is false. Old CoCo could care less about all of this and looks only at the ASCII values for "G" and "D." In ASCII terms, "D" is 68, "G" is 71, and 71 is bigger than 68. The matter is settled right there and "GOLIATH" is greater than "DAVID." If CoCo had to choose between "GOLIATH" and "GARP," it would go on the second letter to find a separation.

Up to now things have been straightforward in that one test was made and one or another action taken depending on whether that test proved true or false. Real life, even in a computer, is that a number of conditions need to be met for some action to take place. I F I can find $600 AND convince my wife that it is necessary, THEN I will buy a disk drive. Well, those who wrote our BASIC saw to it that we would have ways to make decisions based on a number of tests. We have already discussed one way. *IF* (Test 1) *THEN IF* (Test 2) *THEN* (Action). Alternately, this can be written IF (Test 1) AND (Test 2) THEN (Action). In BASIC this might be IF A = B AND A$ > "DAVID" THEN PRINT "SUBSCRIBE TO RAINBOW." To muddy the waters even more this will work—IF A = B AND A$ > "DAVID" OR C > 22 THEN PRINT "SUBSCRIBE TO RAINBOW." See the *OR?*. If C were the contents of your bank account and it were greater than $22 then the message "SUBSCRIBE TO RAINBOW" would be printed irrespective of whether A = B or GOLIATH is greater than DAVID. Tests can be strung together with *AND* and *OR* in almost any conceivable way which makes them powerful tools in the programmer's bag of tricks.

One useful application is enabling either upper or lower case input to control program action. In the June 1983 *Rainbow*, *Mr. Ed*, a simple but powerful word processing program appeared. Its author, Hubert Samm, wrote it for generating and editing assembly language text files. As such, he had little need ever to use it in lower case and provided no way to input lower case commands. I came to discover that the Dugger C compiler that I had recently purchased required characters in the text file that were not supported by *Telewriter*. These are the back slash up arrow and the square brackets in particular. However, much of "C" is written in lowercase which made *Mr. Ed* difficult to use. So with some simple reprogramming, lower case came to *Mr. Ed* by ORing a lower case letter in each *IF* statement. The edited lines of code are below for those who may want to enhance *Mr. Ed*.

```
60 TIMER=0:C$="A":C=1

70 IFC$="N"ORC$="n"GOSUB370

80 IFC$="I"ORC$="i"GOSUB410

90 IFC$="T"ORC$="t"GOSUB450

100 IFC$="D"ORC$="d"GOSUB460
```

```
110 IFC$="F"ORC$="f"GOSUB470
120 IFC$="E"ORC$="e"GOSUB540
130 IFC$="X"ORC$="x"GOSUB620
140 IFC$="R"ORC$="r"GOSUB640
150 IFC$="L"ORC$="l"GOSUB650
160 IFC$="P"ORC$="p"GOSUB720
170 IFC$="Q"ORC$="q"THENEND
180 IFC$="G"ORC$="g"GOSUB780
190 IFC$="C"ORC$="c"GOSUB870
200 IFC$="U"ORC$="u"GOSUB1140
210 IFC$="2"GOSUB1160
211 IFC$="K"ORC$="k"GOSUB1200
212 IFC$=CHR$(94)GOSUB1300
215 IFC$="M"ORC$="m"THENK$=TX$(CL)
    :GOSUB620
217 IF C < 10 THEN C=C+1 : GOTO 70
220 T=TIMER/60:PRINTT" SECONDS"
```

Lines 60, 217 and 220 are my addition to time the speed of execution and are *not* to be changed or added to the *Mr. Ed* program. Also note in line 212 that I had to use CHR(94) rather than an up arrow, another character not supported by *Telewriter*. In this case it's no problem since BASIC fully understands CHR$(94). If you were to put the above code in the computer and run it, it would print 1.3 seconds(approximately). That is the amount of time to loop through the tests only 10 times. This is a worst-case-plus condition; it will take less than one tenth of a second to test and take action. Still, response time is the name of the game and I will propose a fix later. I initially used a *FOR TO NEXT* loop to control the looping but changed to using an *IF THEN* construction for this article. C is the loop counter. It is initialized, C=1 in line 60. In line 217, it is tested and incremented if less than ten and the program is sent back to line 70 for another pass. The function is exactly the same as *FOR TO NEXT*, but a bit slower and not as memory efficient. Indeed, the logic is the same as *FOR TO NEXT* with the counter initialized at the beginning, and tested at the end. In both cases if the counter stays less than the ending count (or greater than FOR we can loop down-count as well as up-count), the program loops back.

As noted, *FOR TO NEXT* can be replaced with a simple *IF THEN* statement. So why use *FOR TO NEXT*? It's easier. All you and the program need to know about the loop are right up there in the line heading up the loop. *FOR TO NEXT* is faster since BASIC knows up front what you want and can use the most efficient machine language coding to

get it done. In the *IF THEN* approach above, the BASIC interpreter never knows it is doing loops. Here is the basic syntax of *FOR TO STEP NEXT*.

FOR (variable & initial value) *TO* (limit value) *STEP* (value) . . . *NEXT*

The initial value, limit value and step value may be positive, negative, integer or decimal. Looping continues as long as the variable remains less than (or more than) the limit value. Any of the following are valid.

FOR C = 1 TO 10 . . . NEXT
FOR C = 132.769 TO 133.543 STEP .008 . . . NEXT
FOR C = 1000 TO -732 STEP -200 . . . NEXT
FOR C = -1 TO -10 STEP -1 . . . NEXT
FOR C = 1E1 TO 1E35 STEP 9E10 . . . NEXT
FOR C = 0 TO 0 . . . NEXT

*"In some listings, you will see the variable name after **NEXT** like **NEXT C**... The computer knows full well which variable it is dealing with and doesn't need the programmer telling it."*

Without *STEP*, BASIC assumed an increment of +1, which is a very typical case. Otherwise there is nothing special about *STEPing*. Just make sure you *STEP* in the right direction to get from the initial value to the limit one. No matter what the values assigned, a *FOR TO NEXT* loop is always executed at least once. It is necessary for the program to get to *NEXT* before any test is made. In some listings, you will see the variable name after *NEXT* like *NEXT C*. No, you don't see it above. The computer knows full well which variable it is dealing with and doesn't need the programmer telling it. From a programming clarity viewpoint, some case can be made for documenting the *NEXT* with its associated variable.

BASIC stores the location of the start of the *FOR TO NEXT* loop on the microprocessor's stack in memory. When the program exits normally, these memory locations are cleared for other uses. When a *FOR TO NEXT* loop is abnormally exited, these memory locations are not cleared. There are occasions when early departure from a loop is needed. There are a number of ways to do this. When this done, there should be code to set the count variable to a number equal or higher than the limit value and follow this with a *NEXT* in order to clear the stack. Here is an example.

```
100 FOR C=0 TO X : IF Y>Z THEN C=X : NEXT
  GOTO 200
110 ' CODE IN THE LOOP
120 NEXT
130 ' CODE FOLLOWING THE LOOP
```

There are times when it is not desirable to go through the loop even once. In our BASIC you must test the governing condition and jump around the *FOR TO NEXT* code or abandon *FOR TO NEXT* writing your own loop control

with IF THEN. Here is one approach.

```
100 A = 1
105 IF A > X GOTO 150
110 'LOOP CODE
      .
      .
      .
140 A=A+1:GOTO 105
```

One reason for the need to do this would be if you were takinga series of strings apart, perhaps for printing fields of records. If the user had put no data in yet and X = 0, but made the mistake of trying to print, a SN ERROR would greet their efforts as soon as the program tried to take an empty string apart. Now, no programmer wants a user to be greeted with a SN ERROR—ever. So constructions like the above are necessary now and then. Some languages and dialects provide alternate statements like DO UNTIL that test first and then act. What is most important is to know what the characteristics of your language are and use them to best advantage. The grass may look greener over the PASCAL fence. That does not mean it's better grass. And while we are at it, you should be aware that the same keywords may act differently in another language. *IF THEN ELSE* seems to be one of the few constants.

The two other program control statements in Color BASIC are ON Z GOSUB and ON Z GOTO, Z is a number starting with 1 which tells the computer to count to a specific line number listed after the *GOSUB* or *GOTO* and go to that line. Earlier I promised an improvement for *Mr. Ed*. Here it is.


```
50 TIMER=0:C$="A"
60 FOR X=1TO10
80 B=INSTR(1,"NnliTtDdFfEeXxRrLlPpGgCcUu2Kk
   1A",C$)
90 ONB GOSUB 370, 370, 410, 410, 450, 450, 460, 460,
   470, 470, 540, 540, 620, 620, 640, 640, 650,650, 720,720,
   780, 780, 870, 870, 1140, 1140, 1160, 1200, 1200, 1300,
   225
215 IF C$="M"ORC$="m"THENK$=TX$(CL):
   GOSUB620
220 NEXT:T=TIMER/60:PRINTT
225 RETURN
```

As before, there are some changes that were made to this code piece for timing purposes that should *not* be made in *Mr. Ed*. These include lines 50, 60, 220 and 225 above which should *not* be changed if you modify the program. This code does the same thing as the *IF THEN* line we discussed before, but does it in one third the time. Further, it uses only 360 bytes versus 716 bytes for the series of *IF THEN* statements. The *INSTR* statement in line 80 hunts through the string of characters trying to find a match for the character in C$. If it finds a match, the test is successful and it puts the count up the string in B. In ON B GOSUB the computer counts up line numbers to the number in B and makes a subroutine call to that line number. For the timing text, I set C$="A" and put A at the end of the string in INSTR. INSTR sets B=31 and the 31st number in 90 is 225. The program goes to 225, returns and then goes on line 220 where it finds the NEXT for another cycle. Obviously the INSTR-ON B . . . technique is quite effective here. If you only had three or four characters or values to test, *IF THEN* would be preferred.

Now, I am aware that *INSTR* is not in Color BASIC, and those without Extended BASIC are out in the cold again. Not necessarily so. *INSTR* is a command for which a BASIC subroutine may be written as a substitute. Take heart, Color BASIC owners, and use the following.

```
10 'SUBSTITUTE FOR INSTR. DEFINE A START
   LOCATION SL, A TARGET STRING TS$ AND A
   STRING OF CHARACTERS, C$, TO SEARCH
   FOR IN YOUR PROGRAM AND DO 'GOSUB 10'.
12 LT = LEN (TS$) : LC = LEN (C$)
14 IFLC+SL>:LT THEN SL=0 : RETURN ELSE F$=
   MID$(TS$, SL, LC) : IF F$ = C$ THEN RETURN
   ELSE 14
16 'IF C$ IS FOUND IN TS$, SL WILL CONTAIN
   THE POSITION IN TS$ WHERE C$ STARTS. IF
   C$ IS NOW FOUND, SL WILL CONTAIN 0.
```

To summarize, program control is accomplished with statements that perform one or more tests and take actions based on whether true or false results are obtained for each test. What action takes place depends on how the programmer writes the statements. The control tools in Color BASIC are *IF THEN ELSE, FOR TO STEP NEXT, ON GOTO* and *ON GOSUB*. *IF THEN ELSE* is extremely powerful and could be used only without seriously limiting one's ability to write complex programs. The other statements provide for program clarity, speed and economy of memory. Each has its place, so serious programmers learn to use each to best advantage through in-depth understanding of each.

Vocabulary Builder Gets Constructive Criticism

By Dr. Charles H. Santee

Vocabulary Builder from Computer Island deals with an important area of learning and is based on some good ideas; however, the program has much room for improvement. The program begins with a modest title page and there is a short pause while data is being read. You are then asked to give your name and to tell if you have a printer on. If you respond with "yes" or "Y" the printer prints "vocabulary words for" and the name entered.

A page appears which is divided into several sections. The top part of the page identifies the number of words presented and the level. This is followed by a short question to answer. Four answers are presented one at a time for a brief period while a tone plays and then all four answers are reprinted for viewing. The person using the program selects the letter in front of one of the answers. If the answer is not correct, the program plays a low tone and the correct answer is shown. A correct answer is greeted with a higher tone and the message "Correct" followed by the person's name. You are then given a message to press ENTER which takes you on to the next question. After 10 responses, the program prints the words needed for study on the screen and with the printer (if that option was selected). After five sets of 10 words the program tells you that you have completed that set and shows the percent correct.

The documentation states that the program is modifiable by the user and gives some short directions. You can purchase programs for three different levels (grades 3—5, 6—8 and 9—12). The program does use a consistent method for entering information and answers. You would always respond by typing an answer and pressing ENTER. Each program in the series contains 200 questions and 1,000 words.

I am a teacher, but I don't claim to be an expert in teaching vocabulary. However, I feel the selection of questions and responses was generally good. In all cases, there appeared to be a clear best answer. There were a few times that I feel the correct answer was an unusual nuance of a word. On those occasions, deriving the meaning of the word from the correct answer may be somewhat misleading. However, this was an exception rather than the rule.

The documentation does not mention what source was used to select the words used and how the level was determined. I think this is important information that should be included.

Now for the bad news. In *Vocabulary Builder,* questions are selected at random. This is generally a good practice in educational software. However, the specific application in this instance has an unfortunate result. Each question is selected at random from the entire list of 200 words. The question is then eliminated from the pool of questions so that you can be certain the same question is not given again. At the end of 50 questions, the program is *RUN* so that the entire pool of 200 words is again available. You might think that doesn't sound so bad. However, using this method you may *never* see all the words in the list. In fact, there is a good probability you could use the program a large number of times and miss some of the words.

My next area of concern was the amount of memory used by this program. The program requires 32K of memory. In fact, with a disk drive installed, you need to *PCLEAR 1* before *RUNning* the program. My own particular feeling is that this program is not worthy of that much memory use. Most of the memory is used by the list of words and questions.

The easiest solution to both problems listed above would be to make four separate programs with 50 words each. Then all questions could be tested and the program would fit into a 16K machine. In fact, I would venture to say this program could be modified to work in 4K. I would recommend putting a series of smaller programs on one cassette. The user could master the words in one program and then progress to the next. I have another better suggestion which I will outline at the end of this review, so don't stop reading here.

The *Vocabulary Builder* documentation outlines a brief description of how to enter data lines to modify the program. You need to enter a question, four answers, the letter of the correct answer, and the correct answer. The documentation states that you must keep the same number of items as the original program. If the documentation had shown how to modify the lines where variables were initialized, you could have easily set this program to accept data lines of any quantity desired. Second, the correct answer is given twice (once in the list of questions and again as the last data item). Since the letter of the correct answer is included, the program could have used this letter as a pointer to the answer and the last data item could have been dropped. This would mean that the program would need less memory and modifying the program would require less typing.

I believe that some of the other problems of this program

were due to an attempt to keep the program short so that a large number of items could be included. Since we have solved that problem, let's see how we might improve the rest of the program. The program uses a standard *INPUT* statement to accept data. This means that the CLEAR key will operate and a child could accidentally erase the entire screen. It would be possible to eliminate the problem by using an *INKEY* routine. The *INKEY* routine could also provide better protection for wrong responses. At the end of 50 questions, the program accepted an ENTER to a yes/no question. I would like the program to only respond to the correct response of YES or NO (Y/N) or to give a message of help when wrong keys are pressed.

When the program asks if the printer is on, a child could accidentally answer YES and the program would hang up looking for the printer. I would like a routine that would check the printer status. We don't need to give kids any hang-ups.

At the end of 10 items, the list of items missed will show about seven items on the screen. If the child misses more than seven, the first few items scroll out of view. I would like to see a routine that would control how much is put on the screen at that point (ie, show seven missed items and then press a key to see the rest).

Although not stated in the documentation, the largest question should be no larger than 28 characters. Second, each answer should be no larger than 14 characters. Data items larger than this do not fit into the format of the screen. With very little modification, this program could be used not only for vocabulary but for any multiple choice type question. However, even for vocabulary I would like to have twice as much space to write a question. The space for answers is adequate for most words, but I might like to add a phrase or two as an option for an answer.

I found the tone and response to each answer monotonous and after a while annoying. I would like to see more variety. Let's add a little flash after 10 correct answers. The rate at which questions were presented was just a little too slow. The author chose to use a *FOR-NEXT* statement with a *POKE* of each memory location to make the background colors in certain screen areas. I believe printing a graphic string would have been quicker.

In my opinion, this program did not make adequate use of the capabilities of a computer. The same thing could have been accomplished by using index cards with questions on one side and answers on the other side.

Let me give an example of what I would like to see in a program to develop vocabulary. This would work best with a disk, but would be feaslble for cassette. The program would have all the questions and vocabulary words saved in several data files that were separate from the program. The program would call up the first file and test the student. The items the child missed would be saved back to disk. The items the child didn't try would be saved as another file. The next time the program was used, the program would call back a few items the child missed the last time and add some new words. The program would progress in this fashion until all words were mastered. While we are at it, let's measure how fast the child responds, how many items a child responds to in one sitting, and the classification of words the child misses (ie, easy-hard or nouns vs. verbs). Let's give the screen some animation and variety as the program (and the child) progresses. Finally, why not include one printout for the child for feedback and review and a second that is activated by a teacher's code word or program for diagnosis of learning needs.

In conclusion, I felt that *Vocabulary Builder* dealt with important subject matter, had a few interesting features (such as printing words missed) but was generally quite disappointing.

(Computer Island, 227 Hampton Green, Staten Island, NY 10312, 32K, three levels at $19.95 each)

Match & Spell Sparks Enthusiasm For Learning

Does your child need some improvement in spelling skills? Perhaps he or she has lost interest in that old spelling list program you wrote. If so, spark up some new enthusiasm for learning and let the kids play *Match & Spell* on their favorite computer. Kids will *ask* to practice their spelling words once they have played this educational game!

Did I say game? The kids in our house and neighborhood think it's a game. My kids, age four, six, and nine, are really excited about *Match & Spell*. This program very nicely incorporated the TV matching game of Concentration with a spelling drill of up to 20 words. Each word in the list can be up to 11 characters in length. The object of the game is to collect squares by matching up the hidden graphics. The educational objective is to improve spelling skills, and my kids have had fun doing both.

After *CLOADing Match & Spell*, you can watch the title page *write* the words "match and spell." From then on, the program very effectively prompts you to do everything you need to do. By the way, the documentation is very clear and well written, from the general setup of the program to the thorough description of the game play and what to expect as the game proceeds.

You start off by loading the word list, either from the keyboard or from a tape file you may have created before. The program comes with a sample spelling list to get you started. However, the ('LIST') is at the very end of the tape after three program saves. So if you don't know where it is you have to wait and wait. Creating your own word list is very easy through the program prompts and includes a chance to change a wrong spelling or change words entirely before saving to tape or using them in the program.

Once the word list is entered, the words are displayed in two columns for you to study as long as you want. When you are ready the game begins. The game prompts you for two players' names, although one person could play both positions. A new screen appears with 12 numbered blue squares. Each player, in turn, chooses two squares in an effort to match two with the same hidden graphics. If those two squares match, the program randomly selects one of the words from the list and presents it to you either correctly or misspelled with a common spelling error. For example, ('computer') might be presented as ('computor.') If this question is answered correctly, the word is removed from the screen and the player is required to type it in correctly, being given three chances to do so. If he is unable to spell the word correctly, he forfeits these two squares and his turn. If a wrong answer is given to the previous question, the player also loses his turn and the squares he had just matched are turned around and become available for the other player if he can correctly match and spell the next word chosen.

If a right spelling is given, then the matched squares are colored in with that player's color and the player is given another turn. Play continues until all 12 squares have been matched. Chances are you would have to play the game at least twice to cover a full spelling list. At the end of the game, the program lists any words that a player had difficulty with and gives you the option of continuing another game with the same word list or using a new list.

I think *Match & Spell* is an excellent program and provides a well balanced and effective intermingling of play time with spelling drills, in contrast to some spelling/math drills which require you to reach a certain performance before being rewarded with play.

It was also noted in the documentation that the random misspelling of the words works best on words of five or more letters. I found this to be true. The program runs well and is *very* easy to use. (My wife wanted me to be sure and say that, as she has commented that *Match & Spell* is one of the easiest programs she has had to help the younger kids with when I'm not home.) Younger children, around age six and under, may need some help and guidance to read the prompts, etc. at first, but they catch on fast! The word list can be changed readily to accommodate appropriate vocabulary. For my youngest daughter, age four, we used words like family members' names, her name, dog, cat, etc.

It may take longer to go through a spelling list using *Match & Spell*, but the kids think it's neat and really enjoy it. Plus, there is no "Awww. Do I have to practice my spelling now?"

Documentation is complete and well written. Really, the only criticisms I had was having to wait so long for the sample list to load and waiting during the delay time between screens after you've made a match and while the computer is selecting a word and misspelling it. It bothered me slightly, but the kids apparently didn't mind as they have never mentioned it.

If you are the programming type you might consider adapting this program to math drills.

This educational game is for all ages (including mom and dad). *Match & Spell* requires 16K Extended Color BASIC.

(Harmonyes, P.O. Box 1573, Salt Lake City, UT 84110, $11.95 tape)

—Kenneth D. Peters

How To Clear All Graphics Pages

You can clear all your graphics pages—and thus increase free memory—by entering the following command: *POKE25,6:NEW*. If you do this, you will not be able to use graphics pages, but you will have more program memory.

Caution: Do not try this POKE with any program in memory. It will destroy whatever program is resident in the 80C.

The Write Idea

USING THE COLOR COMPUTER TO DEVELOP COMPOSITION TOPICS

By Stephen N. Tchudi

As a teacher of school and college writing courses, I have been somewhat disappointed by the computer software programs that have been written for my field. Whether produced by major educational publishers, the computer manufacturers, or basement hobbyists, programs designed to develop language seem to me inordinately concerned with drill in so-called "basic skills"—particularly grammar, vocabulary and spelling. Now a computer can do that sort of drill very effectively, probably more effectively than an English teacher and with a good deal more pizzazz. But using a computer principally for drill seems to me to ignore its much broader potential for aiding youngsters in developing their language skills. Further, the bulk of educational research suggests that drill, even when done well, is not the best way to get students to use language more effectively. Language is a "learn by doing" skill; the more people use it, the better they become at using it.

Since buying my CoCo a little over a year ago, I have been studying language programs and exploring ways to use the computer "creatively" to help students learn to write. The program presented here—I call it "The Write Idea," but "The Wryte Idea" might be better—is intended to help students from the middle grades through adulthood solve one of the biggest barriers to successful writing: finding something to say.

(Stephen Tchudi is a professor of English at Michigan State University, where he directs the English Education Program. He is a member of the Lansing, Michigan Color Computer Users Group.)

Most novice writers undervalue their own experience and fail to realize that they have hundreds and hundreds of potential writing topics in their memories. "The Write Idea" helps a student get some of these ideas out of his or her mind and down on paper.

The program is written for a 16K CoCo and uses a Line VII printer. As the student answers questions flashed on the monitor, the printer springs to life, writing down the answers in the Line VII's CHR$ 31 boldface for dramatic effect. (The answers simultaneously appear on variously colored monitor screens.) The printer also adds headings, formats the answers, and throws in some asterisks to highlight the student's ideas.

The program is written in two parts: Part I helps the writer list a number of potential topics and select one to write about; Part II has the student develop specific ideas as a way of "pre-writing" and planning the paper. In a twenty to thirty minute run-through of the program, a student will characteristically produce a three- to five-foot printout, ending forever the complaint that English teachers hear so often, "I haven't got anything to write about."

* * *

Part I

Lines 5—130 clear string space and present the program title.

Lines 140—290 explain the aim of the program, ask the student's name and instruct him/her to turn on the printer.

Lines 300—1190 quiz the writer about past experiences, asking him/her to "peek and poke" in memory to recall "Firsts" (first love, first pet, first day at school [lines 300-

610]), "Celebrations" (birthdays, Christmas/Chanukah, Valentine's Day [lines 620—950]) and "Events" [lines 960—1190].

Lines 1200—1420 then ask the student to examine the printout and to select a "best" topic to write about. Students should also save the printout for future reference. Typically a student will create more than one and possibly dozens of potential writing ideas, each of which can be developed separately through Part II.

Lines 1430—end give the student the option of plunging directly into drafting a paper or, if more preparation is needed, moving on to Part II.

Part II

Lines 2000—2190 introduce this portion and ask the student to re-identify him or herself, turn on the printer and enter the topic that has been chosen.

Lines 2210—2310 have the student describe the general idea of the topic.

Lines 2320—2840 ask the writer to brainstorm for details about the topic, including Sights [2340—2450], Sounds [2460—2540], Other Senses [2550—2640], Actions [2660—2680] and People [2760—2840].

Lines 2850—3020 focus on the audience or intended readership of the paper. Who will read it? What does the writer want them to understand?

Lines 3030—3210 ask the writer to jot down some sentences that can be used as a possible beginning, middle and ending.

Lines 3230—end conclude the program by pointing out that the writer has, in effect, planned the entire paper, that the only remaining task is drafting. The text of the program also reminds the student that people, not computers, write papers; that there is no electronic substitute for a writer.

* * *

The model provided by "The Write Idea" can be modified to treat virtually any kind of writing, from exposition and argument to letter writing or even creative writing. Unlike drill programs, it does not insist on "right" answers; rather, it uses the computer as a stimulus to creativity and as an electronic notepad. I believe programs like it deserve a place in the computer-oriented language laboratories springing up in schools and colleges all over the country. I hope such programs will replace the "old wine" of skill-and-drill programs with a richer vintage more appropriate to the new bottle of the Color Computer.

```
150 ___ 01E4
390 ___ 04DE
610 ___ 074F
880 ___ 0A6E
1110 __ 0CE6
1400 __ 102E
END __ 12EO
```

Listing 1:

```
5 CLEAR 2000
10 CLS
20 PRINT @  97, "T H E    W R I
T E    I D E A":PRINT
30 PRINT @ 166,"'FINDING SOMETHI
NG TO"
40 PRINT @ 202,"WRITE ABOUT"
50 PRINT @ 229, "FROM PERSONAL E
XPERIENCE'"
60 PRINT:PRINT
```

```
70 PRINT @ 330,"VERSION 1.4"
80 PRINT @ 358, "COPYRIGHT (C) 1
983"
90 PRINT @ 392, "STEPHEN TCHUDI"
100 PRINT @ 423,"**THE INKBYTE**
"
110 PRINT @ 455,"2011 PAWNEE TRA
IL"
120 PRINT @ 465,"OKEMOS, MICHIGA
N 48864"
130 FOR T= 1 TO 5000:NEXT T:CLS(
3)
140 PRINT "LOTS OF PEOPLE HAVE D
IFFICULTY  FINDING SOMETHING TO
WRITE       ABOUT."
150 GOSUB 1530
160 PRINT "THIS PROGRAM WILL HEL
P YOU FIND IDEAS IN YOUR OWN EXP
ERIENCE."
170 GOSUB 1530
180 PRINT "ALL YOU HAVE TO DO IS
 PEEK AND  POKE IN YOUR MEMORY T
O RECALL   INTERESTING EVENTS AN
D PEOPLE."
190 GOSUB 1530
200 PRINT "THE COMPUTER WILL ASK
 YOU A FEW QUESTIONS."
210 GOSUB 1530
```



```
220 PRINT "PRESS <ENTER> WHEN RE
ADY":INPUT G$:CLS(6)
230 PRINT
240 PRINT "WHAT IS YOUR NAME?":I
NPUT NAME$:PRINT
250 PRINT "OK, " NAME$
260 FOR T=1 TO 500:NEXT T:PRINT
270 PRINT "TURN ON YOUR PRINTER.
"
280 PRINT: PRINT "PRESS <ENTER>
WHEN THE PRINTER  IS ON."
290 INPUT T$:CLS(5)
300 PRINT "WE'LL BEGIN WITH A CA
TEGORY      CALLED:"
310 FOR T=1 TO 500:NEXT T:PRINT
320 PRINT "     * * F I R S T S *
 * "
330 GOSUB 1530
340 PRINT "THINK OF SOME OF THE
'FIRSTS' INYOUR LIFE:"
350 GOSUB 1540
360 PRINT "  FIRST LOVE      OR F
IRST KISS"
370 GOSUB 1540
380 PRINT "  FIRST PET          FI
RST DEATH"
390 GOSUB 1540
400 PRINT "  FIRST DAY AT SCHOOL
"
410 GOSUB 1540
420 PRINT "  FIRST REMEMBERED NI
GHTMARE"
430 GOSUB 1530
440 PRINT "OK, GOT THE IDEA?"
450 GOSUB 1530
460 PRINT "PRESS <ENTER> WHEN RE
ADY": INPUT G$: CLS(0)
470 PRINT
480 PRINT "NOW LIST ONE OF THE '
FIRSTS' IN YOUR LIFE. THEN PRESS
 <ENTER>."
490 PRINT
500 INPUT S$
510 PRINT #-2, CHR$(31); "SOME '
FIRSTS' IN THE LIFE OF ": PRINT
#-2, NAME$:PRINT #-2
520 PRINT #-2, "   ***" S$:PRINT
 #-2
530 PRINT: PRINT: PRINT "GOOD"
540 GOSUB 1540: CLS(0)
550 PRINT "NOW LIST ANOTHER FIRS
T IN YOUR   LIFE. <ENTER> (IF YO
U'RE OUT OF 'FIRSTS', PRESS <XXX
 ENTER>.)
560 PRINT
570 INPUT S$
580 IF S$="XXX" THEN 620
590 PRINT #-2, "   ***" S$:PRINT
 #-2
600 PRINT:PRINT:PRINT "KEEP GOIN
```

```
8":FOR T = 1 TO 500: NEXT T
610 GOTO 540
620 PRINT "OK, " NAME$:PRINT "LE
T'S TRY ANOTHER CATEGORY"
630 FOR T=1 TO 500:NEXT T:CLS(2)
640 FOR T=1 TO 200:NEXT T: PRINT
650 PRINT " * * C E L E B R A T
I O N S * *"
660 GOSUB 1530
670 PRINT "LIST SOME OF THE MOST
 IMPORTANT CELEBRATIONS IN YOUR
LIFE."
680 FOR T = 1 TO 1000:NEXT T:PRI
NT
690 PRINT " FIRST REMEMBERED BIR
THDAY"
700 GOSUB 1540
710 PRINT " OTHER BIRTHDAYS   UN
BIRTHDAYS"
720 GOSUB 1540
730 PRINT " CHRISTMAS
 CHANUKKA"
740 GOSUB 1540
750 PRINT " JULY 4TH        VALEN
TINES DAY"
760 GOSUB 1540
780 GOSUB 1540
790 PRINT " ? ! ? GROUND HOG'S D
AY ! ? !"
800 GOSUB 1530
810 PRINT "OK? PRESS <ENTER> WHE
N READY.": INPUT G$:CLS(5)
820 FOR T = 1 TO 500:NEXT T:PRIN
T
830 PRINT "NOW LIST ONE OF THE C
ELEBRATIONSIN YOUR LIFE. THEN PR
ESS <ENTER>"
840 INPUT P$
850 PRINT #-2, "CELEBRATIONS IN
THE LIFE OF":PRINT #-2, NAME$:PR
INT #-2
860 PRINT #-2,"   ***" P$:PRINT
#-2
870 PRINT:PRINT:PRINT "THAT GETS
 YOU STARTED"
880 GOSUB 1540: CLS(5)
890 PRINT "NOW LIST ANOTHER CELE
BRATION IN YOUR LIFE. <ENTER>"
900 PRINT:PRINT "(PRESS <XXX ENT
ER> IF YOU'RE OUT OF IDEAS."
910 INPUT P$
920 IF P$="XXX" THEN 960
930 PRINT #-2,"   ***"P$:PRINT #
-2
950 GOTO 880
960 CLS(2):PRINT: PRINT "LET'S D
```

```
O ONE MORE CATEGORY."
980 FOR T = 1 TO 250:NEXT T:PRIN
T
990 PRINT "      * * E V E N T S
 * * "
1000 GOSUB 1530
1010 PRINT "                OR"
1020 GOSUB 1530
1030 PRINT "   * * H A P P E N I
 N G S * * "
1040 GOSUB 1530
1050 PRINT "BY NOW YOU KNOW THE
GAME, SO BE-GIN LISTING IMPORTAN
T EVENTS IN YOUR EXPERIENCE."
1060 GOSUB 1530
1070 PRINT "TYPE IN AN IMPORTANT
 EVENT": PRINT
1080 INPUT E$
1090 PRINT #-2,"EVENTS IN THE LI
FE OF": PRINT #-2, NAME$:PRINT #
-2
1100 PRINT #-2, "   ***" E$:PRIN
T #-2
1110 GOSUB 1540:CLS(2)
1120 PRINT: PRINT "OK, " NAME$
1130 PRINT: PRINT "NOW LIST ANOT
HER EVENT": PRINT
1140 PRINT "PRESS <ENTER> WHEN F
```



```
INISHED":PRINT
1150 PRINT "IF YOU'RE OUT OF IDE
AS, PRESS   <XXX ENTER>"
1160 INPUT E$
1170 IF E$= "XXX" THEN 1200
1180 PRINT #-2, "   ***"E$:PRINT
 #-2
1190 GOTO 1110
1200 CLS(4)
1210 PRINT:PRINT "THAT'S THE END
 OF THE IDEA GATH-ERING.":GOSUB
1530
1220 PRINT "NOW LET'S PICK A TOP
IC TO WRITE ABOUT.": GOSUB 1530
1230 PRINT "TAKE A LOOK AT YOUR
PRINTOUT.""
1240 GOSUB 1530
1250 PRINT "WHICH OF THE IDEAS O
N THE PRINT-OUT ARE MOST INTERES
TING TO YOU?":GOSUB 1530
1260 PRINT "PRESS <ENTER> WHEN R
EADY TO PRO-CEED.":INPUT G$:CLS(
4)
1290 CLS(4)
1300 PRINT:PRINT "OF ALL THOSE I
DEAS, WHICH ONE IS THE MOST APPE
ALING TO YOU?":GOSUB 1530
1330 PRINT
1340 PRINT "IT WILL MAKE A GREAT
 WRITING    IDEA!":GOSUB 1530
1350 PRINT "TYPE IN THE TOPIC.":
PRINT
1360 INPUT T$
1370 PRINT #-2, CHR$(31); "THE T
OPIC SELECTED BY " NAME$ " IS:"
1380 PRINT #-2:PRINT #-2, "
 " T$
1390 CLS(4):PRINT "GOOD, " NAME$
1400 GOSUB 1530
1410 PRINT "THE TOPIC YOU'VE PIC
KED IS:"
1420 GOSUB 1530
1430 PRINT "   "T$:GOSUB 1530
1440 PRINT "YOU MAY FEEL READY T
O WRITE     ABOUT IT RIGHT NOW."
:GOSUB 1530
1450 PRINT "IF SO, SAVE YOUR PRI
NTOUT,":GOSUB 1540
1460 PRINT "     POP IN THE WORD
 PROCESSING      CARTRIDGE,":GO
SUB 1540
1470 PRINT "          AND START
DRAFTING.":GOSUB 1530
1480 PRINT "IF NOT, PRESS <ENTER
>":INPUT G$:CLS(4)
1490 PRINT "IF YOU WANT MORE HEL
P BEFORE    WRITING,":GOSUB 1530
1500 PRINT "LOAD THE NEXT PROGRA
M ON THE    CASSETTE.": GOSUB 15
30
```

```
1501 PRINT "  1. PRESS <BREAK>."
1502 PRINT "  2. TYPE <NEW ENTER
>"
1503 PRINT "  3. PREPARE RECORDE
R TO <PLAY>"
1504 PRINT "  4. TYPE <CLOAD ""I
DEAS 2 "" ENTER>."
1510 PRINT @ 426, "**THAT'S ALL
FOR NOW"
1520 GOTO 1520
1530 FOR T = 1 TO 1500:NEXT T:PR
INT:RETURN
1540 FOR T=1 TO 1000:NEXT T:RETU
RN
```

```
2170 ___ 0263
2340 ___05C2
2560 ___086A
2860 ___0BF5
3040 ___0F0A
3260 ___11F6
END ___ 14CA
```

Listing 2:

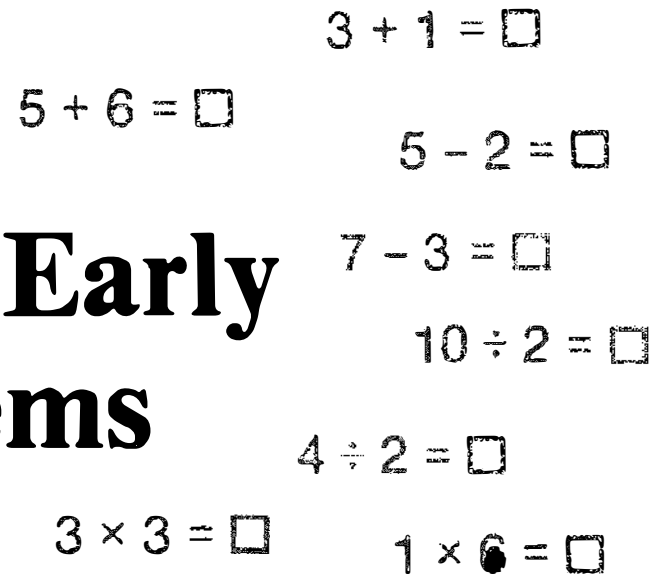
```
2000 CLEAR 2000
2010 CLS
2020 PRINT @ 97,"T H E     W R I
T E     I D E A"
2030 PRINT @ 172, "PART TWO"
2040 PRINT @ 225, "'DEVELOPING Y
OUR WRITING IDEA'"
2050 PRINT @ 330, "VERSION 1.3"
2060 PRINT @ 358, "COPYRIGHT (C)
 1983"
2070 PRINT @ 392, "STEPHEN TCHUD
I"
2080 PRINT @ 423,"**THE INKBYTE*
*"
2090 PRINT @ 455, "2011 PAWNEE T
RAIL"
2100 PRINT @ 485,"OKEMOS, MICHIG
AN 48864"
2110 FOR T = 1 TO 5000:NEXT T:CL
S(5)
2120 FOR T = 1 TO 500:NEXT T
2130 PRINT:PRINT "I REGRET TO SA
Y I HAVE FORGOTTENYOUR NAME.":GO
SUB 3430
2140 PRINT "PLEASE ENTER IT AGAI
N":PRINT:INPUT NAME$:GOSUB 3430
2150 PRINT "THANK YOU, " NAME$:G
OSUB 3430
2160 PRINT "IN OUR LAST EPISODE
YOU PICKED AWRITING TOPIC. PLEAS
E LIST THAT TOPIC:"
2170 PRINT:INPUT T$:CLS(5)
2180 PRINT "THANKS AGAIN.":PRINT
: PRINT"PLEASE MAKE CERTAIN YOUR
 PRINTER IS TURNED ON.":PRINT
2190 PRINT "PRESS <ENTER> TO PRO
```

```
CEED.":INPUT G$:CLS(8)
2200 PRINT: PRINT "OK, " NAME$
2210 PRINT "PLEASE TELL ME A LIT
TLE ABOUT"
2220 PRINT T$
2230 GOSUB 3430
2240 PRINT "PLEASE TYPE IN SOME
PHRASES      ABOUT IT.":PRINT
2250 PRINT "(P.S. DON'T USE PUNC
TUATION      MARKS. I CAN'T DIGES
T THEM.)":PRINT
2260 PRINT "(PRESS <ENTER> WHEN
FINISHED.)
2270 PRINT:INPUT INFO$
2280 PRINT #-2, CHR$(31); "SOME
WRITING IDEAS DEVELOPED BY ":PRI
NT #-2,"    " NAME$: PRINT #-2
2290 PRINT #-2, "THE TOPIC IS: "
 :PRINT #-2,"    "T$: PRINT #-2
2300 PRINT #-2, "THE GENERAL IDE
A OF THE PAPER WILL BE: ":PRINT
#-2, "    " INFO$:PRINT #-2
2310 GOSUB 3430:CLS(2)
2320 PRINT: PRINT "ONE OF THE MO
ST IMPORTANT THINGSABOUT WRITING
 IS TO BE SPECIFIC,TO USE LOTS O
F EXACT DETAILS.":GOSUB 3430
2330 PRINT "LIST SOME DETAILS AB
OUT YOUR SUBJECT.": GOSUB 3430:C
LS(2)
2340 PRINT "FOR EXAMPLE, LIST SO
ME OF THE   'SIGHTS' ABOUT IT. W
HAT SHOULD  YOUR READER SEE?"
2350 PRINT: PRINT "LIST A SIGHT
OR 'VISUAL DETAIL.'":PRINT
2360 INPUT S$
2370 PRINT #-2, "SOME DETAILS AB
OUT THE TOPIC:":PRINT #-2
2380 PRINT #-2, "    ***SIGHTS:":
 PRINT #-2
2390 GOSUB 3450
2400 PRINT: PRINT "GOOD BEGINNIN
G": GOSUB 3430: CLS(2)
2410 PRINT "NOW LIST ANOTHER SIG
HT": PRINT
2420 PRINT "(OR PRESS <XXX ENTER
>IF YOU'VE  FINISHED)"
2430 INPUT S$
2440 IF S$="XXX" GOTO 2460 ELSE
GOSUB 3450
2450 CLS(2):GOTO 2410
2460 CLS(2):GOSUB 3430:PRINT "NO
W LIST SOME SOUNDS OR NOISES:"
2470 PRINT #-2, "    ***SOUNDS":P
RINT #-2
2480 PRINT:PRINT "PRESS <ENTER>
AFTER EACH SOUND.":PRINT
2490 PRINT "PRESS <XXX ENTER> WH
EN FINISHED.":PRINT
2500 INPUT S$
```

```
2510 IF S$= "XXX" THEN 2540
2520 GOSUB 3450
2530 CLS(2):GOTO 2480
2540 CLS(3):PRINT
2550 PRINT "THINK OF OTHER DETAI
LS USING    THE SENSES.": GOSUB
3440
2560 PRINT "  SMELLS?    TASTES?
  FEELING?"
2570 PRINT #-2, "    ***OTHER SEN
SES":PRINT #-2
2580 PRINT "LIST SOME SENSORY DE
TAILS.":PRINT
2590 PRINT "PRESS <ENTER> AFTER
EACH DETAIL.":PRINT
2600 PRINT "PRESS <XXX ENTER> WH
EN FINISHED": PRINT
2610 INPUT S$
2620 IF S$="XXX" GOTO 2650
2630 GOSUB 3450
2640 CLS (3):GOTO 2580
2650 CLS(4): PRINT
2660 PRINT "THINK ABOUT ACTIONS
(OR VERBS IFYOU TALK 'GRAMMAR').
":PRINT
2670 PRINT #-2, "    ***ACTIONS":
PRINT #-2
2680 PRINT "LIST SOME ACTIONS FO
R YOUR WRIT-ING IDEA.":PRINT
2690 PRINT "PRESS <ENTER> AFTER
EACH ONE.": PRINT
2700 PRINT "OR <XXX ENTER> TO MO
VE ON.":PRINT
2710 INPUT S$
2720 IF S$="XXX" THEN 2750
2730 GOSUB 3450
2740 CLS(4):GOTO 2680
2750 CLS(5):PRINT
2760 PRINT "AND NOW LIST PEOPLE.
":GOSUB 3430
2770 PRINT #-2, "    ***PEOPLE":
PRINT #-2
2780 PRINT "WHO ARE THE MAIN 'CH
ARACTERS'    IN THIS 'DRAMA'?":PR
INT
2790 PRINT "PRESS <ENTER> AFTER
EACH NAME    OR <XXX ENTER> TO MO
VE ALONG.":PRINT
2800 INPUT S$
2810 IF S$="XXX" THEN 2840
2820 GOSUB 3450
2830 CLS(5):GOTO 2790
2840 CLS(6):PRINT
2850 PRINT "EVERY COMPOSITION  H
AS AN        AUDIENCE--":GOSUB 34
30
2860 PRINT "    THE PEOPLE WHO RE
AD IT.": GOSUB 3430
2870 PRINT "THINK OF THE AUDIENC
E FOR THIS  PAPER."
2880 FOR T=1 TO 1500:NEXT T:CLS(
6)
2890 PRINT "WHO WILL READ IT?":G
OSUB 3430
2900 PRINT "      FRIENDS?":GOSUB
 3440
2910 PRINT "            FAMILY?":G
OSUB 3440
2920 PRINT "                  CLASS
MATES?":GOSUB 3440
2930 PRINT "
TEACHER?":GOSUB 3430
2940 PRINT "DESCRIBE YOUR AUDIEN
CE":PRINT: PRINT"PRESS <ENTER> W
HEN FINISHED."
2950 INPUT A$
2960 PRINT #-2, "THE AUDIENCE FO
R THIS PAPER IS:":PRINT #-2,"
" A$: PRINT #-2
2970 CLS(6)
2980 PRINT "IN A FEW WORDS, TELL
 WHAT YOU'D LIKE THEM TO LEARN O
R UNDERSTANDFROM YOUR WRITING.":
PRINT
2990 PRINT "PRESS <ENTER> WHEN F
INISHED."
3000 INPUT U$
3010 PRINT #-2, NAME$ " WOULD LI
KE THE AUDIENCE TO LEARN OR UNDE
```

```
RSTAND:": PRINT #-2, "    "U$:PRI
NT #-2
3020 CLS(7): PRINT
3030 PRINT "IMAGINE YOU WERE TAL
KING TO     SOMEONE IN THAT AUDI
ENCE.":GOSUB 3430
3040 PRINT "YOU ARE GOING TO TEL
L THAT PER- SON YOUR STORY.":GOS
UB 3430
3050 PRINT "HOW WOULD YOU BEGIN?
":GOSUB 3430
3060 PRINT "TYPE IN YOUR OPENING
 WORDS.":PRINT
3070 PRINT "THEN PRESS <ENTER>"
3080 INPUT W$
3090 PRINT #-2, "A GOOD BEGINNIN
G MIGHT BE: ":PRINT #-2,"    " W$
:PRINT #-2
3100 CLS(7): PRINT
3110 PRINT"WHAT NEXT?": GOSUB 34
30
3120 PRINT "WHAT WOULD YOU TELL
THEM NEXT?": GOSUB 3430
3130 PRINT "TYPE IT IN. PRESS <E
NTER> WHEN DONE.":PRINT
3140 INPUT M$
3150 PRINT #-2, "NEXT WOULD COME
: ": PRINT #-2, "    "M$: PRINT #
-2
3160 CLS(7):PRINT
3170 PRINT "AND WHERE WOULD YOU
END?": GOSUB 3430
3180 PRINT "WRITE DOWN HOW YOUR
STORY ENDS": PRINT: PRINT "PRESS
 <ENTER> WHEN FINISHED."
3190 INPUT E$
3200 PRINT #-2, "AND IT MIGHT EN
D LIKE THIS: ": PRINT #-2, "    "
E$
3210 CLS(8):PRINT
3220 PRINT "OK, " NAME$
3230 PRINT "YOU'VE TOLD ME A LOT
 ABOUT:":PRINT "    "T$
3240 GOSUB 3430
3250 PRINT "YOU'VE LISTED SPECIF
IC DETAILS:":GOSUB 3430
3260 PRINT "  SIGHTS
   SOUNDS":GOSUB 3430
3270 PRINT "  ACTIONS
   PEOPLE": GOSUB 3430
3280 PRINT "YOU'VE DESCRIBED YOU
R READERS.": GOSUB 3430
3290 PRINT "AND YOU'VE TOLD ME A
BOUT YOUR   BEGINNING, MIDDLE, A
ND ENDING."
3300 GOSUB 3430
3310 PRINT "PRESS <ENTER>.":INPU
T G$:CLS(2)
3320 PRINT @ 224, "* * * IT'S TI
ME TO WRITE! * * *"
3330 FOR T = 1 TO 2000: NEXT T
3340 CLS (2):PRINT
3350 PRINT "USE YOUR PRINTOUT AS
 A GUIDE.": GOSUB 3430
3360 PRINT "YOU'VE PLANNED YOUR
WHOLE PAPER":GOSUB 3430
3370 PRINT "IF YOU ARE USING A W
ORD PROCESS-OR,PLUG IN THE CARTR
IDGE AND BE-GIN DRAFTING.":GOSUB
 3430
3380 PRINT "OTHERWISE, USE THAT
GOOD OLD    WORD PROCESSOR:":GOS
UB 3430
3390 PRINT "          PENCIL & PA
PER."
3400 GOSUB 3430
3410 PRINT "**********GOOD LUCK
**********"
3420 GOTO 3420
3430 FOR T = 1 TO 1000:NEXT T:PR
INT:RETURN
3440 FOR T=1 TO 500:NEXT T:RETUR
N
3450 PRINT #-2, "       " S$:PRIN
T #-2:RETURN
```

This Quiz Could Be Solution To Early Math Problems



By Bruce K. Bell, O.D.

Seeing my six-year-old daughter's performance drop in first grade math brought to mind nightmarish thoughts of grueling over flash cards and parent imposed homework. But then as I searched for a solution, I recalled how her petite face would light up when I would allow her to interrupt me typing in line 6099 of *Space Ghouls* just to type in her name and see it on the TV screen. Why not a computer based math drill program?! And so . . . *Color Math Quiz 1.0!*

Color Math Quiz is designed to be full of color (the eye's delight) and simple enough to be operated by pre-school children; yet includes levels to challenge junior high students. Although designed for run in Extended Color BASIC, it can be easily modified for Color BASIC. The program uses less than 2K memory, so will easily run on a 4K computer.

The Run

1st Screen: Here the upper limits of the operands are requested. For young children, this is entered by the supervising adult. If zero (0) is entered for either or both of the operands, then decimals less than one (1) are used. This screen is viewed only after *RUN*.

2nd Screen: This is the operations menu. Five menu items are presented; addition, subtraction, multiplication, division and fractions, respectively. To the left of the first item appears a reversed question mark (?). If addition is desired, then the Y key is pressed. Otherwise, the N key is pressed. The program uses *INKEY$* so ENTER is not necessary. Additionally, the program is key specific and will accept no other responses. After the response, the question mark is displayed to the left of the second menu item. As many, or as few of the operations may be chosen as desired by simply pressing Y or N by the appropriate item. After entering the fifth menu choice, the third screen appears automatically.

3rd Screen: At the top of the screen, the operation choices are displayed. Below this, the first problem is given. If the answer given is correct, then a colorful banner is displayed. Otherwise, the correct answer is given. After six problems "charge" is played and the number correct is shown at the top of the screen. After a few seconds, the menu is redisplayed.

END: To end the program merely press that little red key—the BREAK key.

Fractions: If fractions are chosen, then the operands will be fractions less than one (not decimals unless the upper limit is zero) with the upper limits of each numerator and denominator as set in the first screen. If fractions are not chosen, then integers are used as previously discussed.

To input an answer to a fraction problem, type the numerator followed by a comma (,) and the denominator, then ENTER it.

COLOR BASIC MODIFICATIONS: For using in Color BASIC, simply omit lines 3, 8, 115 and 305. Change the phrase "FNR(X)" in lines 255 and 400 to INT(X+.005)*100)/100 where X is the appropriate variable. For example the first command in line 255 would be changed to R+INT((R+.005)*100)/100.

INTERNAL FUNCTION

Lines	1—8	Define variables
Lines	10—60	Input operand limits
Lines	100—140	Input from Menu
Lines	150—310	Quiz is performed
Lines	400—430	Build fractions
Lines	1000—3000	Mathematical operations performed
Lines	9000—9030	Subroutine for color banners
Line	9050	INKEY$ subroutine

Color Math Quiz incorporates several programming functions to make it user friendly. Line 120 uses *POKEs* to the video memory located from 1024 to 1536, each corresponding to a *PRINT*@ location (0—511). The *POKE* method is useful to obtain characters not available by CHR$ or when a cursor no-update is desired. To see the full array of characters available by this method the following program may be used:

```
10 CLS
```

(Bruce K. Bell, a practicing optometrist, has developed a computer program for analysis of optometric data and uses it in his practice. He has written and lectured on vision and it's role in learning disabilities.)

```
20 FOR X=0 TO 255: POKE 1024 +X,X: NEXT
30 GO TO 30
```

By making menu selections in this manner, any combination of operations can be chosen.

The *DEF FN* and *FN* commands are useful when a single variable mathematical operation occurs several times during the program. Each time X=FNR(X) is encountered, X (or whatever the current variable) is rounded to two decimal places by the formula given in line 3.

The operands and operators are randomly selected in line 200. If fractions are to be performed, then the random seleetion of numerators and denominators occurs in line 400.

Line 2000 tests to see if the difference is less than zero, while line 4000 tests to see if the quotient is a decimal number. If neither is true, then randomizing reoccurs. The exception is during fractions where negative numbers are allowed.

The color banners are created in lines 9000 and 9010. The Boolean operation AND is performed on X to keep its value between ZERO (0) and seven (7). AND compares bit by bit the value of X and 7. If both are set (equal to one) then the resultant is set. For example, (5AND7)=5 because

```
      0101 = binary 5
AND   0111 = binary 7
      0101 = binary 5
```

On the other hand (8AND7)=0 because

```
      1000 = binary 8
AND   0111 = binary 7
      0000 = binary 0
```

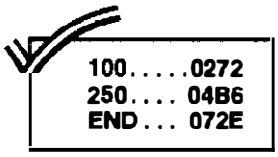

To see the values C, in line 9000, takes on, type the following program.

```
10 CLS
20 FOR X=1 TO 32: C=(XAND7)
30 PRINT X": "C, : NEXT
40 GO TO 40
```

Variables		
	H1$, H2$, H3$	Title/Credits
	0(X), 0$(X)	Arithmetic operators
	X,K,Q$,S,T,O	Work variables
	CH$	Charge chorus
	M$	Menu heading
	A,B,AR,BR	Operands
	AN,BN,AD,BD	Fraction operands
	R,RN,RD,C	Responses, eorrect answer
	NC	Number correct

Possible Modifications

As written, *Color Math Quiz* reinitializes the number correct each time the menu is encountered. For cumulative scores, delete the statement NC=0 in line 100. Also you may desire to establish lower limits as well as upper limits. Merely add the desired lower limit to the resultant of the random operand selection in line 200. By removing the randomizing function of BR in line 200 and letting B=BR you can now do, for example, multiplication tables for the sixes (6s).



The listing:

```
0 'COLOR MATH QUIZ MAY BE FREELY
 COPIED AND MODIFIED FOR PERSONA
L USE
1 H1$="COLOR MATH QUIZ 1.0": H2$
="COPYRIGHT (C) 1983 BRUCE BELL"
: H3$="EXTENDED COLOR BASIC 16K"
2 DIM O(5),O$(5)
3 DEF FNR(X)=INT((X+.005)*100)/1
00
5 H1$="COLOR MATH QUIZ 1.0": H2$
="COPYRIGHT (C) 1983 BRUCE BELL"
: H3$="EXTENDED COLOR BASIC 16K"
7 O$(1)="+": O$(2)="-": O$(3)="X
": O$(4)="/": O$(5)=","
8 CH$="T7;O3;L5;G;B;O4;C;D;P10;O
3;B;O4;L2;D"
10 CLS: PRINT H1$,H2$,H3$: PRINT
50 PRINT: PRINT"ENTER THE UPPER
INTEGER LIMITS  FOR THE OPERANDS
."
60 PRINT: INPUT"FIRST LIMIT";AR:
 INPUT"SECOND LIMIT";BR
100 NC=0: CLS: PRINT H1$,H2$,H3$
,,,: K=0: M$=" OPERATION MENU ":
 GOSUB9000: PRINT
110 PRINTTAB(7)"1. ADDITION (+)"
 TAB(39)"2. SUBTRACTION (-)" TAB
(39)"3. MULTIPLICATION (X)" TAB(
39)"4. DIVISION (/)" TAB(39)"5.
FRACTIONS (,)": PRINT
```

```
115 SCREEN0,1
120 FOR X=1TO5: POKE1253+X*32,63
130 GOSUB9050: IF Q$="Y" THEN O(
X)=1: POKE1253+X*32,25 ELSE IF Q
$="N" THEN O(X)=0: POKE1253+X*32
,14 ELSE130
140 NEXTX
150 M$="": FOR X=1TO5: IF O(X)=0
 THEN NEXTX ELSE M$=M$+O$(X)+" "
: NEXTX
160 S=(16-LEN(M$))/2: M$=STRING$
(S," ")+M$+STRING$(S," ")
170 K=0: CLS: PRINTTAB(7)"COLOR
MATH QUIZ 1.0": GOSUB9010: PRINT
190 FOR T=1TO6
200 A=RND(AR): B=RND(BR): O=RND(
4): IF O(O)=0 THEN200
205 A=FNR(A): B=FNR(B)
210 IF O(5)=1 THEN400
220 ON O GOTO1000,2000,3000,4000
240 IF O(5)=1 THEN255
250 PRINT A;O$(O);B"=";: INPUTR
255 R=FNR(R): C=FNR(C)
260 IF R<>C THEN PRINT "NO,"C"IS
 CORRECT" ELSE PRINT"RIGHT!";: F
ORX=0TO25: PRINT CHR$(143+16*(X
AND 7));: NEXTX: NC=NC+1
270 NEXTT
300 PRINT@7,"YOU GOT"NC"CORRECT!
"
305 SCREEN0,1: PLAY CH$
310 FOR X=1TO5*490: NEXTX: GOTO1
00
400 AD=RND(AR): BD=RND(BR): AD=F
NR(AD): BD=FNR(BD)
410 PRINT A"/"AD;O$(O);B"/"BD"="
;
420 INPUT RN,RD: IF RD=0 THEN420
425 R=RN/RD: A=A/AD: B=B/BD
430 ON O GOTO1000,2010,3000,4010
1000 C=A+B: GOTO240
2000 IF A-B<0 THEN200
2010 C=A-B: GOTO240
3000 C=A*B: GOTO240
4000 IF A/B<>INT(A/B) THEN200
4010 C=A/B: GOTO240
9000 FORX=1TO32: C=(X AND7): PRI
NT CHR$(143+16*C);: NEXT: PRINT
9010 FOR X=1TO8: C=(X AND7): PRI
NT CHR$(143+16*C);: NEXT: ON K G
OTO9030
9020 PRINT M$;: K=1: GOTO9010
9030 RETURN
9050 Q$=INKEY$: Q=VAL(Q$): IF Q$
="" THEN 9050 ELSE RETURN
```

Eeeeeee Yah! *Ninja Warrior* Is Here

I've always had a secret desire to be a martial arts expert. I even took a couple of Judo lessons once, but never found the time or the energy to finish them. I guess I'm really a lover, not a fighter.

But now! I'm a *Ninja Warrior*. A *Ninja* warrior is a deadly kicking, high jumping, *Sai* (pronounced "sigh" and looks like a trident, a three-pronged spear) carrying martial arts expert, who destroys boulders with a kick, catches fiery meteors on the Sai, jumps over deep chasms, and fights enemy Ninja.

As you've probably guessed, *Ninja Warrior* is a new arcade-style game for the CoCo and the TDP-100. Charles Forsythe, a 17-year-old M.I.T. student is the author of this machine language program from the Programmer's Guild. It is designed to run on any 16K machine and comes on a protected cassette (auto-load and *EXECute*) in a handsomely decorated box and with one sheet of instructions. The game responds to either keyboard control or the left joystick. The scenario of the game is as follows: You're a Ninja Warrior running "along a dangerous mountain road" (though the mountain road is perfectly flat). You encounter obstacles along the way which must be dealt with. Boulders are everywhere and can be kicked to dust or jumped over. At higher levels, fire appears in the road, too, but can be jumped. At even higher levels, pits open at your feet and you have to jump for your life. Also in the higher levels, fiery meteors appear falling from the sky and, if you're agile enough, you can catch them on your Sai.

As you progress, things get more complicated as enemy Ninja appear and have to be destroyed, boulders begin appearing stacked two high, and in the upper levels, the boulders start shooting arrows. At the highest level, when you become a grandmaster, you get to confront the Ultimate Ninja! (No, I haven't had the pleasure yet, but I can't wait.)

Actually there are 16 levels or belts (like brown belt, black belt, etc.) of play, each getting more difficult as more and more of the obstacles begin to appear. Each level seems to add a new type of obstacle, not all of which are documented.

You initially have three Ninja Warriors and get an extra one for each 10,000 points. I don't know how many you can have at one time, as I've not been able to keep more than three and the instructions don't even mention extra warriors. The game can have one to six players and the top three scores are displayed on the rather colorful title screen (generated by the auto-loader).

The Ninja is controlled by either the keyboard or left joystick (the instructions failed to mention it was the left one). The keyboard control uses the spacebar to kick, the up-arrow key to jump and the right-arrow key to "speed up" the Ninja. The joystick uses the fire button to kick. The jump is achieved by pushing the joystick forward and, for "speed up," you push the joystick to the right. I recommend the use of a center return-type joystick because if the stick isn't returned to center after a jump or speed up, you can't kick. I don't have a center return-type and found it very difficult to keep it centered myself, so I used the keyboard control and found it to be very good.

The graphics are well done, if not overwhelming, and the sound is good, though the running sound starts to grate a bit after a while. The illusion of motion is produced by having the Ninja "running" in place at the left side of the screen and the obstacles and the sky move from right to left; the effect is pretty good. The speed-up is just moving the Ninja toward the right edge of the screen, giving you less time to decide what to do for each newly appearing obstacle. This speed-up option adds a third dimension of control which really strains the old hand-eye coordination. After playing this game for a couple of hours now, I still haven't mastered it (but coordination was never a strong point for me).

My two resident arcade players, Cheryl, age 12, and Tim, age nine, initially were not impressed with the game, probably because of the high level of coordination necessary to play. I told them to stick with it for a while and, now, I need a reservation to get at my CoCo.

On a slightly sour note, I found the instructions to be confusing, misleading and not very complete. Fortunately, the game itself is simple enough not to require a lot of information to play. My adventurous son, who never reads the instructions anyway, tried all sorts of things, some of which worked in spite of the instructions telling me otherwise or not mentioning them at all.

So, if you like to be challenged, give yourself a treat and become a Ninja Warrior.

(The Programmer's Guild, P.O. Box 66, Peterborough, NH 03458, $29.95 on cassette)

—C.L. Pilipauskas

Logo, Pilot And Computer Literacy

By Larry Kheriaty

Probably the most common justification for buying a Color Computer is education, whether it be at home or in the classroom. Most of the people I know who have bought a personal computer say that at least one of their reasons is to allow themselves or their family members to learn a little about how computers work. In my town there are some excellent computer literacy courses based on the Color Computer being taught at schools ranging from Assumption Grade School to Western Washington University. I am also aware of several businesses or schools offering very poor instruction in computer literacy. What distinguishes a good course from a bad one? To answer that question we need to talk first about what we mean by computer literacy.

"For older persons there is often the problem of an already acquired mistrust of computers or a feeling that they are too mathematical or too complicated to understand. The best way to solve this is, again, hands-on experience with a computer."

The goals of any computer literacy program are the same whether you are enrolled in a formal course taught by an instructor, or you are learning at home on your own. There are essentially three goals. First, and most important is to raise the student's comfort level with the computer. This is accomplished by hands-on experience designed to simply let the student see firsthand that nothing will explode if the wrong button is pushed. The younger the person, the easier it is to develop a relaxed familiarity with the computer, and there are fewer misconceptions, fears or anxieties about the computer. Television and video games have paved the way for this aspect. For older persons there is often the problem of an already acquired mistrust of computers or a feeling that they are too mathematical or too complicated to understand. The best way to solve this is, again, hands-on experience with a computer. If they start out on the right foot, people are usually pleasantly surprised at what they can do without much technical knowledge.

The second goal is to build realistic expectations about what a computer can do. This requires that you know, at least in general terms, how a computer does what it does. As an analogy, we are all literate enough about automobiles to avoid such mistakes as to expect one to fly, or to attempt to use a sports car when a dumptruck is called for. It isn't necessary to be a mechanic in order to use a car effectively, but most drivers understand the car well enough to know the difference between a flat tire and running out of gas. These examples are so obvious that they sound absurd. But time and time again, I have spoken to people who have bought a computer expecting it to do a job that it simply could never do. The choices they make are so far from realistic that it is clear they did not have enough basic understanding to do anything other than buy what the salesperson said would be good for them. Other common misconceptions are expecting the computer to be intelligent, or expecting information back out of the computer that is not first put into the computer. An example of this is the storekeeper who would like to be able to check with the computer to see the present state of his inventory at any time; while not realizing the amount of work necessary to keep that information up to date on the computer, and not realizing the amount of data storage consumed by such a system. To become literate about cars we ride in them, drive them, buy a car and cope with the problems that come up as we use it. To become computer literate you should run one, try various programs and cope with the problems that come up. In this way you see firsthand what the various parts of the computer do and how things are done on the computer.

The third concept of computer literacy is to understand what software is, and what its relationship to the computer is. In fact, what is learned here is that computer literacy is actually software literacy. The computer itself is inert without a program to direct it and tell it how to respond to the user. Traits that might be thought of as the computer's are really the program's. And the personality or "soul" of the computer can be changed at any time by running a different program. A good way to think of it is that the computer is only a delivery medium. When a program is run, there are two parties in the conversation; the person and the program.

(Larry Kheriaty is owner of Washington Computer Services in Bellingham, WA and is author of Radio Shack's Pilot, Logo, Game Writer *and several other Radio Shack company products.)*

When you talk to someone on a phone, you do not converse with the telephone itelf, it is only the medium by which you communicate with the other person. When you can think of running a program on a computer as similar to talking with a person on a phone, you are on the right track. Along with this view of software, it is necessary to understand how software tells the computer what to do. Just as it is not important to become a mechanic in order to drive a car, it is not important to become a programmer to run a computer. But it is necessary to be exposed to programming enough to see how things are done. In the sense that a computer thinks by running a program, you can use a computer best if you know how it thinks.

The concept of software is the onc that makes the computer so different from any other tool. It is also the concept which most often confuses the new user. To add to the confusion people are told that BASIC is about as easy as it gets. Then, when they find that it is a very slow and complicated process to write a BASIC program they can get discouraged. Since software is the most unfamiliar idea to the new user, it is the one that should be approached most carefully. Since it is not a goal of computer literacy to make programmers out of everyone, it is more important that the first contact with programming be truly easy, and be truly representative of what software is about. So when it comes down to it, the most important aspect of any computer literacy program is the selection of the software to be used for the person's first experience.

This is where *Logo* comes in. *Logo* is specifically designed to introduce people to programming. The results of a program are always a picture on the screen. This gives the student immediate visual feedback on what has happened. To the student, *Logo* appears as a "turtle" which lives on the display screen. At any time, the turtle is visible at a specific place on the screen, and is pointed in some direction. The user tells the turtle to walk forward so many steps, or turn right or left so many degrees. As the turtle moves, it leaves a colored line on the screen. The effect of seeing an immediate action take place on command gives the student a good feel for the concept of telling the computer exactly what to do. For example, the student would type a command like *FORWARD 50* and the turtle would immediately move 50 steps in the direction it is pointing. For children too young to read, or type, *Logo* is supplied with a keyboard overlay which labels some of the keys with pictures that represent turtle commands. Using the "doodle mode" the turtle can be directed by single keystrokes. Once the student is comfortable with giving one command at a time, she can combine several commands into a procedure, give the procedure a name, then tell the turtle to do the procedure. For example:

```
TO BOX
  FORWARD 50          or      TO BOX
  RIGHT 90                      REPEAT 4
  FORWARD 50                     (FD 50)
  RIGHT 90                       RT 90)
  FORWARD 50                  END
  RIGHT 90
  FORWARD 50
  RIGHT 90
END
```

The BOX procedure on the left tells the turtle how to draw a square on the screen. The BOX procedure on the right does the same thing, but uses abbreviations for the words FORWARD (FD) and RIGHT (RT). It also saves typing by using the REPEAT command which just says to do what is in the parentheses four times. A procedure always starts with the word TO followed by a name. It ends with the word END. Once the procedure BOX is typed into the computer's memory then the student can tell it "BOX," and the turtle will follow the procedure named BOX. The concept here is that by writing the procedure the student has told the turtle how "to box" when it is told to do so. If the student were to tell the turtle "FLY," then a message "I DON'T KNOW HOW TO FLY" would come back. This would mean that no procedure has been entered to give the instructions to be followed when told to fly. The student can enter any number of procedures; each one defines a new action that the turtle knows how to do. By actually doing it the student learns what programming is about, and since *LOGO* procedures are so easy to understand, the chances for a good experience are much greater than with something like BASIC. One of the main benefits of *LOGO* is that the student learns structured, logical thinking by placing commands in logical order. Because of the visual nature of *LOGO*, there seems to be a high degree of incentive on the part of the student to experiment with ideas. In fact, the basic premise is that *LOGO* is a tool for experimentation and creativity. It is very easy to master the commands needed to use *LOGO*, but they can be combined in an infinite variety of interesting ways. The concept of modular or "building block" logic is fostered since the student can use previously defined procedures to build new procedures. For example:

```
TO FAN
  REPEAT 20
    (BOX
     RIGHT 18)
END
```

FAN produces a beautiful circular fan of 20 boxes rotated around at 18 degree intervals. Notice that it uses the previously defined procedure BOX.

To enter the procedures into memory, the student uses a very simple screen oriented text editor which is built into *LOGO*. The student just types on the keyboard and the screen shows what the text looks like. The various arrow keys are used to move about in the text on the screen. The editor is designed to be a simple example of how a typical word processor works. In fact, it can be used to enter any text, which could be printed on a printer later. So in the proc , the student learns the underlying concepts of word processing, probably the most common business use of microcomputers.

Even though *LOGO* was designed for kids, it turns out to be the best place to start for a person at any age since it starts out simple and lays an appropriate foundation for further learning. In this short article we have only scratched the surface of *LOGO's* capabilities. For the more advanced person, the concepts of recursion and multi-tasking can be explored since *LOGO* allows as many turtles as desired to all be running their own programs simultaneously. But beyond computer literacy, *LOGO* is a great medium for "playing" with logical concepts that expand the child's ability to think logically. And all experience so far shows that kids consider the challenge fun. I don't know where the word *LOGO* came from, but if it is derived from the Greek word "logos" it lives up to its name, which in different contexts means "word," "idea," "concept," "discovery" or "thought."

A second very useful educational program for the Color Computer is Color *PILOT*. It is used in quite a different manner. With *LOGO*, the student learns by writing *LOGO* commands and programs. It is the experimentation with writing *LOGO* programs that provides the educational experience. *PILOT*, on the other hand, is used to write an educational program that will be run by the student. The writing of the *PILOT* program is like the preparation of a lesson. The student's learning experience comes from executing the lesson program. Also, with *LOGO* the subject matter to be learned is computer literacy and logical thinking, whereas *PILOT* is used in situations that have commonly been called "computer assisted instruction." As an example, a *PILOT* program could be written that would help the student study spelling. In this example, the author of the *PILOT* program would write the program so that the student is presented with problem words. The author is using *PILOT* to create some learning environment for the student. Students later run the lesson program and interact with it. In this case, the computer is a teaching medium for some other subject matter. The student is not necessarily learning about computers per se. *PILOT* has the same built-in text editor as *LOGO*, so the mechanics of entering a program are very simple. The *PILOT* language has commands that simplify the presentation of text and graphics. There are also simple commands to accept and judge student replies, then give appropriate feedback. In general, *PILOT* is much easier to learn for this type of programming than BASIC because *PILOT* is specifically designed for this application. The following *PILOT* statements present a question, accept a reply, then give appropriate feedback. It is, of course, too simple to be useful but it does show the essential elements of a question/answer frame.

```
T:What element is found in      (type a question)
T:all organic compounds?        . . .
A:                              (accept an answer)
M:carbon                        (match for correct reply)
TY:That is correct.             (correct reply feedback)
TN:No, the answer is carbon.    (incorrect reply feedback)
```

There are three contexts in which Color Computer *PILOT* would be useful. The first is in the preparation of educational software to be sold commercially. In this case the process is analogous to the writing of a textbook in that much work is spent on the original so that it can be published and used many times. The second environment for *PILOT* use is the classroom. The teacher uses *PILOT* to create a lesson, drill or lab exercise that complements the classroom material. This provides a way to give students extra practice where needed, supplementary material for advanced students or to demonstrate some point that can't be demonstrated by other means. The third context for *PILOT* is as a means of supplementing the student's school work at home. This could be done by the parent writing lessons, or by buying or checking out materials already written in *PILOT*. Even though *PILOT* would be easy enough for a child to learn, it would not normally be used as a language in which a child would learn to write programs. It is intended to be used to write programs that teach something, so to make effective use of *PILOT* one would have to know what is to be taught, and how to teach it. *PILOT* is simply the way to put the lesson into the computer so that it can be delivered later to the student.

LOGO and *PILOT* are both very useful educational programs that no Color Computer should be without. *LOGO* is used for computer literacy, and *PILOT* is used for computer assisted instruction.

Catacomb Is *Amaze*-ing Game

Oregon Color Computer Systems has come out with a new game, *Catacomb*. Just put in your cassette and *CLOADM*. The program does the rest. As a programmer, I wanted to see how the auto loader worked. When loaded into my *EDTASM+*, the source code didn't mean anything to me, and a couple of the commands even messed up the EDTASM+. Enough of this, let's get to the game.

You are trapped in a maze-like catacomb with the enemy patrolling all of the corridors. Although you have the energy to break out of the *Catacomb* when you start, it is recommended by this reviewer that you commandeer a full load of fuel to make it through the second screen. You must move through the *Catacomb* seeking fuel blocks until you feel you have enough. There are guards patrolling the *Catacomb*, so watch out! There are two types of guards, one type looking sort of like an "@" (at sign) and the other resembling swooping eagles. The "@" guards come through either the left or right walls and travel the horizontal passages. The eagle guards come in at either the top or the bottom of the screen and patrol the vertical passages.

When you turn on the computer, the color the game will be is already decided. It will either be a red maze with you blue or a blue maze with you red, the monsters are always white. If you are like me and are picky about what is what when it comes to color, this little routine in BASIC should clear up the problem. Just type it in when you power up. If the screen is red, then the maze will be red. If the screen is blue, the maze will be blue. Remember, type this in before loading the program.

```
10 POKE &HFF22,&HF8
20 POKE &HFFC5,&HF8
30 POKE &HFFC1,&HF8
40 FOR I=&H400 TO &H1BFF
50 POKE I,&HAA
60 NEXT I
```

You should see a screen with many red, white, and blue blips on a black background. The screen should turn red from the top to the bottom (or blue). When the program finishes, type *NEW:CLOADM*. *Catacomb* should now load in. If the color isn't what you want it to be hit reset and then *RUN* the program again before you erase it. Because of the nature of the graphics screen you may have to reset the computer more than once. By the way, the BASIC program will work in Extended or Disk Extended BASIC. It might work in Color BASIC, but I don't know. *Catacomb* runs in all the BASICs.

Back to the program. To get to the second screen, you push the fire button. The second screen is the hyper-space corridor. You must maneuver around the enemy vessels which fly straight down, and avoid the space mines which fly straight down until they detect you, at which point they crisscross the screen. You can fire now (go die!) and your mission is to fly to the other end of the corridor, through the starport, and to the mother ship. A full load of fuel will guarantee that you have enough to reach the mother ship, but there are energy blocks along the way that you can try to pick up if you don't have a full load.

Pulling back on the joystick during flight will increase speed; pushing forward will slow it down. I can't describe the mother ship or the stargate, because I haven't made it there yet, but the screen gets rather full with perhaps 20 enemy vessels and 15 space mines crisscrossing in front of, behind and all around you.

On the first screen, you receive 25 points for each fuel block you steal, and 1,000 points for having a full load. On the second screen you get 35 points for each enemy vessel you destroy, 50 points for each space mine, and 1,000 points for reaching the mother ship.

This is a great program with very good graphics.

If you like arcade games, purchase this one—it's good!

(Oregon Color Computer Systems, P.O. Box 11468, Eugene, OR 97440, $19.95 disk or cassette)

—Scott Sehlhorst

Pick Which: Original, But Lacks Playability

Many of the popular CoCo games today have their roots in the arcades. Others are adaptations of different types of pastimes, such as sporting and gambling games. Coming up with an original game that is also a fun one can be a very taxing chore.

Pick Which is a 16K machine language game that comes as close to original as any game that I have ever seen. It features good graphics and sounds, and can be played by up to four people at once.

Playing *Pick Which* is about as easy as you can get. After you select the desired number of players, you are ready to begin. A big picture appears on the screen, and there are 16 smaller, but not visible, pictures behind it (so to speak). By moving your joystick around, each of these smaller pictures can be made to appear one at a time. After a short period of time, the picture you are viewing at that moment begins to enlarge until it fills the screen. Depending on which picture you wind up with, several things can happen. Sometimes you will gain points, and other times you will lose points. You could also wind up on a picture that will have the program pick the next one for you at random. One of the pictures will reveal the location of all of the others, while another will give you a short rest. Finally, the picture of the "stop sign" will end the game.

Pick Which is an interesting game, but I am unsure as to who it is best suited for. It is too fast paced for very young children, and will probably not hold the interest of older children or adults. I guess the six to 10 year old range would be about right. If you have children in this age range, and would like to try something out of the ordinary, you might want to consider *Pick Which*.

(Spectral Associates, 141 Harvard Ave., Tacoma, WA 98466, $9.95 tape, $13.95 disk)

—Gerry Schechter

Software Review

Bugger's Talking Intro Not Matched By Graphics

All heads turned toward the computer in our house when, upon loading *Bugger*, a tiny voice originating from my TV monitor cried out for help for a bug trapped in a web with six spiders.

"Wow, what a game this is going to be," was the thought that raced through my mind as we waited for the program to unfold.

What followed was not to match the excitement I originally felt, although the game was challenging. It's another "Pac-type" game with our hero trapped in a maze and defying the odds against some lightning-quick web-weavers.

The object, of course, is to escape the complex alive, which I found to be almost impossible since the spiders' movements are so unpredictable. While you're improving your skills you are able to eat tidbits of food that have been left lying around. The number of tidbits equals your score for a round. Until you're finally able to escape, you'll have to be content with increasing your intake of these morsels.

Maneuverability is managed through use of the arrow keys, which after you have become accustomed to them, prove to be fairly easy to use. I had to unplug my left joystick, however, to get them to function correctly.

There are three levels of play and after each round you are given the choice of a new skill level. I found the game flowed more smoothly by just pressing the ENTER key each time.

Being an old hand at games like *Pacman* and *Doodlebug*, I was not that impressed by *Bugger's* graphics. However, I'll have to admit that the game is challenging and quite contagious. It's a nice game if you're watching your budget.

(Mr. R's Software, 68 Kelly Road, So. Windsor, CT 06074, $14.95 on tape)

—Charles Springer

Ambushed By *El Bandito* (And Enjoying It)

Arcade games come in several fairly standard models. The primary requirements to make a new entry marketable are that it is a new game, there is a new twist or there are some nice bells and whistles. *El Bandito*, written by David Crandall and distributed by Mark Data Products, is a game that manages to qualify on all three counts. It runs on a 16K CoCo and is played with joysticks.

First, I want to tell you what it is not. It is not, as the name implies, a shoot 'em type arcade game. It also is not, as some may assume, a trade on anything ethnic.

It is a fast action, maze/chase type arcade game for one or two players. The "Bandito" referred to in the title is your, or your opponent's, picnic ant, which must steal as much food as possible hidden in the maze, and return it to a home lair. There are obstacles, of course.

At the start of each game you have three "ant lives." You select a skill level, and a maze difficulty level, each rated from 1 to 8.

The skill level is easiest to explain. Each level represents the number of spiders which can do in your ant. The spiders randomly (and rapidly) move about the maze; and if they contact your ant, that's the end of the ant life.

The maze remains basically the same, except each higher maze level enables more side passages and escape tunnels. Your ant may use these, but the spiders may not. That sounds good until your ant goes in an escape tunnel, and materializes in a room full of spiders!

Points are awarded for various fruit returned to the lair, only five points for a banana, but a maximum of 100 for an apple. The smallest point award, but maybe most important is only one point, for an ant egg. With that one point, though, comes a bonus life. At the higher skill levels you'll be desperate for those extra lives.

Some things which I did not care for, were that the joysticks respond a bit spongy. After a bit of play, though, it seems almost natural and ant-like for the response to be a little helter-skelter. Also, a score continuation feature for beating the maze would have been nice, but was not included. Finally, you have to shut the computer off to remove the program. (Piracy made that necessary; it's costing us all.)

Some nice features are the ability to freeze and restart play, halt play and start a new game, a continuous "HIGH SCORE" display, and the ability to easily shift from one to two player mode. You can mix skill and maze levels for almost any degree of difficulty you choose. Perhaps the neatest features are (1) simultaneous (non-rotational) play of opponents, and (2) the ant moves slower when carrying food, but may drop it to escape the spiders. With these two, your ant may carry an apple nearly home, only to have to drop it to escape a spider. Then, your opponent's ant picks it up at his door step for an easy 100 points. Fun? You bet. Mark Data has another winner with *El Bandito*.

(Mark Data Products, 24001 Alicia Pkwy., No. 207, Mission Viejo, CA 92691, $24.95 on tape, $29.95 on disk)

—Gary Smith

Old McDonald's Farm Vowels
A—E—I—O—U

Old McDonald's Farm Vowels is an educational drill program for beginning readers in kindergarden through the second grade. The goal of the program is to teach vowel sounds to the young reader. The program is written in machine language and requires 16K ECB.

As soon as I began reading the instruction manual, it was obvious that this was a well-documented program. Much effort was put into the development and writing of the manual. After reading the instructions only once, I thoroughly understood what the program was supposed to do and how to use it. (If only the instructions for putting Christmas toys together were half as good.) The author uses detailed examples and portions of actual screen displays to describe the program in the 11-page manual.

Old McDonald's Farm Vowels does exactly what it claims to do and that is to drill the student over the long and short vowels. Initially, each vowel is pronounced in its long and short form and then used in a word. Next, the word is used in a sentence to provide the proper context. After this short tutorial, the drill practice actually starts. The student is presented a graphic display of a farm animal such as a dog, goat or pig. The tape player is then turned on automatically and a voice artieulately pronounces the name of the animal and asks the student to indicate whether the vowel was used in its long or short form. To respond, the student only has to push either the right or left arrow key.

Correct responses are rewarded by a smiling face and "Old McDonald's Farm" is heard. If a response is incorrect, the student is given a clue and prompted to try again. Once the correct response is given a smiling face is seen. Negative reinforcement is never used. Scoring is kept at the bottom of the screen. The scoring format is handled nicely and allows the parent or teacher to easily determine which items were missed on the initial try.

Old McDonald's Farm Vowels has excellent graphics and the author used good creativity in designing the format of presentation. There are several bells and whistles that make the program interesting, and each one adds to the educational value of the program. They are:

1) The score is constantly displayed on the screen so that the student can see his progress.
2) A surprise graphic is displayed at the end of the exercise if the student gets all items correct.
3) The exercise can be made more or less difficult by changing the allowed response time.
4) The scoring display provides a record of correct and incorrect responses.
5) The correct arrow key responses are randomized to help eliminate positional memorization of inputs.

If you are looking for a good educational program for your child, I think this would make an excellent collection for your library.

(Teksym Corp., 14504 County Road 15, Minneapolis, MN 55441, $14.95 on tape)

—Michael K. Hunt

(continued from page 14)

One of the things which has confounded me for some time now is why we do not get more submissions from teachers, their students and from computer classes and/or clubs in schools.

In thinking about that, it seems to me that one of the reasons may stem from the same reason as why we do not get more submissions for non-Extended Color BASIC programs.

I have long believed the reason we do not get non-Extended programs is because most of the people who have non-Extended CoCos are beginners. And, seeing the many well-written and innovative programs we publish, they feel their own efforts might not be "up to snuff" for us to use.

By the time someone upgrades to Extended, he has some expertise and feels more comfortable making a submission. And so, many of the programs you see in *the Rainbow* are for Extended.

We were all beginners. And I want to say here, for the record, that we encourage the submission of non-Extended programs. Often, I find, beginners have some of the freshest ideas and do some of the most creative programming.

As to youngsters, many of them are beginners, too. But I am frankly surprised that we have not seen more submissions from them.

One of my long-time interests has been for a youngsters' section. We at *the Rainbow* are most interested in encouraging young people and their use of CoCo. Such a section could be devoted to their programs — and would be an interesting outlet for young people to share some of their programs, ideas and techniques.

We would like to have a section of this sort, but can do so only with your help. If you are a youngster, please send a program or two to our Submissions Editor. And, when you do, please let us know how old you are. OK?

And, for all the teachers and students who are reading this: Why not make it a project this fall to do something for submission to *the Rainbow*? No, we certainly cannot guarantee we will print everything — but if we get sufficient response we *can* institute a special feature each month — perhaps with photos of the class at work.

I want to report here a story which reaches me with mixed emotions.

No doubt you have seen me mention the name of Jon Shirley, the vice president for computer merchandising for Radio Shack, in this space before. Jon announced this month that he is leaving Radio Shack to accept a position with Microsoft, Inc. in Washington State.

All of us owe a debt of gratitude to Jon Shirley. It was under his administration that the Color Computer came to be and it has been through his work that CoCo has become a "word" in so many thousands of households. Thank you, Jon, from so many of us.

Next month will mark the beginning of our Second Annual Adventure Contest! The first contest was one of the highlights of the year for us and we look forward to another great one.

You will see the formal announcement next month, but it might be wise to start considering an entry now. I believe you will be impressed with the prize list.

And, speaking of contests, the Simulation Contest has some really excellent entries. We'll be naming the winners soon and printing their programs. So, stay tuned.

Judging from my mail, it appears that the record in the Anniversary issue was an unqualified success. With a few exceptions, everyone who wrote (and there were hundreds of you who took the time to do so) thought binding in a record was an excellent idea.

The proposal that we increase the price of *the Rainbow* to finance a record each month generated comments ranging from wild enthusiasm to concern for cost. And, the bottom line appears to be that we will not do it as a regular feature.

The reason is simply that the record adds something like 25 cents to the cost of every magazine, once the manufacture, binding, mailing and other associated costs are taken into account. That, coupled with two additional facts, makes it an unwise move on our part.

Fact One is that the cost to us I quoted above is based on what is, essentially, a three-program record. To do more would cost considerably more.

That, in and of itself, seems to be satisfactory to a majority of those who wrote. But Fact Two is simply that we are forced to raise our cover price next month, anyhow. Effective with the October issue, single copies of *the Rainbow* will cost $3.95. There is no change in the subscription price.

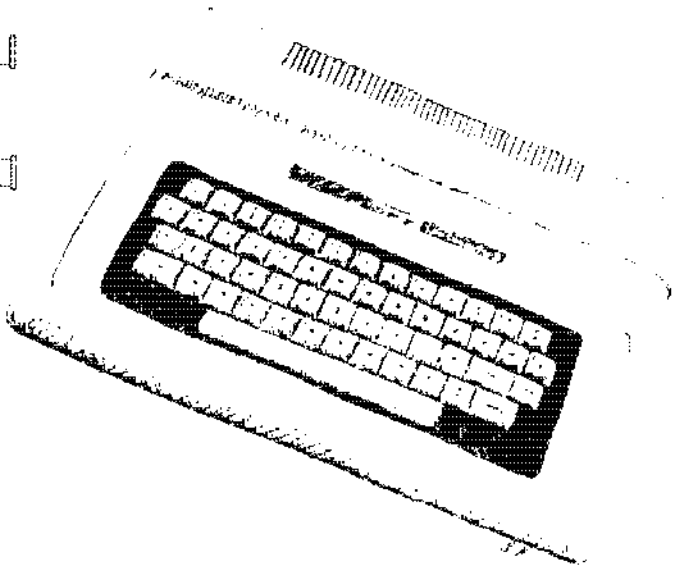
Why are we raising the cover price? First, because the cost of paper has gone up again, as has the cost of just about everything else that we do. The $2.95 cover price was set a year ago this month — when we had 106 pages. The number of pages has tripled since then.

The second reason is that shipping costs have increased as well. Part of this is due to rate increases. The other to the sheer size of *the Rainbow*. Last month's issue weighed in at over 20 ounces!

Will there be a subscription price increase? Yes, in time. But, before we do, we will let you know here — and give you to opportunity to extend your subscription at the present rate. For now, though, our increase in price affects only single copy sales.

—Lonnie Falk

PIPELINE



Not one, but two new Color Computers are in the fall lineup from Radio Shack.

The 64K Color Computer, (see cover) from an "inside" point of view, merely confirms what many of us have known for a long time—that just by making a few internal changes the "32K Color Computer" could become, in effect, a 64K machine.

However, the introduction of the 64K CoCo also coincides with the introduction of the OS-9 software package, providing a true disk operating system from Radio Shack for the first time. Yes, OS-9 has been rumored for many months, too. But, now, here it is.

And make no mistake about it—it's a full-fledged disk system based on the popular UNIX system (the Bell Telephone Laboratories operating system for large minicomputer systems). It comes with an editor, assembler and a debugger. With four hefty manuals to explain things.

This month's Pipeline is devoted to Radio Shack's two new Color Computers. An artists's sketch of the Color Computer 2 appears above.

Photos of the 64K CoCo and an in-depth view of the OS-9 appear on page 236.

Not completely expected is the second major Radio Shack CoCo—the Color Computer 2. It is smaller than the "traditional" CoCo, thanks, primarily, to some excellent design and engineering at Tandy Corp. And, with one single exception, it is electrically the same as its larger brothers.

What do these two new CoCos have in common?:

For one thing, their color. It isn't white, but fawn gray—an off-white if you will.

For another, both feature a new keyboard- with three-color sculptured keys which have more "travel" than the "old" CoCo keyboard.

And, both are, essentially, software compatible with every Color Computer which has ever been produced (TDP-100s, too).

But, much more than that, the new Color Computers position Radio Shack squarely in the middle of the personal computer market for some time to come. These are well-built machines, based on the state-of-the-art 6809 microprocessor from Motorola. Especially with their new colors they are attractive, durable and—because of the software capability—come to the marketplace with a vast library of programs which can be run today.

The 64K CoCo featured on this month's *Rainbow* cover is available now from your local Radio Shack. So, too, is the OS-9 software. The computer sells for $399 and the software for $69.95. (For more details on OS-9, see technical editor Dan Downard's article on the new 64K CoCo in this issue).

Color Computer 2 should be available within weeks. The non-Extended BASIC version will retail for $239.95. Extended BASIC goes for $319.95. And, yes, you can buy the keyboard separately—for $39.95 plus installation.

The 64K CoCo

There is really no internal change between the 64K Color Computer and the one which Radio Shack has been selling for some months. One of the worst-kept secrets for some time has been that the 32K Color Computer was equipped with 64K chips which, likeas not, with some little modification, could support a "bank-switched" system.

By "bank-switched" we mean that either half of the 64K of RAM could be accessed at a time. Programmers soon found ways to use the "other" half of the RAM for several different purposes—everything from storing variables to using that part of the memory as a disk-like storage area for whole programs.

Additionally, a number of language systems, with FLEX being the most popular, were able to use the whole 64K. OS-9 will operate in the same way.

This, incidentally, is no different than the "64K computer" advertised by other firms, such as Commodore. What the Commodore lacks, among other things, is a state-of-the-art microprocessor like the 6809. And a sophisticated operating system.

A look at the motherboard of the new 64K CoCo shows it to be, essentially, no different than the ones already for sale by Radio Shack. Some capacitors have not been installed and the jumpersare set to 64K at the factory. That is what we have been telling you for months you should do to "upgrade" your F-Board CoCo from 32K to 64K. Of course, the 64K chips are included—and certified 100 percent.

Electrically, this is the same computer with some cosmetic changes. But what changes!

The new color should make CoCo a much more welcome addition to the home. It blends in with most any decor. And, because the color is moulded into the plastic cabinet, you end up with a machine from which the color will not wear off. It should be easier to clean, too.

The keyboard is a great improvement. The keyboard colors—white, fawn gray and red—blend in well with the case itself. The keys are wider, flared out, and have better tops than the previous ones available from Radio Shack

They have more "travel" too. That is, they move further down when pushed. Though they stop somewhat short of

what a touch-typist would consider "full-travel," they are a vast improvement for one of the areas which spawned the most complaints about the original CoCo. Their color coordination gives them, and CoCo, a more complete look, too.

And, while the only other change is the location of the Radio Shack logo, the color somehow makes the new CoCo look more sleek. We spent several minutes comparing the "battleship gray" and fawn gray models before reaching the conclusion that there is no difference between the cases. Yet, the new one looks "better," somehow.

Inside, the only difference is the existence of a bronze clip which rests atop the RF shield and presses up against the bottom of the keyboard—which is covered with metal. We assume this is a grounding adjustment.

Color Computer 2

The first thing one would notice in looking at Color Computer 2 (CoCo 2) is that it is quite a bit smaller than its older brother. Here, Tandy engineers have performed some wonders in compressing everything (yes, everything) into a case which is about two-thirds the size of the original.

Again, CoCo 2 features the fawn gray—slightly slicker—case, sports the new keyboard and is, essentially, the same electrically as the 64K model.

The one difference, electrically, from earlier CoCos is that CoCo 2 does not have a 12-volt power line running to the expansion port. A few pieces of equipment which draw power from the 12-volt line (including Tandy's own X-Pad) will have to undergo modification to use CoCo 2.

While available as a 16K model for both non-Extended and Extended BASIC, CoCo 2 is upgradable all the way to 64K. At least, we understand, that is possible and it should be, because CoCo 2 has the same logical workings (and chips) as does every other Color Computer thus far sold.

It is almost worth breaking the factory seal and voiding your Radio Shack warranty to see how it was done. Some of the ideas are easy to spot—others buried in the well-designed motherboard. One of the most obvious is the turning of the RF modulator on its side. It now takes up less than a third of the space on the board.

CoCo 2 should be a big seller. It is nice looking, compact, and priced right. It is upgradable and, again, has a wealth of software available for it.

New ROMS

Both the 64K CoCo and the CoCo 2 feature the new ROMs, which have been a part of every Color Computer shipped in recent months. These ROMs fix a number of problems with the original version of the operating system, but may be a problem for some machine language programs which used what Radio Shack calls "undocumented" calls to the ROMs.

Most software manufacturers have been aware of the problem already, and will offer modifications to their programs when necessary. As a service to machine language programmers, Technical Editor Dan Downard, offers a rundown on the changes to the ROM.

Those lucky enough to have seen the new CoCo thus far have been favorably impressed. Radio Shack, we understand, is bullish on the two new models and more support is on the way.

With its sleek new styling, model alternatives and a full-blown and sophisticated operating system, we see the future as exceedingly bright for our favorite computer.

Fast Reflexes? You'll Like *Fast Lane*

When I first read the directions for *Fast Lane*, the idea of the game intrigued me. After playing the game for several hours with a few friends, we all decided it was rather hopeless.

Fast Lane is a machine language, arcade-type game, the idea of which is to steal weapons from the enemy. When the game begins, you find yourself on the right side of a screen full of space traffic. You must maneuver your spaceship, using the arrow keys, to the left side of the screen. The faster you accomplish this, the more fuel you will save, and the more points you will receive. Having done this, you now go to the second screen. Here you have entered the enemy spaceship which has electrified walls and moving guards. You enter on the right side of the screen and must get through the rooms via the doors, which open and close, to the weapon in the upper left corner. After you get the weapon you must return it to the lower right corner. Again, you must do all of this before your fuel runs out.

If you succeed in stealing the weapon your score so far, will be shown. You then push the spacebar to go to the second level which is supposedly more difficult. You can push the "P" key to pause your game any time during the play.

The biggest drawback to this game is the fact that it uses the arrow keys for movement. It would play much smoother and faster if it used the joysticks. The directions say, "you should only tap the arrow keys" to move your spaceship. However, even in tapping, (especially for right-left movement,) the keys are just too sensitive. You will usually end up smashing into a wall.

Fast Lane requires very fast reflexes, which I don't have, so I invited some friends to try their skills. Of four different people trying the game for a couple of hours, the highest scorer got 93 and she was only able to get through the first screen.

The graphics are fairly good, but simple—nothing spectacular. Rating the game on a scale of 1 (worst) to 10 (best), I would have to give it a 4. It is a game you easily get frustrated with before you get good at it.

(Acesoft Computer Products, 1680 North Page Drive, Deltona, FL 32725, $24.95 cassette)

—Jim Stewart

'Cooling It' Is A Breeze With *CoCo-Cooler*

It has just occurred to me that I now have a minimum of six switches to turn on just to get my CoCo up and running. I have to turn on the monitor, Drive 0 and Drive 1, the multi-pak interface, the computer itself and *CoCo-Cooler*—the subject of this review.

Well, six switches or not, since installing a *CoCo-Cooler* on our 4K turned 64K machine, I would hesitate to use the CoCo without it—reason being that it very effectively covers all the air vents on the left side of the computer and, thus, if unused it would make the machine get even hotter than it did prior to installation.

Who needs a *CoCo-Cooler?* Let me tell you, I practically had to be hit in the head with a board to become convinced I did. You see, I'm not one who stays up nights worrying about replacing the SAM chip if it should overheat. And, several weeks ago when I noticed some weird stuff happening on the screen, I immediately blamed it on the disk controller which deserves most all of the blame it gets, except this time.

Though the top of the machine was hot enough to cook an egg on, I exhausted every remedy before deciding that maybe—just maybe—I might be suffering from that dreaded malady "hot CoCo," (tacky, but irresistible). Well, *CoCo-Cooler* cooled my machine and my temper.

As advertised, *CoCo-Cooler* takes about one minute to install; you just remove the protective strips covering the adhesive, align it with the side of the machine, press it into place and plug it into a wall outlet. That's it. The *CoCo-Cooler* is contoured to fit snugly in place and the adhesive will sure enough keep it there, once placed, so don't misalign it.

In use, the *CoCo-Cooler* keeps things cool and calm inside your machine, even if a "basket case" is pounding the keyboard. While I haven't gone to the extreme of placing thermometers inside and outside the CoCo to test *CoCo-Cooler's* effectiveness, the top of the machine is now cool to the touch whereas it had been markedly warm before. According to figures supplied by the manufacturer, REM Industries, Inc., in tests during which room temperature was controlled at 80 degrees, temperature readings were taken on the computer case surface just above the power supply and, without *CoCo-Cooler*, a 16K machine's temperature rose to 98 degrees while a 64K machine hit 110 degrees. With *CoCo-Cooler* then attached, both machines cooled off rapidly in 15 minutes—and at the end of an hour the 16K machine registered 84 degrees while the 64K machine had leveled off at about 88 degrees. What more can you ask? Well, maybe a couple of things.

While I have no complaint about *CoCo-Cooler's* effectiveness, I do wish the fan was a bit quieter. True, it makes no more noise than my electric typewriter does—but, then, I turn my typewriter off and on several times a day just so I don't hear that, albeit slight, extra noise or feel the vibration. Personally, I think a smaller fan might do the job just as well and perhaps a measure quieter. Both the fan housing and the fan blade itself are made of high impact plastic and offer little chance of tiny fingers getting into the wrong place. All in the cause of consumer safety, however, I deliberately stuck my finger into the fan and, I'm glad to report, the blade stopped easily.

While it poses no problem in my particular situation, I should point out that the *CoCo-Cooler* does extend five inches out from the left side of the computer—about the same distance the disk controller juts out on the right side. This could possibly pose a problem for those with compact installations—the very same people who are most likely to have overheating problems.

All in all, *CoCo-Cooler* does exactly what it is supposed to do and it does have the professional look of a quality piece of hardware that is designed to do a specific job as efficiently as possible. Individual circumstances will dictate whether you do or don't need a cooling fan for your CoCo, but, if you do, *CoCo-Cooler* is as welcome as a cool breeze.

(REM Industries, Inc., 9420 "B" Lurline Ave., Chatsworth, CA 91311, $39.95)

—Jim Reed

Mathmenu — A Cornucopia Of Mathematical Programs

By Dr. David Finkel

Do you solve mathematical problems in your job? Do you need help with your math homework? Or are you just interested in exploring mathematics? If you answered "yes" to any of these, then you ought to consider *Mathmenu*, from Inter+Action. As a mathematician (Dr. Finkel teaches mathematics and computer science at Bucknell University.), I enjoyed using these programs; perhaps you will, too.

Mathmenu is a collection of 15 mathematical programs, ranging in size from about 1K to over 9K. They are all well-written, well-documented, and—with a few exceptions mentioned below—present mathematically valid approaches to solving problems.

The most impressive of the programs is the surface plotting program, *3D PLOT*, which draws high resolution plots of three-dimensional surfaces on the *PMODE4* screen. The program begins with a menu of choices. One choice gives you several screens of information on how to use the program. Another allows you to enter the function to be graphed. You do this by entering Y as a function of X and Z in standard Color Computer notation, like Y = X*X + SIN(X).

In addition to specifying the function, you have to specify the range of X and Z values to be shown on the screen; these automatically determine the range of Y values. You can also, through the use of the Special Features Menu, control the Y axis scale, the Z axis scale, and the graphing resolution. All these options make for a flexible and powerful graphing capability. You might have to try several different settings of the graphing parameters before you get the graph just right. Fortunately, it's easy to do this; you can return to the menu at any time, erase the current graph, change parameters, and start graphing again, all in a few seconds. You also have the option of graphing more than one function on the same screen.

Once you get a graph just the way you want it, you might like to save it. You can save graphs to tape or disk through the Special Features Menu, which guides you through the necessary steps. The *Mathmenu* package comes with two sample graphs already saved, but I wasn't able to load either of them. However, the graphs I saved to tape myself loaded without any problems. One way to use this save to tape or disk feature is for class demonstrations. You can spend all the time you need beforehand getting the graph to look the way you want, and then demonstrate the finished product. And, you can use it to impress your friends.

This *3D PLOT* program is easy to use, clearly documented and very useful. I do have some suggestions for its improvement, though. First, the axes are turned around from what's standard in math books. Usually, three-dimensional graphs are drawn with the Z axis going up, the Y axis going to the right, and the X axis coming out at you. There's no special reason for this arrangement, but it is

standard; look in any calculus book. *Mathmenu* has the axes turned around. Also it has Y as a function of X and Z, instead of the usual Z as a function of X and Y. Finally, it would help if there were more examples of how to use the various graphing parameters. This program has many powerful options and some examples would help the novice learn how to use them more quickly.

In addition to *3D PLOT*, there are 14 other programs in this package. I'll describe them in less detail.

2D PLOT works like *3D PLOT*, but does two-dimensional plotting. It's somewhat easier to use, since two-dimensional plotting is simpler than three-dimensional plotting. It also offers the save to tape or disk feature.

MATRIXOP and *VECTOROP* do matrix and vector calculations, on matrices up to size 8 by 8, and vectors with up to 20 entries. They perform all the calculations you might like: matrix sum, difference, and product; transpose, inverse, and determinant; and vector sum and difference, cross and dot product, length of a vector, and angle between two vectors. The calculations are all done with good accuracy, subject to the limitations of the Color Computer's numerical operations—about eight accurate places. Since matrix calculations can involve a large number of individual calculations, the small round-off error from each one can accumulate and cause a significant error in the final result. This is especially a problem if the matrix contains both very large and very small entries. Use the results from such calculations with caution.

There's an unfortunate bug in *MATRIXOP*. If you enter a singular matrix—that is, one whose determinant is zero—the program correctly identifies the matrix as singular, but then prints out a non-zero number for the determinant. The author assures me that this bug is being corrected.

Two programs perform numerical calculus functions. *NUMDIFN* calculates approximate derivatives, and *NUMINTEG* calculates approximate definite integrals. The derivative at X is approximated by (F(X+h)-F(X))/h, where h = .00001*X. Because of this fixed formula for h, the approximation may be unreliable if the derivative or the value of X is large. *NUMINTEG* approximates the integral by Simpson's Rule. In this program, you choose the number of intervals, so you can make the approximation as accurate as you like.

The program *LSTSQRS* finds the best fitting curve to a set of up to 100 data points. After you choose whether you want to fit a linear, quadratic, or cubic curve, the program displays the best fitting equation, according to the usual least squares calculations. If you want to do serious analysis of curve fitting problems, you'll probably want more than this program offers: more kinds of curves, a graphics capability, and ways of examining how well the points fit the curve. But *LSTSQRS* does a good job of basic curve fitting.

There are two other programs that are a lot of fun, although it's hard to imagine practical applications for them. *LG#ADD* and *LG#MULT* add and multiply large numbers exactly, with no round-off error. *LG#ADD* handles up to 100 digit numbers, and LG#*MULT* up to 30 digits. Whether or not you need to do calculations with 100 place accuracy, it's nice to be able to, just to show off.

There are six more programs that do fairly routine calculations. *QUADEQN* finds the roots of quadratic equations, *PRIMECK* checks numbers up to one million to see if they're prime or not, *BINOMEXP* calculates binomial coefficients, *BASECONV* converts numbers from one number system to another, for example from decimal to hex, *RECTPOL* converts coordinates between the rectangular and polar systems, and *RPL* allows you to use your Color Computer like a Reverse Polish Logic calculator. All these programs are easy to use, and perform their functions accurately.

All the programs in *Mathmenu*, except for the graphing programs and *RPL*, allow you to print the results out on your printer. The graphing programs include instructions for using your own screen print program to print out the graphs you've created.

Throughout the package, the on-screen instructions are outstanding. Each program includes instructions explaining what the program does and how to use it. Clear prompts are given for entering the data, and for starting the calculations. The printed documentation is adequate, but would be improved with the addition of more examples, especially for the graphing programs and for *RPL*.

And now for the big question: should you buy *Mathmenu*? First, the programs in *Mathmenu* are excellent. They provide a wide range of useful mathematical functions in an easy to use format. Second, the price is a real bargain for a package of 15 programs. The two graphing programs alone are well worth the price. So, if you have use for even a few of these programs, for work or school or your personal interest, you'll get your money's worth from *Mathmenu*.

(Inter+Action, 113 Ward St., Dept. R, New Haven, CT 06519, $44.95 for 16K tape, $49.95 for 32K disk, ECB)

GI Sound Generator: Software Control For Complex Sounds

By Tony DiStefano
Rainbow Contributing Editor

Last month's project was a practical one. This month we'll have some fun. How about making some interesting sound effects? Well, I'll show you how to interface the General Instrument's programmable sound generator number AY-3-8910 to the Color Computer. The features of this chip according to GI are:

- Full software control of sound generation.
- Interfaces to most 8-bit and 16-bit microprocessors.
- Three independently programmed analog outputs.
- Two 8-bit general purpose I/O ports (AY-3-8910).
- Single +5 Volt supply.

This Programmable Sound Generator (from now on, known as the PSG) is a LSI Circuit which can produce a wide variety of complex sounds under software control. Its flexibility makes it useful in applications such as music synthesis, sound effects generation tone signalling and even FSK modems (with a little extra circuitry). All of these sounds can be produced with just a few simple *POKEs*, leaving the processor free to do other tasks like calculating more sounds, updating the screen or doing graphic anamation (in the case of arcade type games). One or two pokes can produce sounds that carry on for several seconds, or even continuously.

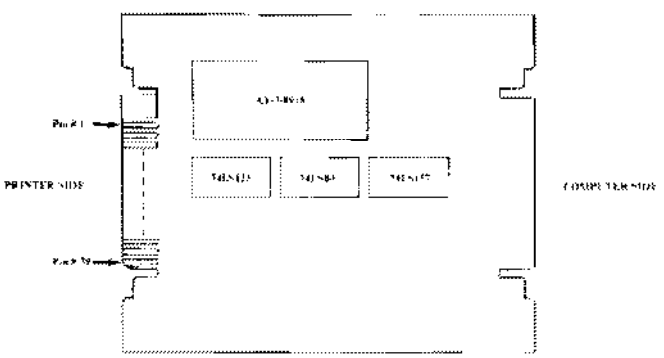
This PSG is a register-oriented device. This means that communication between the processor and the PSG is based on the concept of memory-mapped I/O. The control commands are issued to the PSG by writing (*POKEs* or *STAs*) to two memory locations. The first location (memory mapped at hex address FF65) is to select which internal register you wish to access. The second memory location is for the data you wish to enter or retrieve and is at hex address FF64. All functions of the PSG are controlled through 16 registers which once programmed, generate and sustain sounds on its own. More on how to program it later.

Refer to List #1 to get all the parts needed for this project. It is not hard to put together, but like any electronic project, care should be taken when working with static sensitive IC's.

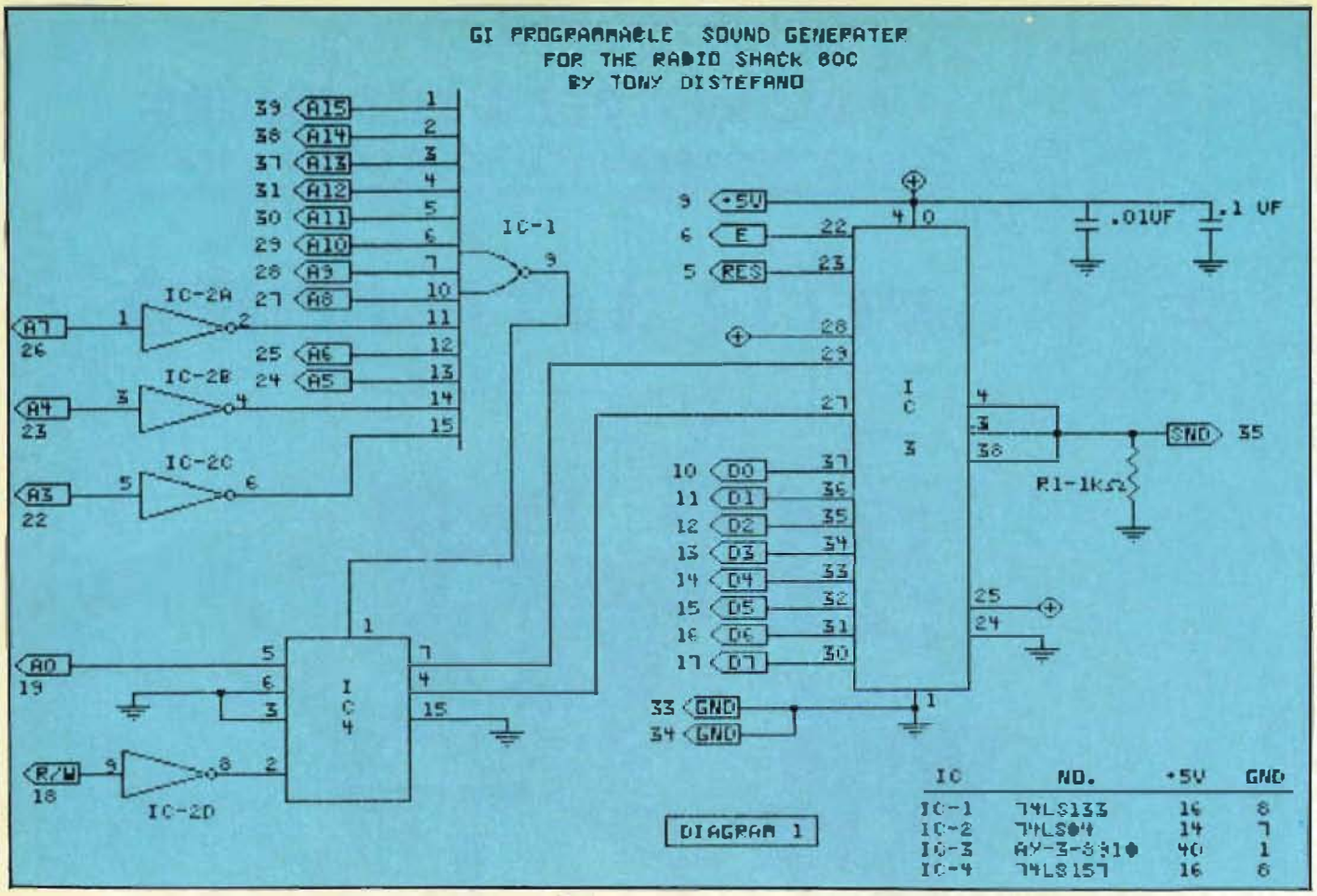
Quantity	Part #	Desc
1	AY-3-8910	PSG
1	74LS133	TTL
1	74LS04	TTL
1	74LS157	TTL
2	16 PIN	SOCKET
1	14 PIN	SOCKET
1	40 PIN	SOCKET
2	.01 UF	CAPACITORS
1	10K OHMS	RESISTOR
1	PROJECT BOARD	RS # 276-163

LIST # 1

With the schematic (Diagram 1) in one hand and a soldering iron in the other, it's time to put the board together. Start by getting the ground and B-Plus buses wired in. It is best to wire all the connections to the connector on the side of the board that has the lowest number. That's the side with the number one on the edge. The two buses run close to the edge connector, it will be easier to connect to. Refer to Diagram 2 for the proper layout of the sockets. The rest of the soldering is quite straightforward. Follow the schematic and cross off each line after it's done. This will eliminate any missed wires. When you are finished, clean the board in the usual manner. Check again the wiring with the schematic, remember that the Y'er is not buffered and is not forgiving of wiring errors. A short can cause many headaches. When you're finished, insert the chips (remember pin 1's) and plug it in. Turn the



(Tony DiStefano is well known as an early specialist in Color Computer hardware projects. He is one of the acknowledged experts on the "insides" of CoCo.)



computer on, when you get the familiar sign on, turn up the sound, type in and *RUN* this program.

```
10 AUDIO ON
20 POKE $HFF01,$HB4
30 POKE &HFF03,&H3F
40 SR = &HFF65 : WD = &HFF64
50 POKE SR,RND(15)-1 : POKE WD,RND(256)-1 :
   GOTO 50
```

This short program will generate ramdom sounds, beeps, pops, and whistles in the speaker of your TV. This is more or less just a test to make sure that the circuit is working. (You will have to use your imagination to come up with better software.) If you do not get any sound, check the wiring again; this circuit does work. I have a working model right here in front of me. Here, just put your ear a little closer and listen. Can you hear it? I told you it works. Okay, enough foolin' around, the following descriptions of the PSG are excerpts taken from the GI product description manual.

Sound Generating Blocks

The basic blocks in the PSG which produce the programmed sounds include:

Tone Generators	Produce the basic square wave tone frequencies for each channel (A, B, C)
Noise Generator	Produces a frequency modulated pseudo random pulse width square wave output.
Mixers	Combine the outputs of the Tone Generators and the Noise Generator. One for each channel (A, B, C).
Amplitude Control	Provides the D/A Converters with either a fixed or variable amplitude pattern. The fixed amplitude is under direct CPU control; the variable amplitude is accomplished by using the output of the Envelope Generator.
Envelope Generator	Produces an envelope pattern which can be used to amplitude modulate the output of each Mixer.
D/A Converters	The three D/A Converters each produce up to a 16 level output signal as determined by the Amplitude Control.

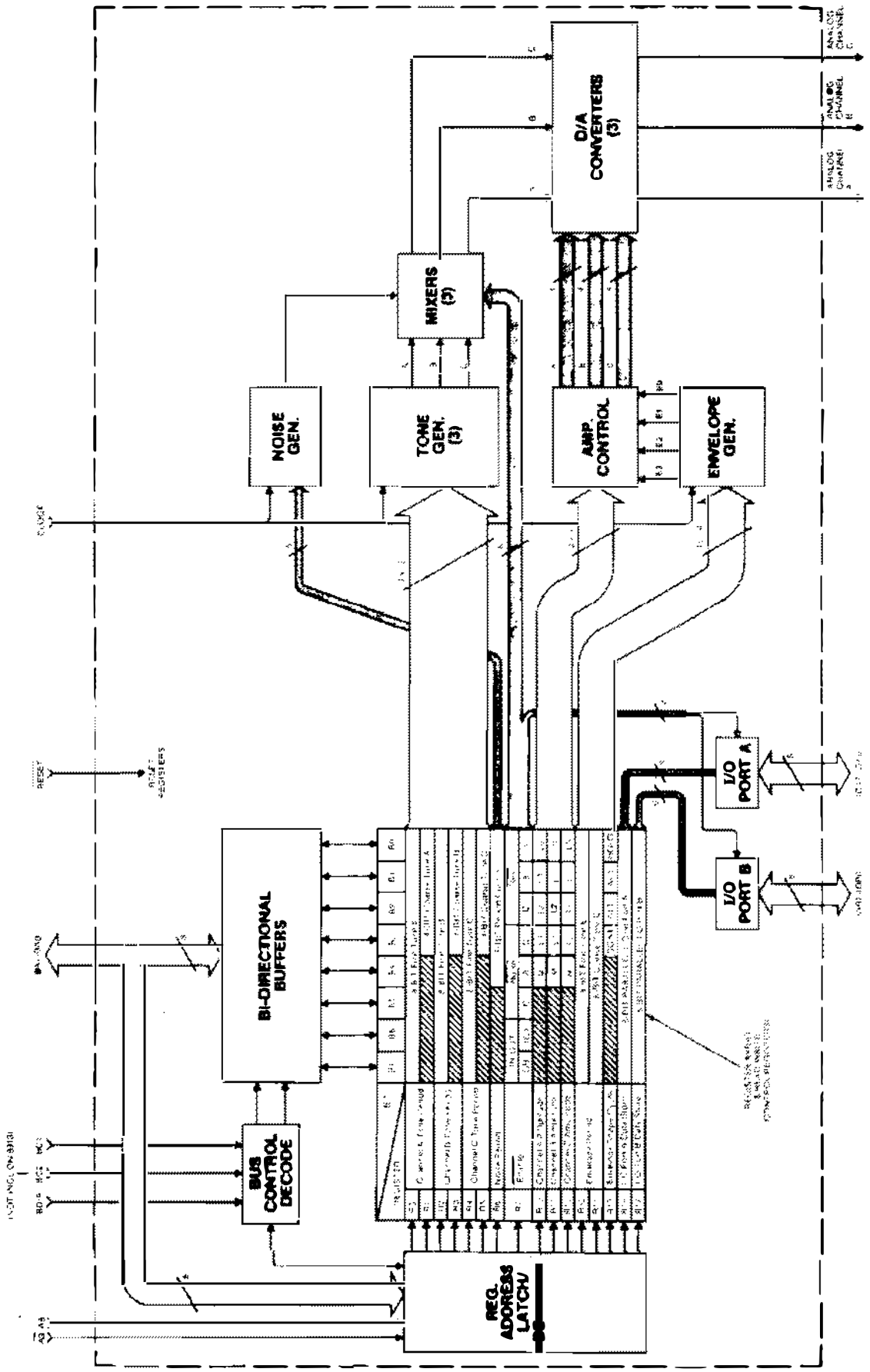
Operation

Since all functions of the PSG are controlled by the processor via a series of register loads, a detailed description of the PSG operation can best be accomplished by relating each PSG function to the control of its corresponding register. The function of creating or programming a specific sound or sound effect logically follows the control sequence listed:

Operation	Registers	Function
Tone Generator Control	R0-R5	Program tone periods.
Noise Generator Control	R6	Program noise period.
Mixer Control	R7	Enable tone and/or noise on selected channels.
Amplitude Control	R10-R12	Select "fixed" or "envelope-variable" amplitudes.
Envelope Generator Control	R13-R15	Program envelope period and select envelope pattern

Tone Generator Control (Registers R10, R1, R2, R3, R4, R5)

The frequency of each square wave generated by the three Tone Generators (one each for Channels A, B, and C) is



PSG BLOCK DIAGRAM

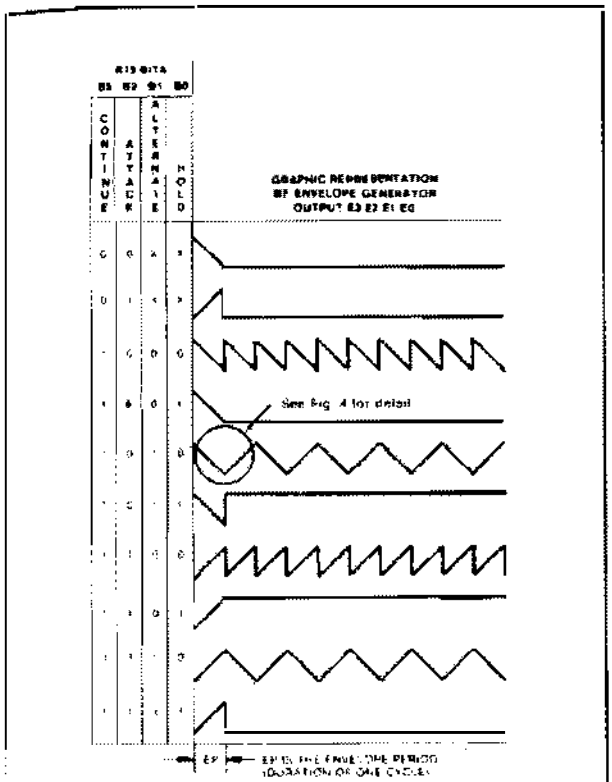


Fig. 1 ENVELOPE SHAPE/CYCLE OPERATION

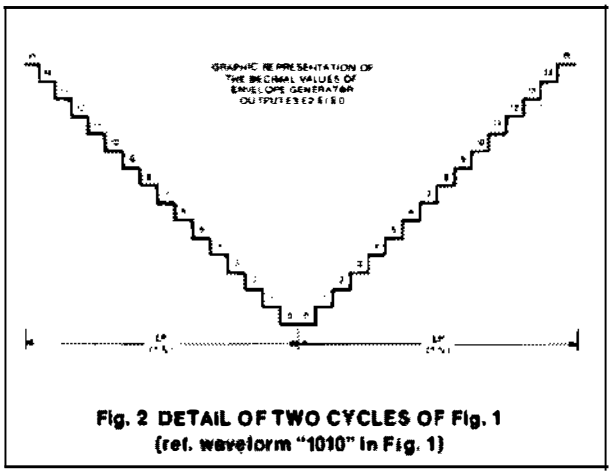


Fig. 2 DETAIL OF TWO CYCLES OF Fig. 1
(ref. waveform "1010" in Fig. 1)

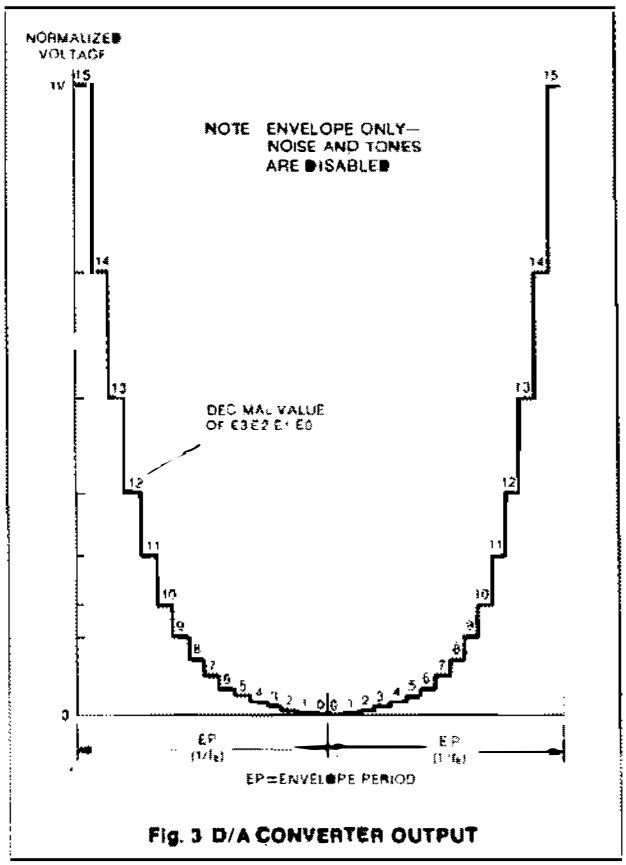


Fig. 3 D/A CONVERTER OUTPUT

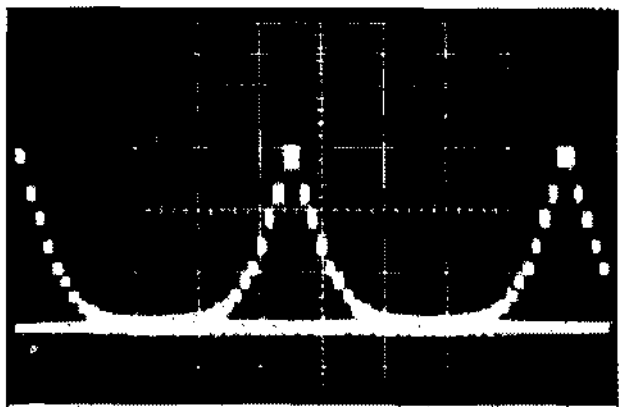


Fig. 4 SINGLE TONE WITH ENVELOPE SHAPE/CYCLE PATTERN 1010

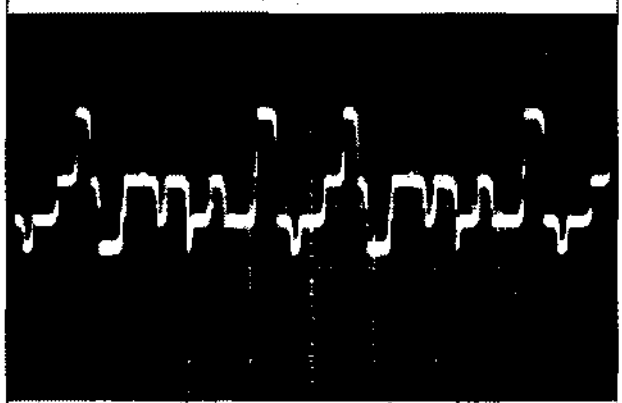
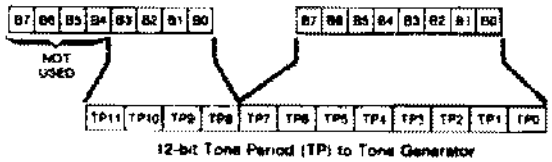


Fig. 5 MIXTURE OF THREE TONES
WITH FIXED AMPLITUDES

obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 12-bit Tone Period value. Each 12-bit value is obtained in the PSG by combining the contents of the relative Coarse and Fine Tune registers, as illustrated in the following:

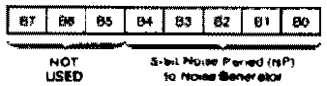
Coarse Tune Register	Channel	Fine Tune Register
R1	A	R0
R3	B	R2
R5	C	R4



Noise Generator Control (Register R6)

The frequency of the noise source is obtained in the PSG by first cutting down the input clock by 16, then by further counting down the result by the programmed 5-bit Noise Period value. This 5-bit value consists of the lower 5 bits (B4-B0) of register R6, as illustrated in the following:

Noise Period Register R6



Mixer Control—I/● Enable (Register R7)

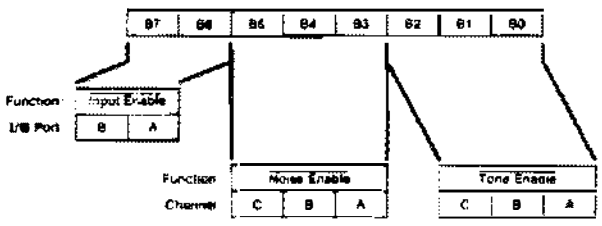
Register R7 is a multi-function Enable register which controls the three Noise/Tone Mixers and the two general purpose I/O Ports.

The Mixers, as previously described, combine the noise and tone frequencies for each of the three channels. The determination of combining neither/either/both noise and tone frequencies on each channel is made by the state of bits B5-B0 or R7.

The direction (input or output) of the two general purpose I/O Ports (IOA and IOB) is determined by the state of bits B7 and B6 of R7.

These functions are illustrated in the following:

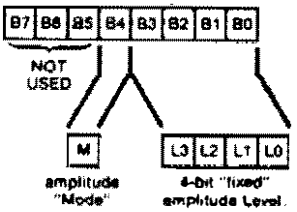
Mixer Control—I/O Enable Register R7



Amplitude Control (Registers R10, R11, R12)

The amplitudes of the signals generated by each of the three D/A Converters (one each for Channels A, B, and C) is determined by the contents of the lower 5 bits (B4-B0) of registers R10, R11, and R12 as illustrated in the following:

Amplitude Control Register	Channel
R10	A
R11	B
R12	C



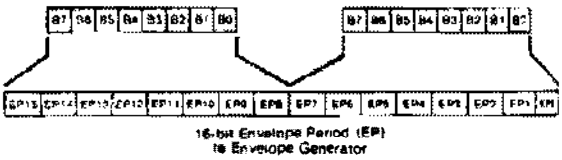
Envelope Generator Control (Registers R13, R14, R15)

To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG: first, it is possible to vary the frequency of the envelope using registers R13 and R14; and second, the relative shape and cycle pattern of the envelope can be varied using register R15. The following paragraphs explain the details of the envelope control functions, describing first the envelope period control and then the envelope shape/cycle control.

Envelope Period Control Registers R13, R14)

The frequency of the envelope is obtained in the PSG by first counting down the input clock by 256, then by further counting down the result by the progrmmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated in the following:

Envelope Coarse Tune Register R14	Envelope Fine Tune Register R13

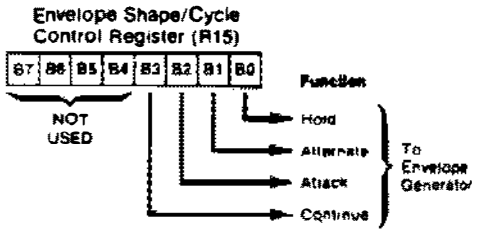


Envelope Shape/Cycle Control (Register R15)

The Envelope Generator further counts down the envelope frequency by 16, producing a 16-state per cycle envelope pattern as defined by its 4-bit counter output, E3 E2 E1 E0. The particular shape and cycle pattern of any desired envelope is accomplished by controlling the count patteru (count up/count down) of the 4-bit counter and by defining a single-cycle or repeat-cycle patern.

This envelope shape/cycle control is contained in the lower 4 bit (B3-B0) of register R15. Each of these 4 bits

controls a function on the envelope generator, as illustrated in the following:



D/A Converter Operation

Since the primary use of the PSG is to produce sound for the highly imperfect amplitude detection mechanism of the human ear, the D/A conversion is performed in logarithmic steps with a normalized voltage range of from 1 to 1 Volt. The specific amplitude control of each of the three D/A Converters is accomplished by the three sets of 4-bit outputs of the Amplitude Control block, while the Mixer outputs provide the base signal frequency (Noise and/or Tone).

(Reprinted by permission, courtesy ***General Instruments/Microelectronics)***

Blow By Blow Account Of An Emerging Combat System

By Bill Nolan
Rainbow Contributing Editor

Welcome to the Dragon's Byte. As most of you know, this is a column about using your computer to aid you in the playing of fantasy role-playing games. Every few months I repeat the following information, so if you missed it, here's your chance. Fantasy role-playing games—such as *Dungeons and Dragons, Runequest, Tunnels and Trolls,* or *Worlds of Wonder*—are not computer games. They are also not board games. What they are is a separate type of game in which the players each try to assume the personality of a made up character and act out their part in some sort of adventure.

Just as the term "fantasy role-playing game" implies, participants are playing a role in some sort of fantasy. Children (we were all children) play simple versions of these games and call them "cops and robbers" or "cowboys and Indians." The adult versions are the same in concept, but are far more complex. The rule books all appear to have been written by former legislators, and make about as much sense as the tax laws. In this column we strive together to get help from our computers as we play our favorite fantasy game.

The programs here are not games. They are programs designed to simplify and speed up some of the more tedious parts of playing these very enjoyable games.

Enough of that. For the last two months, we have been working on a computerized combat system. In doing this we have examined some ways to make the not very random *RND* unpredictable, and we have looked at the reasons for putting a combat system on the computer. We have also determined the major things we want the system to do for us. If you missed this column in July or August, you will want to get those issues and read them, because a lot of the background for what we are doing this month was presented there.

Included this month you will find a program. This is not the whole combat system quite yet, but the program works fine as is. In future columns we will be adding some things to it, but we will try not to change much of this foundation program. After all, I don't want people sitting at their keyboards cursing me.

You will see that the program has five main sections, and five choices on the main menu. Contrary to what you may believe, this was actually planned, and was not a coincidence. Line numbers from 0 to 999 are used to set up the program. Lines from 1000 to 1999 are used to save and load information, those from 2000 to 2999 are for editing character, those from 3000 to 5999 are reserved to handle the combat itself, and the lines from 6000 to 6999 are for looking at the information about each character.

In addition to this, we have lines from 7000 to 7999 which we use for subroutines, and lines beginning at 10000 for *DATA* statements. In many cases, as you can see below, we don't need all those line numbers, but starting each section on an even program also keeps track of the current hit points of each fighter and tells you when someone is down. When one side of a combat is down, it skips over that one in the future. The program tells you what round you are in, and after every round it gives you the option of breaking off the combat.

Lines 3000 to 3050 find out how many combats there will be, who is involved in each, what weapon they are using, and whether it has any magical plusses. In line 3050, the round counter "RC" is set to 1.

The combat loop starts in line 3060, and the values needed to hit are figured in lines 3085 and 3090. This is figured like this: (base chance) - (level - 1) - (class bonus) - (weapon plus) - (strength bonus) + or - (magical adjustment) + (10 - opponent's armor class) + (opponent's speed bonus) + ((opponent's level - 6) + opponent's class adjustment) - ((fighter's level - 6) + fighter's class adjustment). See, nothing to it!

Once it ("it" is the computer) knows the numbers needed to hit, it either rolls the dice or has you input the results. This is done in lines 3120 to 3160. In line 3170, the program figures the initiative. This is based on the difference between the number needed to hit and the actual roll, so if I need a 10, but roll a 20, and you need a 4, but roll a 15, then you have initiative, because your difference is greater.

(Bill Nolan is a principal in Prickly-Pear Software, DM's a weekly game of Dungeons & Dragons, and teaches Programming In BASIC at a local college.)

Lines 3200 to 3410 compute the results of that combat round, and line 4000 recycles the whole thing. Keep in mind that in doing their jobs these line numbers call out to subroutines from time to time. These subroutines are located in lines 7000 to 7500, and I will cover each of them in a moment.

Lines 6000 to 6100 simply print out each combatant stored in the file one after the other.

Line 7000 gets a single number from the keyboard within the range of 1 to XX, with XX being set in the line that calls the routine. It also helps out the random a bit, and then returns the number in the variable K.

Line 7010 just waits for any key to be pressed and then returns, as in "PRESS ANY KEY TO CONTINUE."

Line 7100 gets "Y" for yes or "N" for no from the keyboard and then returns.

Line 7200 prints an error message if you try to hold a combat and have less than two combatants available. Remember, it takes two to make a fight.

Line 7300 gets the weapon damage multiplier for those natural combatants using natural weapons like teeth, claws, or tails. Your best bet here is to look at line 10010, which has the weapon data. The number following each weapon is its multiplier, so if you know what the natural attack can do as compared to some number, it is much easier to keep track of where everything is located. It is more than a little embarassing to get lost in your own program.

What we are going to do this month is go through the program section by section, so you can see what is done where. That way, if you don't like something, you'll know where to go to change it. Fantasy games are very personal, and the size of the original rules is exceeded only by the size of the changes made by the average group of players.

The first part is pretty standard. Lines 10—17 set up the arrays and read some *DATA* into them. As you can see, we are using several one and two dimension arrays, and one array of three dimensions. Line 20 prints the menu of choices, line 30 gets your choice, and line 40 branches to the section you have chosen. The block of lines from 100 to 270 asks you to input all of the necessary information about a potential combatant. This could be either a character or a monster, and when you have input the information, the program returns you to the menu. The variable CC (Control Counter) is used to keep track of how many combatants are in the file, and that array—M$—is dimensioned to hold 31. We are keeping track of name, class, combat adjustment, level, strength adjustment, speed adjustment, number of attacks, base hit points, current hit points, magical adjustments (not including weapon adjustments, which are handled elsewhere), and armor class. By the way, the program does not trap out multiple uses of the same name. Since the searches are by name, and since they stop searching as soon as they find the target name, you must be careful not to use the same name twice.

The area from 1000 to 1130 is used to save or load data. It is currently set up for disk, and if you are using tape, you will have to make some changes. To start with, you will have to change every "#1" to be "#-1," because the disk is device 1, while the tape is device -1. Also, you will have to change the *WRITE* in lines 1055 and 1070 to *PRINT*. After you have input all of your characters and monsters, you should save them to disk or tape before continuing. Be sure you have a disk in the drive or a tape in the recorder with play and record pressed before trying to save, or you may lose your data. If you ever bomb out of the program, you can usually recover your data by immediately typing GOTO 20 and ENTER. This should put you back into the menu.

As we said above, lines 2000 to 2140 are used to change a single data item about a character or monster. If you want the change to be permanent, you will have to save your file again.

That brings us to line 3000, and a sudden increase in complexity. The program will handle nine combats at the same time, with two participants in each. The same combatant can be involved in more than one of these combats. The program figures the number needed to hit, the initiative, and the damage. It gives you the option of letting it roll the dice or having you roll them and enter the values. This is important, as many gamers (myself included) like to roll the dice themselves. The weapon, you can easily figure out a suitable multiplier. For example, a two-handed sword has a multiplier of 1.5, so if you think a particular dragon's bite (byte?) is twice as bad as the sword, give it a multiplier of three. If you don't like the damage you are getting, change all the multipliers.

Lines 7400 and 7410 roll the combat dice if you select that option.

Line 7500 checks to see if you want to break off the combat. Notice that this subroutine calls to the subroutine in line 7100.

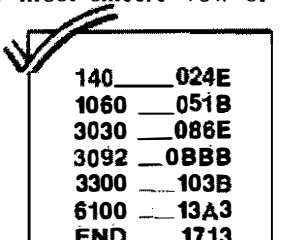
The subroutine at 7600 is used to run a check when one of the combatants is down. Since the combatant may be involved in more than one melee, the program checks to see if there are any others involving this name. If any are found, a "0" is inserted in the array "F" at that number. A "1" in array "F" indicates a fight in progress.

Well, I guess that about covers the program on a line by line basis. The actual use of the program is pretty simple. Once you have typed it in and have all those nasty typos corrected, *RUN* the program. The first thing you will see is the menu, and your first selection should be "ADD A COMBATANT." You can't use any of the other options until you have put at least a couple of combatants into the file, so get out a couple of your favorite character sheets and type in the information as the program asks for it.

Once you have your characters and monsters in the file, save the file to tape or disk. If you don't do this, you'll be sorry.

Now you can examine your file, edit a particular combatant, or try out a combat. I think you will find the program to be pretty much self-prompting, and wherever possible, I have tried to trap out bad inputs.

Next month we will be working on some expansions and improvements to this program. We will surely add fumble and critical hit tables, and I have some other ideas I think you'll like as well. In the meantime, if you have suggestions or comments, let me know. If you find any bugs (heaven forbid!), immediately swear your most sincere vow of secrecy and call me so I can fix it.



The listing:

```
10 CLS:X=RND(-TIMER):DIMM$(30,10
),W$(13,1),C$(10),CI$(9,2,3),F(9
),WP(9,2)
```

```
15 FOR X=0 TO 10:READ C$(X):NEXT
 X
17 FOR X=1 TO 13:READ W$(X,0):RE
ADW$(X,1):NEXT X
20 CLS:PRINT@46,"MENU";:PRINT@10
2,"1. ADD A COMBATANT";:PRINT@16
6,"2. SAVE OR LOAD DATA";:PRINT@
230,"3. EDIT A COMBATANT";:PRINT
@294,"4. COMBAT SECTION";:PRINT@
358,"5. EXAMINE COMBATANTS";:PRI
NT@422,"YOUR CHOICE? (1-5)";:K$=
INKEY$
30 XX=5:GOSUB 7000
40 ON K GOTO 100,1000,2000,3000,
6000
100 CLS:PRINT@34,"";:INPUT"COMBA
TANT'S NAME";X$
120 CLS:PRINT:PRINT" 1. FIGHTER"
:PRINT" 2. THIEF":PRINT" 3. CLER
IC":PRINT" 4. MAGICIAN":PRINT:PR
INT:PRINT" KEY YOUR CHOICE (1-4)
":K$=INKEY$
130 XX=4:GOSUB 7000
135 M$(CC,0)=X$
140 ON K GOSUB 150,160,170,180:G
OTO200
150 M$(CC,1)="FIGHTER":M$(CC,2)=
"6":RETURN
160 M$(CC,1)="THIEF":M$(CC,2)="3
":RETURN
170 M$(CC,1)="CLERIC":M$(CC,2)="
2":RETURN
180 M$(CC,1)="MAGICIAN":M$(CC,2)
="0":RETURN
200 CLS:PRINT@34,C$(3);:INPUT M$
(CC,3)
210 CLS:PRINT@34,C$(4);:INPUT M$
(CC,4)
220 CLS:PRINT@34,C$(5);:INPUT M$
(CC,5)
230 CLS:PRINT@34,C$(6);:INPUT M$
(CC,6)
240 CLS:PRINT@34,C$(7);:INPUT M$
(CC,7):M$(CC,8)=M$(CC,7)
250 CLS:PRINT@34,C$(9);:INPUT M$
(CC,9)
260 CLS:PRINT@34,C$(10);:INPUT M
$(CC,10)
270 CC=CC+1:GOTO 20
1000 CLS:PRINT:PRINT" DO YOU WAN
T TO SAVE OR LOAD?":PRINT:PRINT"
 PRESS 'S' OR 'L'.":K$=INKEY$
1010 J$=INKEY$:IF J$<>"S" AND J$
<>"L" THEN 1010 ELSE SOUND 150,1
1020 IF J$="L" THEN PRINT:PRINT"
 LOADING WILL ERASE CURRENT":PRI
NT" MEMORY.  ARE YOU SURE (Y/N)"
:K$=INKEY$:ELSE GOTO 1040
1030 K$=INKEY$:GOSUB 7100:IF K$=
"N" THEN 20
1040 IF J$="L" THEN1100
1050 OPEN"O",#1,"COMBAT"
1055 WRITE #1,CC
1060 FOR X=0 TO 20:FOR Y=0 TO 10
1070 WRITE #1,M$(X,Y)
1080 NEXT Y,X:CLOSE 1:GOTO20
1100 OPEN"I",#1,"COMBAT"
1105 INPUT #1,CC
1110 FOR X=0 TO 20:FOR Y=0 TO 10
1120 INPUT #1,M$(X,Y)
1130 NEXTY,X:CLOSE 1:GOTO 20
2000 IF CC=0 THEN GOTO 6100 ELSE
 CLS:PRINT:PRINT" EDIT WHAT COMB
ATANT":PRINT" ";:INPUT S$
2010 FOR X=0 TO CC-1:IF S$=M$(X,
0) THEN 2100 ELSE NEXT X
2020 CLS:PRINT"FILE NOT FOUND":F
OR X=1 TO 1000:NEXT X:GOTO 20
2100 CLS:FOR Y=0 TO 10:PRINT STR
$(Y);". ";C$(Y);" = ";M$(X,Y):NE
XT Y
2110 PRINT:PRINT"INPUT THE NUMBE
R TO EDIT":PRINT
2120 INPUT Z:IF Z<0 OR Z>10 THEN
 2120 ELSE SOUND 150,1
2130 CLS:PRINT"COMBATANT = ";M$(
X,0)
2140 PRINT:PRINT" YOU ARE CHANGI
NG":PRINT" ";C$(Z):PRINT:PRINT"
```

```
ENTER NEW INFORMATION":INPUT" ";
M$(X,Z):GOTO 20
3000 CLS:IF CC<2 THEN GOTO 7200
ELSE PRINT:PRINT" HOW MANY COMBA
TS?  (1-9)":K$=INKEY$:XX=9:GOSUB
 7000:NC=K:FOR XJ=1 TO 9:F(XJ)=0
:NEXT:FOR XJ=1 TO NC:F(XJ)=1:NEX
T
3010 FOR X=1 TO NC:FOR Y=1 TO 2
3020 CLS:PRINT:PRINT" COMBAT #";
X:PRINT:PRINT" WHO IS COMBATANT
#";Y:INPUT" ";X$
3030 FOR Z=0 TO CC-1:IF X$=M$(Z,
0) THEN CI$(X,Y,0)=X$:CI$(X,Y,3)
=STR$(Z):ELSE NEXT Z:PRINT"NAME
NOT FOUND":FOR XJ=1 TO 1000:NEXT
 XJ:GOTO 3020
3040 CLS:PRINT X$:PRINT:FOR Z=1
TO 13:PRINT STR$(Z);". ";W$(Z,0)
:NEXT Z:INPUT" CHOOSE WEAPON (1-
13)";WT:IF WT<1 OR WT>13 THEN 30
40 ELSE CI$(X,Y,1)=W$(WT,0):CI$(
X,Y,2)=W$(WT,1):IF WT=13 THEN GO
SUB 7300
3045 CLS:PRINT:PRINT"ENTER WEAPO
N'S PLUS TO HIT":INPUT" ";WP(X,Y
)
3050 NEXT Y,X:RC=1
3060 FOR X=1 TO NC:CLS:IF F(X)=0
 THEN NEXT X:GOTO 20
3070 PRINT"COMBAT #";X;"   ROUND
 #";RC:PRINT"# 1 COMBATANT IS ";
CI$(X,1,0):PRINT"# 2 COMBATANT I
S ";CI$(X,2,0)
3080 X1=VAL(CI$(X,1,3)):X2=VAL(C
I$(X,2,3))
3085 H1=22-(VAL(M$(X1,3))-1)-(VA
L(M$(X1,2)))-WP(X,1)-(VAL(M$(X1,
4)))-(VAL(M$(X1,9)))+(10-(VAL(M$
(X2,10))))+(VAL(M$(X2,5)))+((VAL
(M$(X2,3))-6)+(VAL(M$(X2,2))))-(
(VAL(M$(X1,3))-6)+(VAL(M$(X1,2))
))
3087 IF H1<2 THEN H1=2 ELSE IF H
1>30 THEN H1=30
3090 H2=22-(VAL(M$(X2,3))-1)-(VA
L(M$(X2,2)))-WP(X,2)-(VAL(M$(X2,
4)))-(VAL(M$(X2,9)))+(10-(VAL(M$
(X1,10))))+(VAL(M$(X1,5)))+((VAL
(M$(X1,3))-6)+(VAL(M$(X1,2))))-(
(VAL(M$(X2,3))-6)+(VAL(M$(X2,2))
))
3092 IF H2<2 THEN H2=2 ELSE IF H
2>30 THEN H2=30
3100 PRINT" ";CI$(X,1,0);" NEEDS
";H1;"TO HIT"
3110 PRINT" ";CI$(X,2,0);" NEEDS
";H2;"TO HIT"
3120 PRINT" WANT ME TO ROLL THE
DICE?":K$=INKEY$:GOSUB 7100:IF K
$="Y" THEN 7400
3130 PRINT" ";CI$(X,1,0);"'S ROL
L";:INPUT R1
3140 PRINT" ";CI$(X,2,0);"'S ROL
L";:INPUT R2
3145 R1=R1+1:R2=R2+1:IF (R1-H1)=
<0 AND (R2-H2)=<0 THEN PRINT" NE
ITHER HIT":GOTO 4000
3150 IF (R1-H1)>0 THEN PRINT " "
;CI$(X,1,0);" HIT. ":D1=INT((R1-
H1)*VAL(CI$(X,1,2)))
3160 IF (R2-H2)>0 THEN PRINT " "
;CI$(X,2,0);" HIT. ":D2=INT((R2-
H2)*VAL(CI$(X,2,2)))
3170 IF(R1-H1)>(R2-H2) THEN I=1
ELSE IF (R1-H1)<(R2-H2) THEN I=2
 ELSE I=3
3180 ON I GOTO 3200,3300,3400
3200 PRINT" ";CI$(X,1,0);" HAS I
NITIATIVE":PRINT" ";CI$(X,1,0);"
'S DAMAGE IS";D1:XJ=VAL(M$(VAL(C
I$(X,2,3)),8)):XJ=XJ-D1:M$(VAL(C
I$(X,2,3)),8)=STR$(XJ):IF XJ=<0
THEN PRINT" "CI$(X,2,0);" IS DOW
N":F(X)=0:JJ$=CI$(X,2,0):GOSUB 7
600:GOTO 4000
3210 IF(R2-H2)>0 THEN PRINT" ";C
I$(X,2,0);"'S DAMAGE IS";D2:XJ=V
AL(M$(VAL(CI$(X,1,3)),8)):XJ=XJ-
```

```
D2:M$(VAL(CI$(X,1,3)),B)=STR$(XJ
):IF XJ=<Ø THEN PRINT " "¦CI$(X,
1,Ø)¦" IS DOWN":F(X)=Ø:JJ$=CI$(X
,1,Ø):GOSUB 7600:GOTO 4000
3220 GOTO 4000
3300 PRINT" "¦CI$(X,2,Ø)¦" HAS I
NITIATIVE":PRINT" "¦CI$(X,2,Ø)¦"
'S DAMAGE IS"¦D2:XJ=VAL(M$(VAL(C
I$(X,1,3)),B)):XJ=XJ-D2:M$(VAL(C
I$(X,1,3)),B)=STR$(XJ):IF XJ=<Ø
THEN PRINT" "CI$(X,1,Ø)¦" IS DOW
N":F(X)=Ø:JJ$=CI$(X,1,Ø):GOSUB 7
600:GOTO 4000
3310 IF(R1-H1)>Ø THEN PRINT" "¦C
I$(X,1,Ø)¦"'S DAMAGE IS"¦D1:XJ=V
AL(M$(VAL(CI$(X,2,3)),B)):XJ=XJ-
D1:M$(VAL(CI$(X,2,3)),B)=STR$(XJ
):IF XJ=<Ø THEN PRINT " "¦CI$(X,
2,Ø)¦" IS DOWN":F(X)=Ø:JJ$=CI$(X
,2,Ø):GOSUB 7600:GOTO 4000
3320 GOTO 4000
3400 PRINT" SIMULTANEOUS INITIAT
IVE":PRINT" "¦CI$(X,1,Ø)¦"'S DAM
AGE IS"¦D1:XJ=VAL(M$(VAL(CI$(X,2
,3)),B)):XJ=XJ-D1:M$(VAL(CI$(X,2
,3)),B)=STR$(XJ):IF XJ=<Ø THEN P
RINT" "CI$(X,2,Ø)¦"IS DOWN":F(X)
=Ø:JJ$=CI$(X,2,Ø):GOSUB 7600
3410 IF(R2-H2)>Ø THEN PRINT" "¦C
I$(X,2,Ø)¦"'S DAMAGE IS"¦D2:XJ=V
AL(M$(VAL(CI$(X,1,3)),B)):XJ=XJ-
D2:M$(VAL(CI$(X,1,3)),B)=STR$(XJ
):IF XJ=<Ø THEN PRINT " "¦CI$(X,
1,Ø)¦" IS DOWN":F(X)=Ø:JJ$=CI$(X
,1,Ø):GOSUB 7600:GOTO 4000
4000 PRINT" PRESS ANY KEY":GOSUB
 7010:NEXT X:RC=RC+1:X8=Ø:FOR X9
=1 TO 5:X8=X8+F(X9):NEXT X9:IF X
8=<Ø THEN 20 ELSE GOSUB 7500:IF
K$="N" THEN 20 ELSE 3060
6000 IF CC=Ø THEN 6100 ELSE FOR
X=Ø TO CC-1:CLS
6010 FOR Y=Ø TO 10:PRINTC$(Y)¦"
= "¦M$(X,Y):NEXT Y:PRINT:PRINT"P
RESS ANY KEY":K$=INKEY$:GOSUB 70
10:NEXT X:GOTO20
6100 CLS:PRINT"FILE EMPTY":FOR X
=1 TO 1000:NEXT:GOTO 20
7000 K$=INKEY$:K=VAL(K$):IF K<1
OR K>XX THEN X=RND(Ø):GOTO 7000:
ELSE SOUND 150,1:RETURN
7010 IF INKEY$="" THEN 7010 ELSE
 SOUND 150,1:RETURN
7100 K$=INKEY$:IF K$<>"Y" AND K$
<>"N" THEN 7100 ELSE SOUND 150,1
:RETURN
7200 PRINT"YOU MUST HAVE AT LEAS
T TWO       COMBATANTS IN THE FIL
E TO HAVE  A COMBAT":FOR X=1 TO
1000:NEXT:GOTO 20
7300 CLS:PRINT:PRINT" NATURAL WE
APON.  INPUT WEAPON  DAMAGE MUL
TIPLIER.":INPUT" "¦CI$(X,Y,2):RE
TURN
7400 R1=RND(30):PRINT" "¦CI$(X,1
,Ø)¦"'S ROLL"¦R1
7410 R2=RND(30):PRINT" "¦CI$(X,2
,Ø)¦"'S ROLL"¦R2:GOTO3145
7500 CLS:PRINT:PRINT" ANOTHER CO
MBAT ROUND? (Y/N)":K$=INKEY$:GOS
UB 7100:RETURN
7600 FOR J1=1 TO 9:IF CI$(J1,J2,
Ø)=JJ$ OR CI$(J1,2,Ø)=JJ$ THEN F
(J1)=Ø
7610 NEXT J1:RETURN
10000 DATA NAME,CLASS,COMBAT ADJ
USTMENT,LEVEL,STRENGTH ADJUSTMEN
T,SPEED ADJUSTMENT,NUMBER OF ATT
ACKS,NUMBER OF HIT POINTS,CURREN
T HIT POINTS,PLUS OR MINUS FOR M
AGIC,ARMOR CLASS
10010 DATA SHORT SWORD,1,LONG SW
ORD,1.2,TWO-HANDED SWORD,1.5,DAG
GAR,.5,MACE,1,LONG BOW,1,CROSSBO
W,1.2,AXE,1,SPEAR,.7,SLING,.5,HA
ND AXE,.6,FLAIL,.9,NATURAL WEAPO
N,Ø
```

Talking Speller 'Works' — But Is It Worth Using?

Superior Graphic Software's *Talking Speller* is a drill program using voice playback through the television monitor to preview a set of spelling words, dictate them to a student, prompt corrections and compute the student's test score. It requires 16K of memory, but Extended Color BASIC is not required. When loaded from cassette, the menu shows two program options.

Program Option 1 allows the teacher to create a spelling drill by typing in selected words, then dictating them into the cassette recorder. The program prompts the dictation by flashing the words on the screen, one at a time, then turning the cassette motor on and off, allowing approximately seven seconds for each word to be spoken (and recorded) twice. There is no limit to the number of words that can be entered in the drill, and the teacher can prepare and file on cassette a number of different spelling lists.

Program Option 2 tests the student, who is first asked if he/she wants to preview the list. If the answer is "yes," the words are spelled, one letter at a time, accompanied by tones from the speaker. Then the test begins: The program turns on the cassette motor, plays the dictated words through the monitor speaker, pausing after each word to wait for the student's answer. If the word is spelled correctly any of three times, the program plays a fanfare and prints praise on the screen. If the student misses once or twice, the program supplies hints in the form of blank spaces on the screen showing which, if any, letters the student has correct. After the third unsuccessful try, the program fills the screen with the word, correctly spelled, then moves to the next word on the list. At the conclusion of Option 2, the student's score is computed, including the number of words spelled correctly on the first, second and third try.

Talking Speller does everything its designers claim for it. It frees a teacher from dictating spelling lists to the class and allows for a highly individualized approach to spelling dictation.

However, like so many other language arts software programs on the market, this one devotes the power and appeal of computers to a task that is not only fairly trivial, but pedagogically obsolete. A solid body of language arts research suggests that spelling words are not learned best through drill or dictation. Spelling lists are too far removed from everyday uses of writing, and students therefore do not retain the words for long after the drill. Spelling words are more easily mastered in the process of proofreading a paper containing one's own choice of language. In this respect, having students write papers on computer word processing programs with spelling checkers to guide them in finding errors will prove a superior method of helping students learn to spell.

For a teacher who still drills students through spelling lists, *Talking Speller* might be a useful addition to the classroom software library. But computers in the classroom can be put to much better uses.

(Superior Graphic Software Products, P.O. Box 451, Canton, NC 28716, $19.95 on cassette)

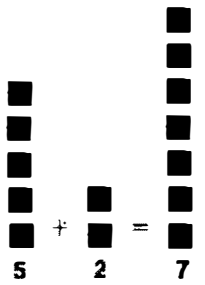
—Stephen N. Tchudi

Mathfact—Educational But Not Exceptional

Mathfact by B5 Software Company has some educational value, but does not utilize the capabilities and power of the CoCo. As an educator, I maintain that the most important aspeet of a program is the ability to convey an idea in a different manner than could be presented in a textbook. *Mathfact* falls short on this point.

Program loading is error free and is configured for a 16K CoCo Extended BASIC. The program is drill and practice in the area of math facts (addition, subtraction, multiplication, division). Two difficulty levels are available for all the fact areas. The first screen is the menu. The user is asked to select the difficulty level and the targeted practice area. It is assumed that the user is old enough to read the menu to choose the proper options. This program must be used with adult supervision for proper implementation as a useful learning tool. The drills are timed and score and time are given at the end of the lesson.

Levelone constitutes a good teaching practice of using the concrete before the abstract is taught. The problems are presented in graphic form as shown in the example.



This provides the concrete practice before the user moves to the more complex levels. This method is standard for the four fact areas.

Level two allows the user to select a number from one to nine. This is the fact that will be practiced. Problems are presented onc at a time and are in page format. If the student misses the problem, the program branches back to the concrete level to allow the user additional practice. The user must correctly answer the problem before continuing in level two. The instructor cannot change the number of problems and there is no consideration for too many incorreet responses in the first level. The user could answer wrong all day.

If users eorrectly answer all questions, they are given the choice to select a game. The game is a number guessing game. The computer picks a number between two limits and the user must guess the number. Hints are given (too high or too low). The user is given only four chances to guess the number. For young children, this is totally inadequatc. A child's reasoning abilities are not sufficiently developed to guess the number in four tries. The suggested grade levels are up to grade five. A fifth grader would have a tough time guessing in four attempts.

After the game is completed, the program branches back to the same drill that was previously selected or a new option may be entered. The program has some major educational flaws that could be corrected to make this a very desirable program for use in the classroom or home.

Report Card

Educational Value = C

Comment—Drill and practice programs are best left for the textbook and worksheet. Not challenging or motivating beyond the normal classroom situation.

Appropriateness = C

Comment—Again, it is appropriate for drill and practice programs to be used, but leave them on paper unless presentation is unique.

Objectives Met = A

Comment—The program meets all the objectives spelled out in the documentation for all areas contained in the program.

Error Handling = B

Comment—Errors are handled satisfactorily in level two. In level one, there is no limit to the amount of incorrect responses. Also, in level one, the user can type in letters or numbers for an answer. If more than a few numbers are entered, the program crashes. Students fooling around could easily bomb the program. The BREAK key is not disabled which can cause problems with smaller children.

Final Grade = B-

Enhancement

The instructor should be able to change the number of problems presented and select the timing option. More graphics and sound would be a nice addition. Limits should be set for incorrect responses. When the limit is reached, the answer should be given. The number of guesses in the game should be increased and the reward for getting the correct answer should be varied and motivating. Good possibilities for an excellent program.

(B5 Software, 1024 Bainbridge Pl., Columbus, OH 43328, $16.95 on tape)

—Rick Cobello

Genealogizers Might Say *Family* Is Great, Great, Grand...

Family is a genealogical data base program for a 32K ECB CoCo with a Line Printer VII. The program uses just about all of CoCo's memory in order to hold data for eight generations. This program has no fancy bells and whistles, but it appears it would be useful for anyone interested in their family tree. The program will allow you to enter the following information about your family's roots: name, date of birth, place of birth, date of marriage, place of marriage, number of children, date of death, place of death, buried at, and occupation. The entries for children and marriage are filed for the husband only to save memory.

The program is menu driven and was easy to use once I figured out the reference number system used to identify each person in the family tree. In addition to creating a file, you can correct previously entered data rather easily by entering the reference number of the file you want to edit and then follow the on-screen prompts.

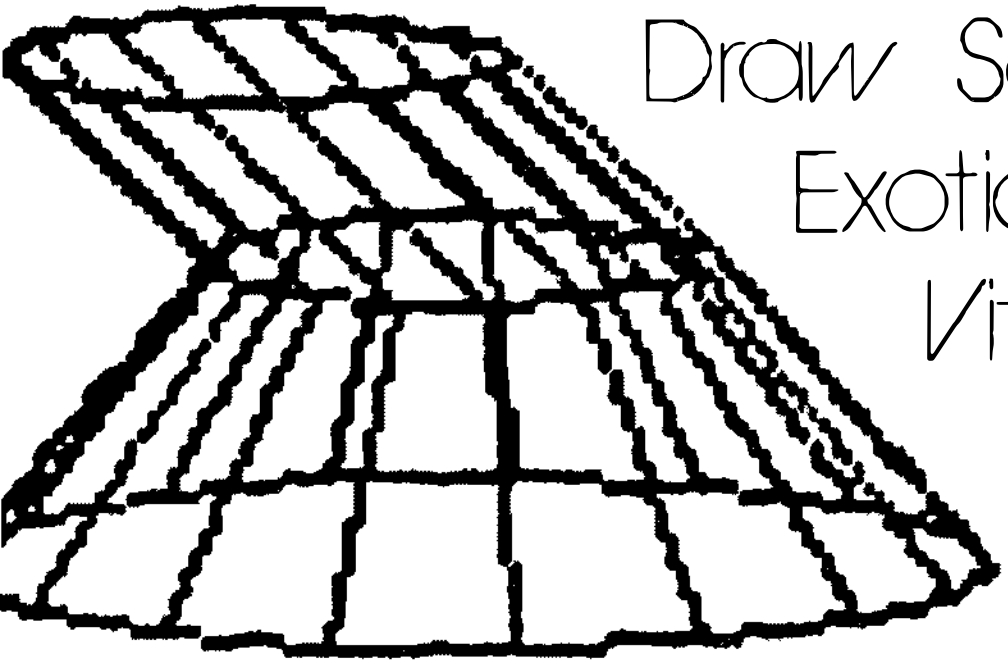
The program requires so much memory it cannot be used with a disk system but the program is written in BASIC so I suppose it could be modified to be compatible with a disk. This, of course, would reduce the number of generations you could enter. In order to list the information to the printer, you must first save the data to tape and then load a second program which will then read your data tape and give you three options for printouts.

Option number one is a pedigree chart. This option will print a chart of any five generations in your file. I found this to be an interesting chart, but could only find information for four generations of my family, so I didn't get full use of the chart. Option number two is a family group chart that prints all available information on a family group (husband and wife). I didn't find this chart as useful as the pedigree chart, but maybe if I knew more about genealogy I could have found a use for it. The last print option is a reference index. This printout will list all the names you have on file and their corresponding reference number. The reference numbers are needed if you want to edit a particular file.

The three pages of documentation are well written and will have you using the program in a matter of minutes. If you are into "family trees" then I think this program could be useful.

(TWM, P.O. Box 232, Lititz, PA 17543, 32K ECB, $9.95)

—Michael Hunt



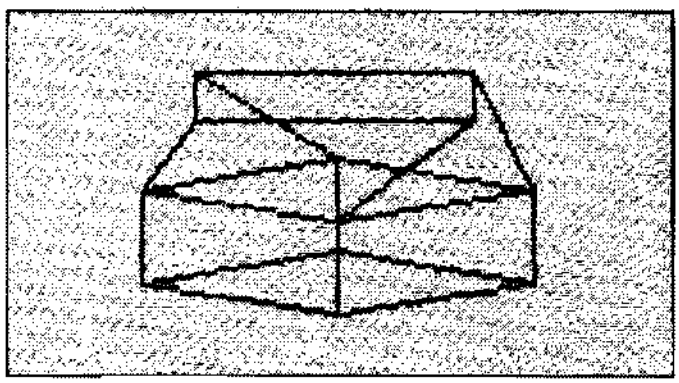
Draw Something Exotic With Vitex93

By Edward Krikorian

Vitex93 is a program which allows the average Color Computer user to create complex images on the 32K BASIC color computer's high resolution graphics screens. The program is completely menu driven. All the user must do is load it and type *RUN*. It is best to clear the screens first. On the first running when the user requests that all screens be cleared, the program will sometimes halt. To correct this problem, simply type the command *RUN* again. At any time a graphic picture is shown, the user can return to the main menu by pressing the space bar.

The program will draw N-sided objects. The more sides an object has, the more that object resembles a circle. The first object or subgroup should be drawn in the "new object" mode. Once this object is completed a new subgroup can be drawn and connected to the one before by using the "connect to previous object" option. You can connect objects only if they contain the same number of sides. Whether the object is at a different angle or not, is unimportant, they will still connect. It is up to the user to determine whether the object will fit on the screen. If the object is too large, the program will halt and report an ?FC Error. Once an image has been drawn, the user can press the Break key and call up a screen dump routine and print the object on a dot matrix printer. Images are not in color, though they produce excellent pictures.

The center of the screen (point 128,96), is considered the point (0,0) by this program. When the user specifies a point, he should keep this in mind. By the way, the point you specify represents the center of the object not the sides. A subgroup is counted each time the user executes the new object or connects to previous object options. For determining angles the best analogy to use is the planet Earth. The equator circle would be a horizontal angle of 0 degrees. A circle drawn from the North Pole to the South Pole would be a horizontal angle of 90 degrees. For vertical angles think



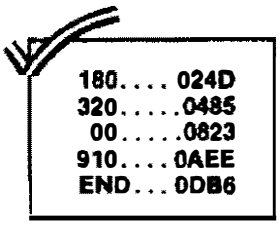
of a cube on a flat table with one side facing you. This represents a vertical angle of 45 degrees. If you turn the cube so that a corner now faces you, you have changed its vertical angle to 0 degree. If you have trouble visualizing these examples, study the sample pictures or try some random input values.

When the "movable objects" mode has been selected the computer will ask for the number of subgroups. It will then prompt you for the pictures data in the usual manner. The only difference being that once all the subgroups are entered the computer will memorize all the moves and draw the object at different horizontal or vertical angles the number of times the user specifies. Each time it will draw your object with the incremented angles. This gives the user a new

(Ed Krikorian is a third year student of computer science and holds an A.A. degree from the University of Florida.)

perspective on what he or she has drawn. As the computer draws, in the movable object mode, it uses the graphic screen that the user is not currently looking at. Once the object has been completed, it will display the other graphic screen and draw on the one you were just viewing. The more subgroups an object contains, the longer it takes the computer to draw them. Have fun and may you draw something exotic.

The listing:




```
10 REM ***VITEX93**********
20 REM ***BY ED KRIKORIAN***
30 REM ***VERSION 3.0*******
40 REM ***DECEMBER 27,1982**
50 PCLEAR 8
60 PI=3.1415926
70 DIM H(200,1),AH(200,1),L(50,6
)
80 CX=128:CY=96
90 CLS
100 CLS:PRINT"MENU":PRINT"0.NEW
OBJECT.":PRINT"1.CONNECT TO PREV
IOUS OBJECT.":PRINT"2.CLEAR SCRE
ENS.":PRINT"3.VIEW SCREENS.":PRI
NT"4.HALT EXECUTION.":PRINT"5.MO
VABLE OBJECTS.":INPUT"YOUR CHOIC
E";D
110 IF D=5 THEN 770
120 IF D=4 THEN PRINT"EXECUTION
HALTED.":END
130 IF D>5 OR D<0 THEN PRINT "IN
VALID COMMAND":GOTO 100
140 PRINT
150 IF D=3 THEN 610
160 IF D=2 THEN INPUT"WHICH SCRE
EN (1,2 OR 3=BOTH)";SW
170 IF SW=1 THEN SW=0:PMODE 4,1:
PCLS:GOTO 100
180 IF SW=2 THEN SW=0:PMODE 4,5:
PCLS:GOTO 100
190 IF SW=3 THEN SW=0:PMODE 4,1:
PCLS:PMODE 4,5:PCLS:GOTO 100
200 PRINT"DEGREES FROM HORIZONTA
L,":INPUT"HIT <ENTER> FOR LAST V
ALUE";TS:IF TS=0 THEN TR=TX ELSE
 TR=TS
210 TX=TR:PRINT
220 TR=PI/(180/TR)
230 PRINT"DEGREES FROM VERTICAL,
":INPUT"HIT <ENTER> FOR LAST VAL
UE";TW:IF TW=0 THEN TY=TU:PRINT:
GOTO 250 ELSE TY=PI/(180/TW)
240 TU=TY:PRINT
250 PRINT"OBJECT'S LOCATION OFFS
ET (X,Y)":INPUT"FROM CENTER OF S
CREEN";XO,YO
260 PRINT
270 INPUT"OBJECT'S RADIUS";R
280 PRINT
290 INPUT"NUMBER OF SIDES";JJ:JJ
=PI/JJ*2
300 PRINT
310 PRINT"IF SAME OBJECT WITH DI
FFERENT":PRINT"ANGLE FROM HORIZO
NTAL IS WANTED":PRINT"ON SCREEN
2, ENTER ITS DEGREES,":INPUT"OTH
ERWISE HIT <ENTER>";AT
320 PRINT
330 IF AT>0 THEN PRINT:PRINT"SEC
OND OBJECT'S DEGREES":INPUT"FROM
 VERTICAL";TE:SF=1:AT=PI/(180/AT
):PA=1:IF TE=0 THEN GOTO 370 ELS
E TE=PI/(180/TE):GOTO 370
340 INPUT"WHICH SCREEN (1 OR 2)"
;PA
350 IF PA=0 THEN PA=1
360 IF PA=2 THEN PA=5
370 PMODE 4,PA:SCREEN 1,0
380 POKE 65495,0:'***DELETE THIS
 LINE IF YOUR SYSTEM DOES NOT HA
VE HIGH SPEED POKE**************
390 REM ***DRAWING ROUTINE****
400 NC=0
410 FOR CN=(0+TY) TO ((2*PI)+TY)
```

```
 STEP JJ
420 Y=INT(R*SIN(CN))
430 X=INT(CX+(R*COS(CN))+SIN(TR)
+XO)
440 Y=CY-((Y*SIN(TR))+YO)
450 PSET(X,Y,1)
460 IF CN<>0+TY THEN LINE(X,Y)-(
OX,OY),PSET ELSE LX=X:LY=Y
470 IF D=0 THEN 500
480 IF SF=0 THEN LINE(H(NC,0),H(
NC,1))-(X,Y),PSET
490 IF SF=1 THEN LINE(AH(NC,0),A
H(NC,1))-(X,Y),PSET
500 IF SF=0 THEN H(NC,0)=X:H(NC,
1)=Y
510 IF SF=1 THEN AH(NC,0)=X:AH(N
C,1)=Y
520 NC=NC+1
530 OX=X:OY=Y
540 NEXT CN
550 LINE(X,Y)-(LX,LY),PSET
560 POKE 65494,126:'***DELETE TH
IS LINE IF YOUR SYSTEM DOES NOT
HAVE HIGH SPEED POKE***********
570 IF SF=1 THEN PA=5:SF=0:TR=AT
:TY=TE:GOTO 370
580 IF QW=1 THEN RETURN
590 IF INKEY$=" " THEN GOTO 100
600 GOTO 590
610 INPUT"VIEW SCREEN (1,2 OR 3=
BOTH)";HJ
620 IF HJ=0 THEN HJ=1
630 IF HJ=2 THEN HJ=5
640 IF HJ=3 THEN 680
650 PMODE 4,HJ:SCREEN 1,0
660 IF INKEY$<>" " THEN 660
670 GOTO 100
680 PMODE 4,1:SCREEN 1,0
690 FOR HH=1 TO 250
700 IF INKEY$=" " THEN 100
710 NEXT HH
720 PMODE 4,5:SCREEN 1,0
730 FOR HH=1 TO 250
740 IF INKEY$=" " THEN 100
750 NEXT HH
760 GOTO 680
770 '***MOVABLE OBJECT ROUTINE**
780 INPUT"NUMBER OF SUBGROUPS";N
L
790 PRINT
800 IF NL>50 THEN PRINT"TOO MANY
 SUBGROUPS":GOTO 780
810 PMODE4,1:PCLS:PA=1:QW=1
820 FOR KE=1 TO NL
830 PRINT"SUBGROUP #";KE;" OF ";
NL
840 PRINT"MENU":PRINT"0.NEW OBJE
CT.":PRINT"1.CONNECT TO PREVIOUS
 OBJECT.":INPUT"YOUR CHOICE";L(K
E,4)
850 PRINT
860 PRINT"OBJECTS OFFSET (X,Y)":
INPUT"FROM CENTER OF SCREEN";L(K
E,0),L(KE,1)
870 PRINT
880 INPUT"RADIUS OF OBJECT";L(KE
,2)
890 PRINT
900 INPUT"NUMBER OF SIDES";L(KE,
3):L(KE,3)=PI/L(KE,3)*2
910 PRINT
920 INPUT"DEGREES FROM HORIZONTA
L";TR:TR=PI/(180/TR):L(KE,5)=TR
930 PRINT
940 INPUT"DEGREES FROM VERTICAL"
;TU:IF TU=0 THEN TY=0:L(KE,6)=0:
PRINT:GOTO 950 ELSE TY=PI/(180/T
U):L(KE,6)=TY
950 D=L(KE,4):JJ=L(KE,3):R=L(KE,
2):XO=L(KE,0):YO=L(KE,1)
960 GOSUB 370
970 A$=INKEY$:IF A$<>" "THEN 970
980 NEXT KE
990 INPUT"NUMBER OF MOVEMENTS";N
T
1000 INPUT"DEGREES OF HORIZONTAL
 MOVEMENT";DR
1010 INPUT"DEGREES OF VERTICAL M
OVEMENT";VR
1020 TR=0
1030 IF DR=0 THEN 1040 ELSE DR=P
I/(180/DR)
1040 QW=1:PA=5
1050 PCLS:PMODE 4,1:PCLS:SCREEN
1,0
1060 FOR BB=1 TO NT
1070 TZ=TZ+DR:VZ=VZ+VR:SCREEN 1,
0:IF PA=1 THEN PA=5 ELSE PA=1
1080 PMODE 4,PA:PCLS
1090 FOR SS=1 TO NL
1100 D=L(SS,4):XO=L(SS,0):YO=L(S
S,1):R=L(SS,2):JJ=L(SS,3):TR=L(S
S,5)+TZ:TY=L(SS,6)+VZ
1110 PMODE 4,PA:GOSUB 380
1120 NEXT SS
1130 NEXT BB
1140 SCREEN 1,0
1150 A$=INKEY$:IF A$=" " THEN 90
 ELSE GOTO 1150
```

Hint . . .

No-List *POKE*

If you would like to keep your program from listing, use *POKE* 383,158. To return to normal listing, type *POKE* 383,0.

—Ken Ostrer

Hidden BASIC Is Pirates' Peeve

"Piracy" is much more common today than in the 16th century when the term originated. Whether it is an otherwise honest computer hobbyist sharing a good game with a friend, or a very dishonest dealer stocking his shelves with illegal copies, software piracy is a recognized industry-wide problem. The number of "doubloons" lost through illegal copying is enough to discourage many from entering the software market. This has forced software distributors into building copy protection into their products. There are coding techniques which offer a measure of protection for programs written in machine code, but there has been little copy protection available for programs written in Color Computer BASIC. The reason for this is simple. The philosophy behind the design of the BASIC operating system used in CoCo is that of simplicity, simplicity to the point of allowing accessibility to even a young child. In doing so, much flexibility with respect to copy protection is compromised.

Hidden BASIC 1.0 is a new tool marketed by Spectrum Projects which can help to protect a BASIC program from being copied. This is a machine language program that is appended to a BASIC program that you would like to protect. The resulting composite program, which is loaded as a machine language program, disables several commands normally found in BASIC. These are commands such as *LIST, LLIST, CSAVE, CSAVEM, EDIT, CLOAD, CLOADM,* and *TRON* and *TROFF*, which presumably could allow either direct duplication or listings of BASIC code which could be entered manually.

Hidden BASIC is very easy to use. It is loaded, the program to be protected is loaded, then the new program is saved automatically to tape. When loading, the protected program is treated as a machine language program, ie, *CLOADM*, but it is *RUN* not *EXECuted*. The protection provided for with this program is very effective.

Not mentioned in the advertising or the instructions, is that this program also appears to protect against copying with Disk BASIC. The *SAVE, SAVEM, LOAD* and *LOADM* commands do not work, therefore, a program cannot be taken apart by the add power allowed with the disk operating system. It is important to note that this program is sold only in a format designed to operate with cassette storage.

I recommend *Hidden BASIC* to anyone who has reason to protect a BASIC program. It is straightforward, easy to use, and works as advertised. There is the small matter of a royalty agreement with Spectrum Projects if you market a program using *Hidden BASIC* for protection. I called Spectrum to obtain details relating to royalty agreements on their "voice line," but only talked to a friendly tape recorder waiting to take my order. If you have any questions you should perhaps write or contact them via the data communications number given on the instruction sheet.

(Spectrum Projects, 93-15 86th Drive, Woodhaven, NY 11421, $19.95 on cassette for BASIC or Extended Color BASIC)

—Tom Szlucha

Spell-Rite Makes Cassette Spelling Right

As an adjunct to a cassette-based word processing system, *Spell-Rite* has so much promise that it could take up all your time for a month and still have untapped applications. Yet using *Spell-Rite* can be so frustrating that you'll want to chuck it out of the window after two hours.

The program, written by Darrel Wright and marketed by Eigen Systems of Austin, Tex., is a spelling verifier for text files. It is intended for use in a cassette system in conjunction with a word processing program that will generate ASCII files.

As shipped, *Spell-Rite* includes a ten thousand word dictionary that may be changed by addition or deletion. The program also allows the user to generate any number of specialized dictionaries, each containing up to 10,400 words. To a technical writer, that function alone would be worth the price of admission. The dictionaries are dynamic, in that each file checked can be used to add to the dictionary. In fact, *Spell Rite* can create new dictionaries merely by reading text files. The only limitations are that the words entered be between two and 18 characters long and contain no punctuation other than hyphens and apostrophes.

In use, *Spell-Rite* is strictly menu-driven, requiring one-key responses except when it asks for filenames. The machine language program returns to the main menu at any default, including a reset. The original purchase includes a backup copy just in case the first one crashes. The program carries a 90-day warranty as well. It requires 32K memory and Extended Color BASIC.

The user has four options on start-up. He may elect to get a hard copy of the program's actions, which records the line number, the word position, the questionable word and the action taken with it, or this can be displayed on screen as an option.

There is also a choice in verification mode. In the manual mode, the user must choose what to do with each questionable word—to correct it if it's not in the dictionary, to add it to the dictionary, or to ignore it. In the auto-verify mode, the computer merely indicates "Unsure" for each questionable word and goes on.

There's a catch to all this. The computer doesn't actually correct spelling. All it does is point out words that it doesn't have in its dictionary, leaving any correction up to the user. In effect, it's a typo-catcher rather than a spelling corrector. It's not a substitute for a good conventional dictionary.

Spell-Rite includes a host of other features, such as a way to bypass embedded codes or any other set of characters, a dictionary word count and sort and an index sorter that is actually a separate, bonus program.

I used *Spell-Rite* with *Telewriter-64*, after spending the better part of a day trying to figure out how to make the blasted thing save a file in ASCII format. As a last resort, I turned to the *Telewriter* manual and read the section that tells how to do it.

After saving several existing files in ASCII format, I loaded *Spell-Rite*, then followed its simple menus to load the basic dictionary that comes with it. Then I went to the menu to verify spelling and hit a very high frustration peak.

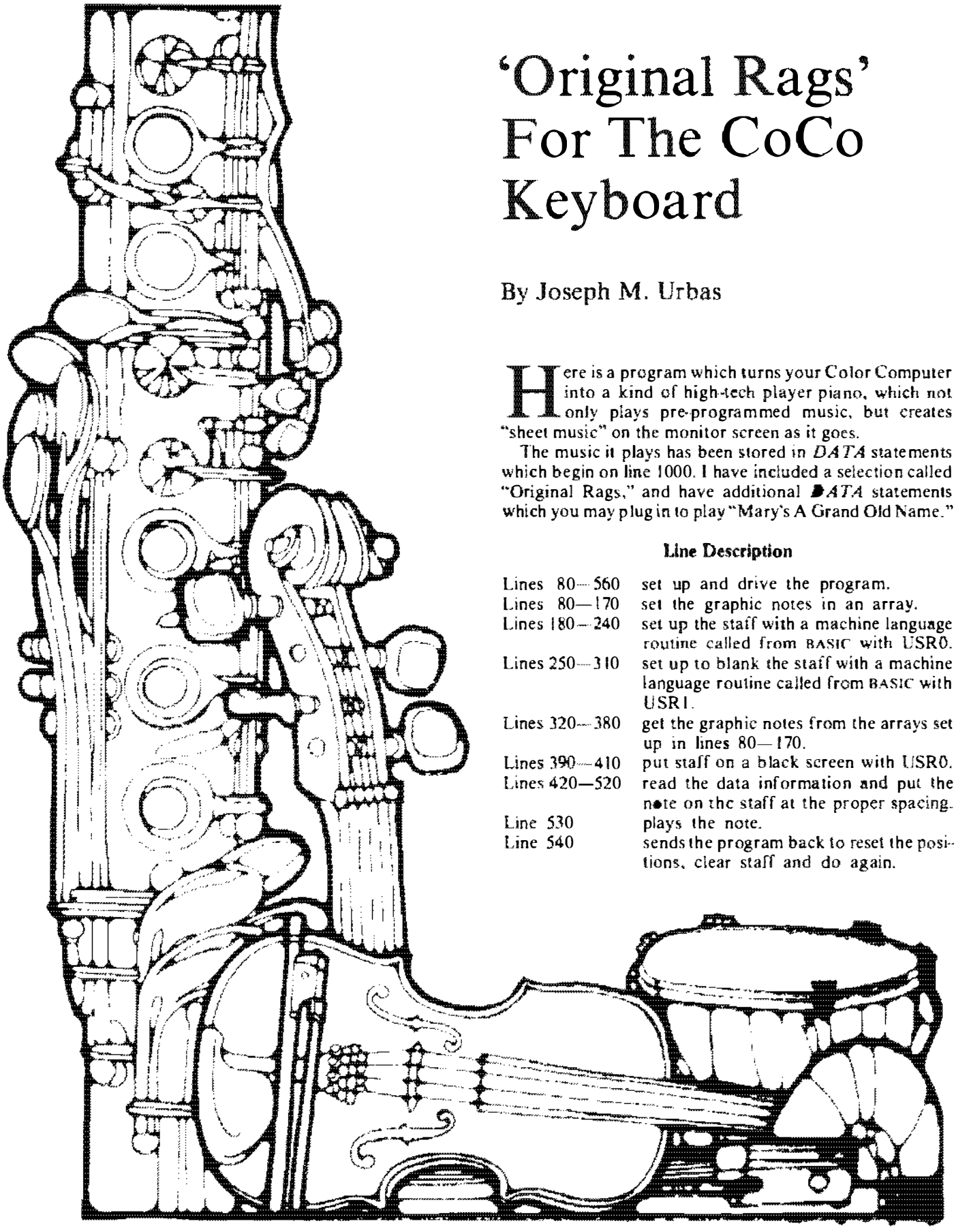
Spell-Rite works by reading in a line of text from the tape, checking each word in it against the dictionary, then loading another line. The cassette recorder starts and stops once for every line of text. My CoCo has a real fondness for the old I/O ERR●R and is especially finicky about the start signal. Ergo, despite verifying the save with *Telewriter*, I was often shunted to the section of *Spell-Rite* that politely informed me that I had a tape error. After much gnashing of teeth and pounding the desktop, I managed to get a pretty good idea of what *Spell-Rite* is all about.

The cassette error problem is not a fault in *Spell-Rite*. It's a hardware problem that seems to be a fact of life with the relay that sends the start and stop signals to the cassette recorder. If that doesn't function correctly, *Spell-Rite* is useless.

I'm intrigued by *Spell-Rite*. It's well written, has very good documentation in the form of a 19-page manual and it has lots of room for experimentation. On the negative side, the program's reliance on the CoCo cassette relay gives it a vulnerability that can lead to frustration.

(Eigen Systems, P.O. Box 180006, Austin, TX 78718, $59.95 on tape)

—Bruce L. Sublett



'Original Rags' For The CoCo Keyboard

By Joseph M. Urbas

Here is a program which turns your Color Computer into a kind of high-tech player piano, which not only plays pre-programmed music, but creates "sheet music" on the monitor screen as it goes.

The music it plays has been stored in *DATA* statements which begin on line 1000. I have included a selection called "Original Rags," and have additional *DATA* statements which you may plug in to play "Mary's A Grand Old Name."

Line Description

Lines 80—560	set up and drive the program.
Lines 80—170	set the graphic notes in an array.
Lines 180—240	set up the staff with a machine language routine called from BASIC with USR0.
Lines 250—310	set up to blank the staff with a machine language routine called from BASIC with USR1.
Lines 320—380	get the graphic notes from the arrays set up in lines 80—170.
Lines 390—410	put staff on a black screen with USR0.
Lines 420—520	read the data information and put the note on the staff at the proper spacing.
Line 530	plays the note.
Line 540	sends the program back to reset the positions, clear staff and do again.

Line 550	is a timer between notes to offset the time it takes to clear the staff.
Line 560	clears the staff at the end of the music.
Lines 570—590	do again.

For each note played, data is put into *DATA* statements in groups of three. . .

1) Graphic note (1—7)
2) Vertical position of the note (73—112)
3) Audio note (A-,A,B-,B,C,C+,D,E-,E,F,F+G)

Graphic Notes

1) Half note, stem up.
2) Quarter note, stem up.
3) Sixteenth note, stem up.
4) Half note, stem down.
5) Quarter note, stem down.
6) Sixteenth note, stem down.
7) Whole note.

Vertical Positions

73--B
76--A
79--G
82--F
85--E
88--D
91--C
94--B
97--A
100--G
103--F
106--E
109--D
112--C

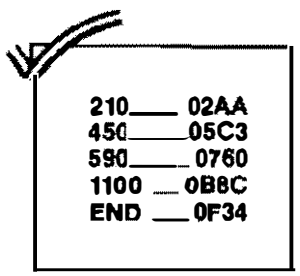
Audio Notes

A-,A,B-,C,C+,D,E-,E,F,F+,G

They work exactly like the *PLAY* statement for the Color Computer with Extended Color BASIC.

Hints

- After your musical selection has been entered into your *DATA* statements, add ,8,8,8 to the end of the last *DATA* statement. This will end your musical selection.
- Keep your *DATA* statements short to make it easier to find a mistake when you make one.
- Sheet music makes putting musical selections in fairly easy.
- The graphic notes are for visual effect, so dotted notes are ignored but use the proper audio note length.
- Start your *DATA* statements with line 1000, so as not to interfere with the driver portion of the program.
- Take a little time to study my *DATA* statement lines.



The listing:

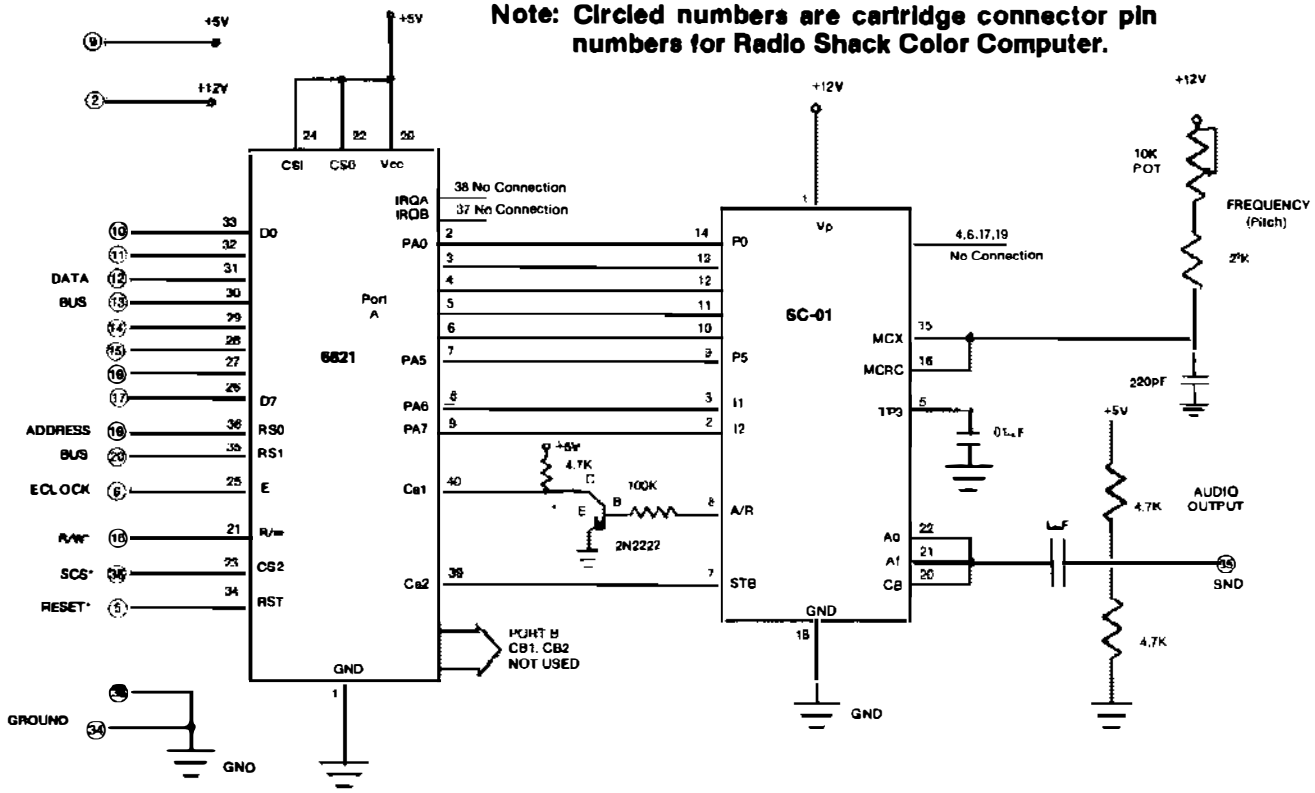
```
10 '     SEE AND HEAR MUSIC
20 '*****************************
30 '*      JOSEPH M. URBAS      *
40 '*        107 VINE ST        *
50 '*  CANONSBURG, PA. 15317    *
60 '*      COPYRIGHT, 1983      *
70 '*****************************
80 CLEAR500
90 DIMA(10),B(10),C(10),D(10),E(
10),F(10),G(10):CLS
100 PMODE4,1:PCLS
110 CIRCLE(4,14),4,,.75:LINE(8,1
4)-(8,0),PSET
120 CIRCLE(19,14),4,,.75:PAINT(1
9,14),5,5:LINE(23,14)-(23,0),PSE
T
130 CIRCLE(34,14),4,,.75:PAINT(3
4,14),5,5:LINE(38,14)-(38,0),PSE
T:LINE(38,0)-(44,3),PSET
140 CIRCLE(49,14),4,,.75:LINE(45
,14)-(45,28),PSET
150 CIRCLE(64,14),4,,.75:PAINT(6
4,14),5,5:LINE(60,14)-(60,28),PS
ET
160 CIRCLE(79,14),4,,.75:PAINT(7
9,14),5,5:LINE(75,14)-(75,28),PS
ET:LINE(75,28)-(81,25),PSET
170 CIRCLE(94,14),4,,.75
180 FORL=16000TO16052
190 READC
200 POKEL,C
210 NEXTL
220 DEFUSR0=16000
230 DATA134,255,142,18,1,167,128
,140,18,31,38,249,142,18,193,167
,128,140,18,223,38,249,142
240 DATA19,129,167,128,140,19,15
9,38,249,142,20,65,167,128,140,2
0,95,38,249,142,21,1,167,128,140
,21,31,38,249,57
250 FORL=15937TO15999
260 READC
270 POKEL,C
280 NEXTL
290 DEFUSR1=15937
300 DATA134,0,142,15,1,167,128,1
40,18,0,38,249,142,18,33,167,128
,140,18,192,38,249,142,18,225,16
7,128,140,19,128,38,249
310 DATA142,19,161,167,128,140,2
0,64,38,249,142,20,97,167,128,14
0,21,0,38,249,142,21,33,167,128,
140,22,160,38,249,57
320 GET(0,0)-(14,28),A,B
330 GET(15,0)-(29,28),B,G
340 GET(30,0)-(44,28),C,G
350 GET(45,0)-(59,28),D,G
360 GET(60,0)-(74,28),E,G
370 GET(75,0)-(89,28),F,G
380 GET(90,0)-(104,28),G,G
390 PMODE4,1:PCLS:SCREEN1,1
400 A=USR(0)
410 FORM=1TO500:NEXTM
420 Q=0:P=20
430 B=USR1(0)
440 READN,V,N$
450 ON N GOTO460,470,480,490,500
,510,520,560
460 PUT(P,V)-(P+14,V+28),A,OR:Q=
```

```
54:GOTO530
470 PUT(P,V)-(P+14,V+28),B,OR:Q=
36:GOTO530
480 PUT(P,V)-(P+14,V+28),C,OR:Q=
18:GOTO530
490 PUT(P,V)-(P+14,V+28),D,OR:Q=
54:GOTO530
500 PUT(P,V)-(P+14,V+28),E,OR:Q=
36:GOTO530
510 PUT(P,V)-(P+14,V+28),F,OR:Q=
18:GOTO530
520 PUT(P,V)-(P+14,V+28),G,OR:Q=
72:GOTO530
530 PLAYN$
540 P=P+Q:IFP=>236BOTO420
550 FORM=1TO10:NEXTM:GOTO440
560 B=USR1(0):FORM=1TO1000:NEXTM
:CLS
570 PRINT@259,"PRESS ENTER TO PL
AY AGAIN."
580 IF INKEY$<>CHR$(13)GOTO580
590 RUN
1000 '*****ORIGINAL RAGS*****
1010 DATA6,88,T504L8D,6,100,03G,
6,94,B,5,88,04L4D,6,94,03L8B,6,8
8,04D,6,94,03B,6,88,04D,6,100,03
G,6,94,B,5,88,04L4D,6,94,03L8B
1020 DATA6,88,04D,6,94,03B,6,85,
04E,5,85,L4E,3,100,03L88,3,97,A,
3,100,G,2,97,L4A,4,94,L2BP8,6,10
0,L8G,6,94,B,6,91,04C
1030 DATA6,88,04L8D,6,100,038,6,
94,B,5,88,04L4D,6,94,03L8B,6,88,
04D,6,94,03B,6,88,04D,6,100,03G,
6,94,B,5,88,04L4D,6,94,03L8B
1040 DATA6,88,04D,6,94,03B,6,94,
L4B,1,97,L2A,3,100,L8G+,3,97,A,6
,94,B,2,97,L4A,3,103,L8F+,2,106,
L4E,2,109,D
1050 DATA6,88,04L8D,6,100,03G,6,
94,B,5,88,04L4D,6,94,03L8B,6,88,
04D,6,94,03B,6,88,04D,6,100,03G,
6,94,B,5,88,04L4D,6,94,03L8B
1060 DATA6,88,04D,6,94,03B,6,85,
04E,6,85,L4E,3,100,03L8G,3,97,A,
3,100,G,3,97,L4A,4,94,L2.B,5,94,
L4B
1070 DATA6,85,04L8E,5,88,L4D,6,9
4,03L8B,5,91,04L4C,5,94,03B,6,94
,L8B,2,97,L4A,3,106,L8E,5,94,L4B
1080 DATA2,97,A,6,94,L8B,2,97,L4
A,6,94,L8B,6,91,04C,6,94,03B,2,9
7,L4A,1,100,L2G,5,79,04L4GP4
1090 DATA6,76,04L8A+,5,73,L4B,6,
76,L8A+,5,73,L4B,6,76,L8A+,6,73,
B,6,82,F+,5,79,L4G,4,85,L2E,6,82
,L8F+
1100 DATA5,79,L4G,6,82,L8F+,5,79
,L48,6,82,L8F+,6,79,8,6,91,C+,5,
88,L4D,4,94,03L3.B
1110 DATA6,91,04L8C+,5,88,L4D,6,
91,L8C+,5,88,L4D,6,91,L8C+,6,88,
D,6,88,D+,5,85,L4E,1,97,03L2A,6,
88,04L8D+
1120 DATA5,85,L4E,6,88,L8D+,5,85
,L4E,6,88,L8D+,6,85,E,6,85,E+,5,
82,L4F+,4,88,L3.D
1130 DATA6,76,04L8A+,5,73,L4B,6,
76,L8A+,5,73,L4B,6,76,L8A+,6,73,
B,6,82,F+,5,79,L48,4,85,L3E
1140 DATA6,82,L8F+,5,79,L4G,6,82
,L8F+,5,79,L48,6,82,L8F+,6,79,G,
6,91,C+,5,88,L4D,1,94,03L3B
1150 DATA3,97,03L8A,2,97,L4A,3,9
7,L8A,5,85,04L4E,5,85,E,6,88,L8D
,5,79,L4G,6,85,L8E,6,88,D
1160 DATA6,94,03B,2,100,L4G,6,97
,L8A,6,94,B,6,91,04C,5,94,03L4B,
3,100,L8G,2,97,L4A,7,100,L2GP4
1170 DATA6,88,04L8D,6,100,03G,6,
94,B,5,88,04L4D,6,94,03L8B,6,88,
04D,6,94,03B,6,88,04D,6,100,038,
6,94,B,5,88,04L4D
1180 DATA6,94,03L8B,6,88,04D,6,9
4,03B,6,85,04E,5,85,L4E,3,100,03
L8G,3,97,A,3,100,8,2,97,L4A,4,94
,L2.B,5,94,L4B
1190 DATA6,85,04L8E,5,88,L4D,6,9
4,03L8B,5,91,04L4C,5,94,03B,6,94
,L8B,2,97,L4A,3,106,L8E,5,94,L4B
1200 DATA2,97,A,6,94,L8B,2,97,L4
A,6,94,L8B,6,91,04C,6,94,03B,2,9
7,L4A,1,100,L2G,5,79,04GP4,8,8,8
```

Mary's A Grand Old Name

```
1000 'MARY'S A GRAND OLD NAME
1010 DATA3,109,T403L16D,3,100,L8
.8,6,94,L16B,4,88,04L2D,4,94,03B
,1,97,A,1,106,E,3,103,L8.F+,3,10
0,L16G,3,97,L8.A,3,103,L16F+,2,1
09,L4D,2,103,F+,1,106,L2.E,3,109
,L16D,3,106,L8.E,3,103,L16F+
1020 DATA2,97,L4A,2,100,G,2,103,
F+,2,106,E,2,103,F+,2,100,G,2,97
,A,2,109,D,1,97,L2A,1,97,A+,4,94
,L2.B,3,109,L16D,3,100,L8.8,6,94
,L16B,4,88,04L2D,4,94,03B
1030 DATA1,97,A,1,106,E,3,103,L8
.F+,3,100,L16G,3,97,L8.A,3,103,L
16F+,2,109,L4D,2,103,F+,1,106,L2
.E,3,109,L16D,3,100,L8.G,6,94,L1
6B,5,88,04L4D,5,88,D+,5,85,L4.E,
6,94,03L8B
1040 DATA5,85,04L4E,5,88,D,5,91,
C,6,94,03L8.B,3,97,L16A,1,106,L2
.E,1,103,F+,1,100,L2..GP4,8,8,8
```

CORRECTIONS



In John R. Kelty's "A Cheap Talker For Our CoCo"(July 1983, page 276), the accompanying schematic had a few errors. Writes Kelty, "Most readers would notice the pin errors on the 6821. It would not really matter if these were reversed anyway, except that the inflections would be inverted and the speech might sound like Chinese." Check pins 8 and 9 on the revised schematic and also note that the 4.7K ohm resistor at CA1 should be "pulled up," +5VDC (not +3). The series resistor with the 10K pot is 2K and, finally, the clock capacitor of 220 pF to ground was left off. We had a number of calls about these mistakes and regret any inconvenience they caused. The corrected schematic is reproduced for your reference.

Submitting Material To the Rainbow

Contributions to *the RAINBOW* are welcome from everyone. We like to run a variety of programs which will be useful/helpful/fun for other CoCo owners.

Program submissions must be on tape or disk and it is best to make several saves, at least one of them in ASCII format. We're sorry, but we do not have time to key in programs. All programs should be supported by some editorial commentary, explaining how the program works. We're much more interested in how your submission works and runs than how you developed it. Programs should be learning experiences.

We do pay for submissions, based on a number of criteria. Those wishing remuneration should *so state* when making submissions.

For the benefit of those who wish more detailed information on making submissions, please send a SASE to: Submissions Editor, *the RAINBOW*, P.O. Box 209, Prospect, KY 40059. We will send you some more comprehensive guidelines.

Please do not submit programs or articles currently submitted to another publication.

Apparently due to an error in paste-up at *the Rainbow*, the first segment of line 10 was left out of the listing for Mike Hall's "Trailin'Tail"(August 1983, page 36). That line, in its entirety, is:

```
10 CLS:DIMCH$(43),X(99),Y(99),A(
4),B(4),E(10),S(11),S$(11),G(50)
:FORZ=1TO43:READCH$(Z):PLAY"L255
;ABA":NEXT:FORZ=1TO10:S(Z)=0:S$(
Z)="    ":NEXT
```

Joel Robbins reports that in his *Autodesigner* listing in "Talk About Your Chopped Down Ford!" (August 1983, page 50), there "were some 'ghosts' from an earlier version that didn't affect the running of the program, but might appear as real mysteries because they are useless commands." To "clean up" the listing, delete lines 103, 106, 107, 122 and 123. Change line 121 to read: 121 RETURN and that should do it.

Ken Ostrer's "Treasures of the Enrakian Empire," (August 1983, page 122), needs a *POKE25,6: NEW* prior to loading in order to free up enough memory. We apologize for not noting this in the article.

Don't ask us how it happened, but a colon was omitted from the end of a line in the listing for Jim Schmidt's "Everything You Wanted To Know About Hex. . ." (April 1983, page 186). Line 16030 should properly read:

```
16030 N1$=HEX$(N1):N2$=HEX$(N2):
A1$=HEX$(N1-N2)
```

RECEIVED & CERTIFIED

The following products have been recently received by *the Rainbow*, examined by our magazine staff and approved for the *Rainbow Seal of Certification*, your assurance that we have seen the product and have ascertained that it is what it purports to be.

This month the *Seal of Certification* has been issued to:

Time Teacher, a 16K ECB educational program for ages 5-11 with six skill levels. It reinforces and assists children in learning to tell time by having them convert the time displayed on a traditional clock into digits using the keyboard. Includes two practice modes, two quiz modes and a detailed progress report with the student's name that can be displayed to screen or printer. Crystal Software, 6591 Dawsey Road, Rock Creek, OH 44084, tape $15.95.

Trekbest, a 16K ECB Trek game. Objective: You are in command of a prototype destroyer with the mission of stopping the Klystar from successfully invading your home space. Hyperion Software, P.O. Box 196, Lagrangeville, NY 12540, tape $16.95.

Babylon, a 16K game with ten skill levels. Objective: You are governor of Babylon and in the best interest of its citizens, you must manage the city's assets; land, grain, and the people themselves (as sources for farmers and as members of the city's defensive army) while contending with enemy invasion, poor crops, plague, and other disasters that arise. Valhalla Enterprises, P.O. Box 243, Sumner, WA 98390, tape $12.50.

Electronic Graph, a 16K ECB function graphing program that will graph any function using low, medium or high-resolution graphics and overlay as many of the functions on a graph as you wish using a LPVII or look-alike printer. Valhalla Enterprises, P.O. Box 243, Sumner, WA 98390, tape $14.56.

Great New Adventures In Wonderland, a 32K 100 percent ML adventure, fantasy world game peopled with the creatures of Lewis Carroll's imagination, "Alice In Wonderland," "Through The Looking Glass," and "The Hunting of the Snark" blended into one landscape. Objective: You are Alice. As you wander through Wonderland, you must become a queen (on the chessboard landscape), help capture the nasty Snark, and get home again. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, tape $24.95, disk $29.95.

TIMSMAIL, Tape Information Management Systems Mailing List, a 16K Extended BASIC modification of TIMS (a generalized data management system) to address the special requirements of label printing that cannot be handled by the more generalized TIMS print mode. TIMSMAIL provides eight fields selected in such a way as to facilitate searching, editing, sorting and printing the mailing list. Includes a 15-page user manual. Sugar Software, 2153 Leah Lane, Reynoldsburg, OH 43068, tape $19.95.

Shaft, a 16K 100 percent ML arcade game. Objective: Maneuver your man across the screen while avoiding the eight crushing elevators, as they move up and down in one of several random patterns, and try to reach the next floor. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, tape $24.95, disk $29.95.

Topsy Turvy, a 16K ECB educational word game for one to six players. Objective: Play against your opponent and form as many three or more letter words as you can from the ten letters chosen at random by the computer. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, tape $19.95, disk $24.95.

Disk Manager, a 16/32K disk utility program that provides all the facilities needed to manage disk files. It will date files on the disk directory; print a directory that includes file size; creation date; granule locations; and machine language program start, end, and exec locations. Disk Manager will also create and save a catalog of all the files on a collection of disks, find a particular file or group of files in the catalog, restore killed files, completely recover a disk with a damaged directory and most output from the program can be sent to the screen, printer or both. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, tape $29.95.

Music Box, a 16K ML program that lets you compose (or type in from sheet music) a song in four-part harmony, assign a different instrument to each voice, or use one instrument for several parts. Includes a song by Bach and instructions. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, tape $24.95, disk $29.95.

Disk Zapper, a 32/64K programmer's aid, disk-fixer, editor and formatting utility. It can be used to protect disks, change the information on disks, and backup many disks that have been altered to make backup impossible. Prickly-Pear Software, 9234 E. 30th Street, Tucson, AZ 85710, disk $34.95.

CoCo Mail, a 16K menu oriented, mailing list system to be used to maintain a list of up to 200 files per disk, containing name, address, city, state, zip, phone numbers, special code, and notes with a maximum of 256 characters per file. The special codes record of each file allows the system user to mark each record for specialized printouts. Files can be moved from one disk to another and prints two across or single labels. Includes a 15-page instruction manual. Reitz Computer and Electronics, 3170 W. Central Avenue, Toledo, OH 43606, disk $59.95.

Graphic Screen Print Program, a 16K ML graphic screen print program that works in all PMODES and lets you shift screen images anywhere on the printed page. Available for both Color BASIC 1.0 and 1.1. Custom Software Engineering, 807 Minutemen Causeway, Cocoa Beach, FL 32931, tape $9.95.

Statement Writer, a 32K program that supplements the capabilities of the *Disk Double Entry (DDE)* program and allows you to select certain *DDE* accounts for special treatment, add supplemental information and format the ledgers for these accounts to satisfy special needs. The program will take receivable accounts, add full customer name and address, and produce statements suitable for billing. **Statement Writer** will also produce summary reports for selected accounts to provide overall visibility of account status and activity. The user should be familiar with the concepts and operation of *DDE* before attempting to use this program. Custom Software Engineering, 807 Minutemen Causeway, Cocoa Beach, FL 32931, tape $34.95.

Data-500, a 16K menu-driven data base utility program that will store up to 3,000 characters on one file and print to screen or printer. Litefrac Software, Inc., 2002 N. Magnolia, Tuscon, AZ 85712, tape $12.

Cosmic Clones, a 16K ML arcade war game. Objective: Achieve the highest score possible by eliminating the Clonial invasion forces while protecting your starbase fuel cells. Mark Data Products, 24001 Alicia Parkway, No. 226, Mission Viejo, CA 92691, tape $24.95.

El Bandito, a 16K ML game for one or two players with eight skill levels. Objective: Maneuver your bandits through passages and into tunnels while dodging menacing spiders and returning bounty to your lair. Mark Data Products, 24001 Alicia Parkway, No. 226, Mission Viejo, CA 92691, tape $24.95.

Flexi-Filer, a 32K data base management system for applications such as mailing lists, inventory, sales records, expense records, etc., with many features that allow you to customize the system. **Flexi-Filer** will collect up to 35 elements per record, with up to 240 characters per record. All or any selected subset or records can be sorted in ascending or descending order by any of the 35 fields. Includes a 32-page instruction manual and requires an 80 column printer. Computerware, Box 668, Encinitas, CA 92024, disk $64.95.

Micromon, a 4K ML monitor designed to allow direct access to the 6803 microprocessor, the hear of the Radio Shack TRS-80, Model MC-10. It contains the necessary tools to: examine the contents of the MC-10 operating system ROM and discover many of the routines contained therein; examine and change contents of memory in RAM based programs; save and load ML programs using assette tape; create your own ML programs; use breakpoints and register displays for debugging and testing ML programs, routines; and dumps listings or memory contents to printer for hardcopy ecording. Includes a 9-page instruction manual. Micro Ten Software Company, 495 Amboy Avenue, Perth Amboy, NJ 08861, tape $19.95.

Spelling Bee, a 16K ECB word game for one or two players of ages seven to adult, using high-resolution animated graphics. Objective: Help Kay Bee find the letters that make up the word hidden in the flower by inputting letters using the keyboard. The game comes complete with a file of 260 words, three game variations and two skill levels. Players can also vary the difficulty by creating word lists that can be saved on tape. Crystal Software, 6591 Dawsey Road, Rock Creek, OH 44084, tape $15.95.

Mathmenu, a 16/32K menu-driven system turns your CoCo into a flexible tool for mathematics and engineering by manipulating matrices and vectors, performing integration and differentiation, solving quadratic equations, and plotting user defined functions. Includes a 22-page reference manual. Inter Action, 113 Ward Street, Department F, New Haven, CT 06519, tape $44.95, disk $49.95.

Bumpers, a 16K ML maze game for one or two players. Objective: Race your opponent through a randomly generated maze with invisible walls, (until the player hits them) while avoiding booby traps and setting new traps for your opponent, as you try to be the first to complete the maze. Mark Data Products, 24001 Alicia Parkway, No. 226, Mission Viejo, CA 92691, tape $24.95.

Glaxxons, a 16K super hi-res, ML space game for one or two players with seven skill levels. Objective: Achieve the highest score by eliminating as many swooping and diving, enemy attacking space crafts as possible, while avoiding your own destruction. Mark Data Products, 24001 Alicia Parkway, No. 226, Mission Viejo, CA 92691, tape $24.95.

Recall, a 16K utility program that will recover material lost to the I/O ERROR in a casse te crash. It reeovers both BASIC and machine language. Time Software, P.O. Box 1014, Clearfield, VT 84015, tape $14.95.

Disk Graphics, a 16K ECB program that allows you to plot a variety of char s and display them on a screen or print them on a graphics printer. With its multifaceted, chart-plotting capabilities, **Disk Graphics** has many practical, business, scientific, and engineering applications. Radio Shack Stores nationwide, disk $49.95.

Amortization, a 16K ECB program designed to provide you with complete information concerning simple interest loans. The program will print the following on the screen or printer: a running total of principal still to be paid; amount of monthly payment applied to the principal balance; amount of monthly payment applied to interest; the total monthly payment; the total amount paid into principal to date; the total of interest paid to date; and the total amount paid to date. Harmonyes, P.O. Box 1573, Salt Lake City, UT 84110, tape $11.95.

C.C. Dialer, a 16K automatic telephone dialing program that outputs telephone digits through your television speaker. You can enter and save directories on tape and each directory will contain up to 196 names and numbers. Requires ECB and touch-tone phone service. Chris Computers, 6299 Alderwood Lane, Delta, B.C., Canada V4E 3E7, tape $29.95.

The Cheap Talk r, a 16/32K ECB speech synthesis system that comes fully assembled and tested. Includes cartridge with s cketed SC-01, program and users manual. Kelty Engine ring, 1440 N. 61st, Lincoln, NE 68505, tape $80. Printed circuit board for Cheap Talker $15.

INTBASIC 1.0, a 16/32/64K interactive, self-contained integer BASIC compiler. The compiler was designed to optimize program space and memory requirements, while still maintaining syntax structures used in Color Basic. **INTBASIC 1.0** breaks a BASIC program into three areas within memory, with the variable storage area occupying the fourth. The four areas of mem ory are used for: main program storage, text table storage (print text), subroutine storage and variable storage. Wasatchware, P.O. Box 510371, Salt Lake City, UT 84151, tape $39.95.

Zeus, a 16K ML arcade game. Objective: You are the mighty wizard and you must fight off the thunderbolts from the Thunder God—Zeus. As the Cone of Cold destroys the thunderbolts, you must use your shield for protection. Aardvark, L.T.D., 2352 S. Commerce, Walled Lake, MI 48088, tape $19.95.

Cassette Directory, a 4K program that will read a tape cassette and print a directory of each file found on the cassette. CoCoPro, P.O. Box 37022, St. Louis, MO 63141, tape $2.50.

Graphic Math Adventure, a 32K ECB educational adventure game in 24 skill levels. Objec ive: Search for treasure on land, river and in caves. Your search is blocked by many obstacles which can be overcome by correctly answering math problems of addition, subtraction, multiplication and division. Software Factory, 1333 Morgan Road, Bremerton, WA 98312, tape $21.95.

Statgraf, a 32K ECB, menu-driven, linear regression analysis package combined with a high resolution plotting/li e graphing system. The user c n enter pairs of data points (either from the keyboard, tape or disk), edit the data, obtain a table of s atistics, and plot the original or transformed data in a variety of user-defined configurations. Includes a 40-page users manual. Sugar Software, 2153 Leah Lane, Reynoldsburg, OH 43068, tape $24.95.

Fury, a 32K talking war game. Objective: you are the sole defender of your homeland. As the Communists move in you must shoot down airplanes, helicopters, hot air balloons, parachutes and magnetic air mines while waiting for the CIA backup to arrive. Computer Shack, 1691 Eason, Pontiac, MI 48054, disk $29.95.

Colorcopy, a 32K menu-driven copy utility that copies data files or programs: disk to tape, tape to disk or disk to disk. It also kills files or programs. CoCoPro, P.O. Box 37022, St. Louis, MO 63141, tape $15.

Payout, a 16K/32K accounts payable system written for the Radio Shack Color Computer and the Line Printer VII. Sector Software, 17 Waynewood Drive, Dartmouth, Nova Scotia, Canada B2W 1G1, tape $29.95.

Payup, a 16K/32K accounts receivable system written for the Radio Shack Color Computer and the Line Printer VII. Sector Software, 17 Waynewood Drive, Dartmouth, Nova Scotia, Canada B2W 1G1, tape $29.95.

Database Mailer/Letter Writer, two programs — a database mailer and a two-part letter writing program, both designed to create, maintain and utilize stored information for 16K and 32K tape or disk systems. The first part of **Letter Writer** will help you create and edit a letter file and the second part will allow you to print using your letter file and to merge with **Database Mailer** progr m. Includes a 36-page user's manual. EVS System Engineering Company, 9528 Miramar Road, Suite 35, San Diego, CA 92126, tape or disk $49.95.

The *Seal of Certification* program is open to all manufacturers of products for the TRS-80 Color Computer, the TDP-100, or the Dragon-32, regardless of whether they advertise in *the Rainbow*. By awarding a *Seal,* the magazine certifies the program *does exist,* but this *does not* constitute any guarantee of satisfaction. As soon as possible, these hardware or software items will be forwarded to *the Rainbow's* reviewers for evaluation.

—Jutta Kapfhammer

MEASURE FOR MEASURE

By Jeff Pyne

Here's a conversion program that will bridge the measurement gap and tell you how wide it is in both feet and meters.

Did you ever want to run in a 16 kilometer race, but decided not to because you weren't sure if you could run that far? Or perhaps you have needed to figure out how many liters of gas are required to fill a 10 gallon tank? Well, here is a program written by Jeff Pyne that will help you solve your problems. *Convert* is a useful program that makes it easy to convert inches to millimeters, inches to centimeters, feet to meters, miles to kilometers, square feet to square meters, U.S. gallons to liters, Imperial (British) gallons to liters, ounces to grams, pounds to kilograms, fahrenheit to celsius and vice versa.

After *CLOADing* and *RUNing*, you will find the menu which lists all the conversions. You will be asked which operation you would like and then asked how many inches, feet, ounces, etc.

This program is a good tool for teaching the young and old alike. So, the next time you are asked to run in a 6K race with some friends, you can tell them, "I can do 3.7284 miles easy!"

```
165.... 037E
290.....0621
425.....0932
END...0BEC
```

The listing:

```
10 ' ******************
15 ' *     metric     *
20 ' *   conversion   *
25 ' *  BY JEFF PYNE  *
30 ' *  PORT MOUTON,  *
35 ' *  N.S. CANADA.  *
40 ' ******************
45 CLS:SOUND 200,1:PRINT@13,"men
u"
50 PRINT"    A) END
55 PRINT"    B) METRIC INPUT"
60 PRINT"    C) INCHES-MILLIMETER
S"
65 PRINT"    D) INCHES-CENTIMETER
S"
70 PRINT"    E) FEET-METERS
75 PRINT"    F) MILES-KILOMETERS
80 PRINT"    G) FEET(2)-METERS(2)
85 PRINT"    H) U.S. GALLONS-LITE
RS"
90 PRINT"    I) IMPERIAL GALLONS-
LITERS"
95 PRINT"    J) OUNCES-GRAMS"
100 PRINT"    K) POUNDS-KILOGRAMS
"
105 PRINT"    L) FAHRENHEIT-CELSI
US"
110 PRINT:PRINTTAB(8)"WHAT OPERA
TION?"
115 AN$=INKEY$:IF AN$="" THEN 11
5
120 AN=ASC(AN$)-64
125 IF AN$<CHR$(65) OR AN$>CHR$(
76) THEN 115
130 ON AN GOTO 135,295,140,155,1
70,185,200,215,230,245,260,275
135 CLS:END
140 CLS:INPUT"HOW MANY INCHES";I
145 PRINT"THAT IS";I*25.4;"MILIM
ETERS"
150 GOSUB 540:GOTO 140
155 CLS:INPUT"HOW MANY INCHES";I
160 PRINT"THAT IS";I*2.54;"CENTI
METERS"
165 GOSUB 540:GOTO 155
170 CLS:INPUT"HOW MANY FEET";F
175 PRINT"THAT IS";F*.3048;"METE
```

```
RS"
180 GOSUB 540:GOTO 170
185 CLS:INPUT"HOW MANY MILES";M
190 PRINT"THAT IS";M*1.6093;"KIL
OMETERS"
195 GOSUB 540:GOTO 185
200 CLS:INPUT"HOW MANY FEET(2)";
FF
205 PRINT"THAT IS";FF*.0929;"MET
ERS(2)"
210 GOSUB 540:GOTO 200
215 CLS:INPUT"HOW MANY U.S. GALL
ONS";G
220 PRINT"THAT IS";G*3.7854;"LIT
ERS"
225 GOSUB 540:GOTO 215
230 CLS:INPUT"HOW MANY IMPERIAL
GALLONS";G
235 PRINT"THAT IS";G*4.546;"LITE
RS"
240 GOSUB 540:GOTO 230
245 CLS:INPUT"HOW MANY OUNCES";O
250 PRINT"THAT IS";O*28.35;"GRAM
S"
255 GOSUB 540:GOTO 245
260 CLS:INPUT"HOW MANY POUNDS";P
265 PRINT"THAT IS";P*.4536;"KILO
GRAMS"
270 GOSUB 540:GOTO 260
275 CLS:PRINT"HOW MANY DEGREES"
280 INPUT"FAHRENHEIT";F
285 PRINT"THAT IS";(F-32)*5/9:PR
INT"DEGREES CELSIUS"
290 GOSUB 540:GOTO 275
295 CLS:SOUND 200,1:PRINT@13,"me
nu"
300 PRINT"    A) END"
305 PRINT"    B) IMPERIAL INPUT"
310 PRINT"    C) MILLIMETERS-INCH
ES"
315 PRINT"    D) CENTIMETERS-INCH
ES"
320 PRINT"    E) METERS-FEET
325 PRINT"    F) KILOMETERS-MILES
"
330 PRINT"    G) METERS(2)-FEET(2
)"
335 PRINT"    H) LITERS-U.S. GALL
ONS"
340 PRINT"    I) LITERS-IMPERIAL
GALLONS"
345 PRINT"    J) GRAMS-OUNCES"
350 PRINT"    K) KILOGRAMS-POUNDS
"
355 PRINT"    L) CELSIUS-FAHRENHE
IT"
360 PRINT:PRINTTAB(8)"WHAT OPERA
TION?"
365 AN$=INKEY$:IF AN$="" THEN 36
5
370 AN=ASC(AN$)-64
375 IF AN$<CHR$(65) OR AN$>CHR$(
76) THEN 365
380 ON AN GOTO 135,45,385,400,41
5,430,445,460,475,490,505,520
385 CLS:INPUT"HOW MANY MILLIMETE
RS";M
390 PRINT"THAT IS";M*.0394;"INCH
ES"
395 GOSUB 540:GOTO 385
400 CLS:INPUT"HOW MANY CENTIMETE
RS";C
405 PRINT"THAT IS";C*.3937;"INCH
ES"
410 GOSUB 540:GOTO 400
415 CLS:INPUT"HOW MANY METERS";M
420 PRINT"THAT IS";M*3.2808;"FEE
T"
425 GOSUB 540:GOTO 415
430 CLS:INPUT"HOW MANY KILOMETER
S";K
435 PRINT"THAT IS";K*.6214;"MILE
S"
440 GOSUB 540:GOTO 430
445 CLS:INPUT"HOW MANY METERS(2)
";MM
450 PRINT"THAT IS";MM*10.764;"FE
ET(2)"
455 GOSUB 540:GOTO 445
460 CLS:INPUT"HOW MANY LITERS";L
465 PRINT"THAT IS";L*.2642;"U.S.
 GALLONS"
470 GOSUB 540:GOTO 460
475 CLS:INPUT"HOW MANY LITERS";L
480 PRINT"THAT IS";L*.22;"IMPERI
AL GALLONS"
485 GOSUB 540:GOTO 475
490 CLS:INPUT"HOW MANY GRAMS";G
495 PRINT"THAT IS";G*.0353;"OUNC
ES"
500 GOSUB 540:GOTO 490
505 CLS:INPUT"HOW MANY KILOGRAMS
";K
510 PRINT"THAT IS";K*2.2046;"POU
NDS"
515 GOSUB 540:GOTO 505
520 CLS:PRINT"HOW MANY DEGREES"
525 INPUT"CELSIUS";C
530 PRINT"THAT IS";(C*9/5)+32:PR
INT"DEGREES FAHRENHEIT"
535 GOSUB 540:GOTO 520
540 PRINT:PRINT"ANOTHER (Y OR N)
?"
545 RP$=INKEY$:IF RP$="" THEN 54
5
550 IF RP$<CHR$(65) THEN 545
555 IF RP$="Y" THEN RETURN ELSE
45
```

Grade Averaging Could Be Handy Teacher's Aid

By Paul N. Luetke

Grade Averaging is an attempt at what I consider useful programming. It isn't that game programs are useless, but I feel that CoCo should also be used to save time so that there's some left in which to play all those great games available for the powerful Color Computer. I began writing this program in November of 1981, about one month after purchasing my Color Computer. I must give due credit to Al Pomerantz, a good friend of mine, who not only convinced me of CoCo's power, but also gave me valuable help with this program.

In order to use this program as is, you must either use my system of grades/record keeping or adapt yours to suit. I keep cumulative records throughout the year using numbers which are later converted to letter grades. The following example should help to illustrate.

A+=0, A=1, A-=2, B+=3, B=4, B-=5, C+=6, C=7, C-=8, D+=9, D=10, D-=11, F=12, and incomplete work counts as a ZERO, or a 40 in this scale.

NAME	Mon.	Tues.	Wed.	Thurs.	Fri.	TOTAL
Frank	1	8	10	16	19	19
grades—	A	C	A-	C+	B+	Bavg.
	1 (1+7)	8 (8+2)	10 (10+6)	16 (16+3)	19	

The first few lines of the program use a neat little routine I learned in order to clear the graphics pages without doing it directly from the keyboard. The program is initializing the numerous strings while the credit page is on screen.

The first menu to appear on screen asks the user to identify himself. The program is presently set up for four teachers and thirty students, but could easily be adapted for more by adding the needed names in line 280 and in the data statements beginning at line 1150. When entering this data be sure that the first name is that of the teacher and that at least 30 bits of data are recorded including empty quotes or you will get an OD error. The dimension statement in line 1090 must also be changed appropriately. CS is for teachers' and students' names. A is the variable for

(Paul Luetke, who is working toward a master's degree in computer assisted education, teaches on the junior high school level at Grace Lutheran School (WELS) in Durand, Michigan.)

daily grades and B is for test grades. PA refers to percentile average, GC refers to grade code, and FA is for the final average. The *FOR/NEXT* loops in lines 1100 and 1110 must also be changed if more teachers or students are to be used. Since I only have data for one teacher in the program, the X loop is 1 TO 1.

The second menu allows you to choose the subject to average. To change these subjects, edit lines 320 and 330 and 2320—2410. Note that the variable GC indicates the number of periods per week that the class is in session. An example of this would be language which is taught four days per week. This item is used in determining the student's overall average.

After entering the subject to average, the program will ask for the number of daily grades that you are averaging for that subject. Next, it asks for the number of test grades to be averaged.

Lastly, the program will ask for the grade weighting desired for each. The total of the two numbers entered here should be equal to 10. I use different weightings for each subject, but if daily grades count for 60 percent of the final grade and test grades count 40 percent, then you would enter a 6 and a 4 respectively. If no tests were given, then enter a 10 for the weighting of the daily grades and a 0 for the test grades.

The actual grade calculations take place in lines 770—790. The printing routine which prints both a class report and individual reports begins in line 1420. The program is written for the Line Printer VII.

Enter *PMODE 0* before loading the program.

I hope that this program will be of use to many of you, especially those of you who are teachers. I have been printing weekly report cards for all of my students for the past two years. It takes me only about 20 minutes to enter all of the grades for all 10 subject fields. I have found that both students and parents appreciate it very much. It helps me also, because there are no surprises at quarterly marking periods anymore.

I realize that the program is quite long, so if you don't feel like typing it in, send me $5 to cover shipping and handling and I will send you a copy. Write to: Paul N. Luetke, 8499 E. M-71 Lot 143, Durand, Mich., 48429.

The listing:

300.... 01AE	1460... 0FED
420.... 049F	1670... 128B
620.... 071F	1830....1472
840.... 097F	2010... 177A
1110... 0B9A	2180... 19AA
1320... 0E19	END... 1DE0

```
100 PMODE0
110 GOTO130
120 GOTO140
130 GOTO2300
140 REM
150 PCLEAR1
160 CLS
170 PRINT@135,"GRADE AVERAGING"
180 PRINT@206,"BY":PRINT@232,"PA
UL N. LUETKE"
190 PRINT@329,"---- --- ----"
200 GOTO220
210 Z$=INKEY$:IFZ$<>""THENRETURN
ELSE210
220 CLEAR500
230 GOSUB1090
240 FORX=1TO30:FORSC=1TO10:FA(SC
,X)=100:NEXTSC,X
250 CLS
260 PRINT:PRINT" NAME          COD
E"
270 PRINTSTRING$(32,"=");:PRINT
280 PRINT"LUETKE          1":PRINT
"HATZUNG        2":PRINT"MUNDT
      3":PRINT"LUEDTKE        4":
PRINT:INPUT"teacher code";TC
290 CLS
300 PRINT"SUBJECT       CODE"
310 PRINTSTRING$(32,"=")
320 PRINT"LANGUAGE        1":GC(1)
=4:PRINT"SOCIAL STUDY  2":GC(2)=
5:PRINT"LITERATURE     3":GC(3)=5
:PRINT"MATH              4":GC(4)=5:
PRINT"SCIENCE          5":GC(5)=5:P
RINT"SPELLING        6":GC(6)=4:PR
INT"PENMANSHIP     7":GC(7)=1:PRI
NT"ART               8":GC(8)=1
330 PRINT"HYMNOLOGY       9":GC(9)
=1:PRINT"WORD OF GOD  10":GC(10)
=4:PRINT:INPUT"subject code";SC
340 ONSC GOSUB2320,2330,2340,235
0,2360,2370,2380,2390,2400,2410
350 CLS:PRINTSTRING$(32,"?");:PR
INT"??????????? HOW MANY ???????
????";:PRINTSTRING$(32,"?"):PRIN
T"DAILY GRADES FOR:";N$;:INPUTN
360 PRINT@192,"TEST  GRADES FOR:
";N$;:INPUTM
370 PRINT@256,STRING$(32,"?");:P
RINT"??????? GRADE WEIGHTING ???
?????";:PRINTSTRING$(32,"?")
380 PRINT@384,"DAILY GRADES FOR:
";N$;:INPUTE
390 PRINT@448,"TEST  BRADES FOR:
";N$;:INPUTR
400 IFE+R<>10THENCLS:PRINT@128,"
THE TOTAL WEIGHT MUST EQUAL 10":
FORX=1TO1000:NEXTX:GOTO370
410 GV=GV+GC(SC)
420 ST=0
430 IFC$(TC,ST+1)=""THEN460
440 ST=ST+1:GOTO430
450 PRINTSTRING$(32,"=");:PRINT
460 CLS:PRINT:PRINT" ENTER CUMUL
ATIVE DAILY GRADES":PRINT:SP$="
      ":QP$="      "
470 PRINTSTRING$(32,"=")
480 FORX=1TOST:W$=C$(TC,X):W$=W$
+SP$:W$=LEFT$(W$,10):IFX<10THENP
RINT" ";X;W$+"  ";:INPUTA(SC,X):
NEXTX
490 PRINTX;W$+"  ";:INPUTA(SC,X)
```

```
:NEXTX
500 CLS:FORX=1TOST:W$=C$(TC,X):W
$=W$+QP$:W$=LEFT$(W$,7):IFX<10TH
ENPRINT" ";X;W$;A(SC,X),:NEXTX
510 PRINTX;W$;A(SC,X),:NEXTX
520 PRINT@418,"DO YOU WISH TO CH
ANGE ANY?":GOSUB210
530 IFZ$="Y"THEN560
540 IFZ$="N"THEN600
550 IFZ$<>"Y"ORZ$<>"N"THEN520
560 INPUT" ENTER THE STUDENT'S N
UMBER";X
570 IFX<1ORX>ST THEN560
580 INPUT"ENTER CORRECTED GRADE
VALUE";A(SC,X)
590 GOTO500
600 IFM=0THEN1390
610 CLS:PRINT:PRINT" ENTER CUMUL
ATIVE TEST GRADES":PRINT
620 PRINTSTRING$(32,"=")
630 FORX=1TOST:W$=C$(TC,X):W$=W$
+SP$:W$=LEFT$(W$,10):IFX<10THENP
RINT" ";X;W$+"  ";:INPUTB(SC,X):
NEXTX
640 PRINTX;W$+"  ";:INPUTB(SC,X)
:NEXTX
650 CLS:FORX=1TOST:W$=C$(TC,X):W
$=W$+QP$:W$=LEFT$(W$,7):IFX<10TH
ENPRINT" "X;W$;B(SC,X),:NEXTX
660 PRINTX;W$;B(SC,X),:NEXTX
```



```
670 PRINT@418,"DO YOU WISH TO CH
ANGE ANY?":GOSUB210
680 IFZ$="Y"THEN710
690 IFZ$="N"THEN750
700 IFZ$<>"Y"ORZ$<>"N"THEN670
710 INPUT" ENTER THE STUDENT'S N
UMBER";X
720 IFX<0ORX>ST THEN710
730 INPUT"ENTER CORRECTED GRADE
VALUE";B(SC,X)
740 GOTO650
750 CLS
760 FORX=1TOST
770 F=(((A(SC,X)/N)*E)+((B(SC,X)
/M)*R))/10
780 PA(X)=PA(X)+F*GC(SC)
790 F=INT(F+.5)
800 IFXY=0THEN1340
810 FA(SC,X)=F
820 IFF=100THENGOSUB1080:GOTO850
830 IFF>12THENF$="F ":GOTO850
840 GOSUB2420
850 W$=C$(TC,X):W$=W$+SP$:W$=LEF
T$(W$,10)
860 PRINT"  "+W$;F$,
870 NEXTX
880 IFQW=0THEN1260
890 PRINT@420,"DO YOU WANT TO AV
ERAGE":PRINT@458,"MORE GRADES"
900 GOSUB210
910 IFZ$="Y"THEN290
920 IFZ$="N"THENPRINT"          are
 you sure??";:INPUTAS$
930 AS$=LEFT$(AS$,1):IFAS$="Y"TH
EN1200ELSE290
940 IFZ$<>"Y"ORZ$<>"N"THEN890
950 F$="A+":RETURN
960 F$="A ":RETURN
970 F$="A-":RETURN
980 F$="B+":RETURN
990 F$="B ":RETURN
1000 F$="B-":RETURN
1010 F$="C+":RETURN
1020 F$="C ":RETURN
1030 F$="C-":RETURN
1040 F$="D+":RETURN
1050 F$="D ":RETURN
1060 F$="D-":RETURN
1070 F$="F ":RETURN
1080 F$="  ":RETURN
1090 DIMC$(4,30),A(10,30),B(10,3
0),PA(30),GC(10),FA(11,30)
1100 FORX=1TO1
1110 FORY=0TO30
1120 READC$(X,Y)
1130 NEXTY,X
1140 RETURN
1150 DATA "LUETKE","MICHAEL","JI
M","SUSAN","CHRIS","DAVID","BECK
Y","SHANNON","BETTY","TINA","WES
```

```
","KATHY","HARMON","MELYNDA","JE
RRY","REBECCA","ERICA","BEN","ME
LANIE","MATTHEW","MARTHA","DEBOR
AH","KAREN","BURT","","","","","
"
1160 DATA
1170 DATA
1180 DATA
1190 DATA
1200 CLS
1210 PRINT"    DO YOU WANT TO SE
E THE    "
1220 PRINT"        OVERALL AVERAG
ES       ":GOSUB210
1230 IFZ$="Y"THEN1300
1240 IFZ$="N"THEN1260
1250 IFZ$<>"Y"ORZ$<>"N"THEN1210
1260 PRINT:PRINT"    DO YOU WANT
 TO SEE THE":PRINT"        DECIMA
L AVERAGES":GOSUB210
1270 IFZ$="Y"THEN1330
1280 IFZ$="N"THEN1420
1290 IFZ$<>"Y"ORZ$<>"N"THEN1260
1300 QW=8:CLS:PRINT"          OVER
ALL AVERAGES":FORX=1TOST
1310 F=PA(X)/GV:FA(11,X)=F:SC=0
1320 GOTO790
1330 CLS:PRINT"          DECIMAL A
VERAGES":FORX=1TOST
1340 W$=C$(TC,X):W$=W$+QP$:W$=LE
FT$(W$,9)
1350 PRINT" "+W$;:PRINTUSING"##.
##";PA(X)/GV;:PRINT"",
1360 NEXTX
1370 PRINT:PRINT:PRINT"PRESS ANY
 KEY TO CONTINUE"
1380 A$=INKEY$:IFA$<>""THEN1420E
LSE1380
1390 FORX=1TOST:B(SC,X)=1:NEXTX
1400 M=1000000000
1410 GOTO750
1420 CLS:PRINT"ENTER ENDING DATE
 OF THE PRESENTGRADING PERIOD
[MON-DAY-YEAR]
 ";:INPUTVB$
1430 CLS:PRINT"IS THE PRINTER RE
ADY?":GOSUB210
1440 IFZ$<>"Y"THEN1430
1450 CLS:PRINT"          WOULD YOU
 LIKE A":PRINT"        CLASS REPO
RT CARD"
1460 GOSUB210
1470 IFZ$="Y"THEN1480ELSE1670
1480 PRINT#-2,CHR$(31);TAB(5)"CL
ASS REPORT ENDING   [";VB$;"]":P
RINT#-2,CHR$(30)
1490 PRINT#-2,""
1500 GOSUB2150:FORX=1TOST
```

```
1510 W$=C$(TC,X):W$=W$+SP$:W$=LE
FT$(W$,10)
1520 PRINT#-2,W$;"    ";
1530 FORSC=1TO10
1540 IFFA(SC,X)=100THENGOSUB1080
:GOTO1560:ELSEIFFA(SC,X)+1>12THE
NF$="F ":GOTO1560
1550 ONFA(SC,X)+1GOSUB950,960,97
0,980,990,1000,1010,1020,1030,10
40,1050,1060,1070
1560 PRINT#-2,F$+"    ";
1570 NEXTSC
1580 F=PA(X)/GV:F=INT(F+.5)
1590 IFF=100THENGOSUB1080:GOTO16
10:ELSEIFF>12THENF$="F ":GOTO161
0
1600 GOSUB2420
1610 PRINT#-2,"    "+F$+"    ";
1620 PRINT#-2,USING"##.###";FA(1
1,X)
1630 NEXTX
1640 PRINT#-2,"":PRINT#-2,"":PRI
NT#-2,""
1650 CLS:PRINT"WOULD YOU LIKE AN
OTHER?":GOSUB210
1660 IFZ$="Y"THEN1480ELSE1670
1670 CLS:PRINT"WOULD YOU LIKE IN
DIVIDUAL REPORT  CARDS FOR ALL Y
OUR STUDENTS":GOSUB210
1680 IFZ$="Y"THEN1700
```



```
1700 FORX=1TOST:W$=C$(TC,X):W$=W
$+SP$:W$=LEFT$(W$,10)
1710 PRINT#-2,CHR$(30);TAB(15)"
    REPORT FOR    ";CHR$(31);W$;C
HR$(30);"ENDING    [";VB$;"]":PRI
NT#-2,""
1720 GOSUB2150
1730 W$=C$(TC,X):W$=W$+SP$:W$=LE
FT$(W$,10)
1740 PRINT#-2,W$;"    ";
1750 FORSC=1TO10
1760 IFFA(SC,X)=100THENGOSUB1080
:GOTO1790
1770 IFFA(SC,X)+1>12THENF$="F ":
GOTO1790
1780 ONFA(SC,X)+1GOSUB950,960,97
0,980,990,1000,1010,1020,1030,10
40,1050,1060,1070,1080
1790 PRINT#-2,F$+"    ";
1800 NEXTSC
1810 F=PA(X)/GV:F=INT(F+.5)
1820 IFF>12THENF$="F ":GOTO1840
1830 GOSUB2420
1840 PRINT#-2,"    "+F$+"    ";
1850 PRINT#-2,USING"##.###";FA(1
1,X)
1860 PRINT#-2,"":PRINT#-2,"    A+
=0  A=1  A-=2  B+=3  B=4  B-=5
C+=6  C=7  C-=8  D+=9  D=10  D-=
11  F=12":PRINT#-2,"":PRINT#-2,"
":PRINT#-2,""
1870 NEXTX
1880 CLS:PRINT"  WOULD YOU LIKE
A REPORT CARD       FOR A SPECIF
IC STUDENT":GOSUB210
1890 IFZ$="Y"THEN1910
1900 IFZ$="N"THEN2140
1910 CLS:FORX=1TOST:W$=C$(TC,X):
W$=W$+SP$:W$=LEFT$(W$,10):IFX<ST
 THENPRINT" ";X;W$,:NEXTX
1920 PRINT:INPUT"PLEASE ENTER TH
E STUDENT NUMBER";X
1930 INPUT"PLEASE ENTER THE STUD
ENT'S NAME";XA$
1940 PRINT#-2,CHR$(30);TAB(15)"
    REPORT FOR ";CHR$(31);XA$;"
";CHR$(30);"ENDING    [";VB$;"]":
PRINT#-2,""
1950 PRINT#-2,""
1960 GOSUB2150
1970 W$=C$(TC,X):W$=W$+SP$:W$=LE
FT$(W$,10)
1980 PRINT#-2,W$;"    ";
1990 FORSC=1TO10
2000 IFFA(SC,X)=100THENGOSUB1080
:GOTO2020:ELSEIFFA(SC,X)+1>12THE
NF$="F ":GOTO2020
2010 ONFA(SC,X)+1GOSUB950,960,97
0,980,990,1000,1010,1020,1030,10
40,1050,1060,1070,1080
```

```
2020 PRINT#-2,F$+"    "1
2030 NEXTSC
2040 F=PA(X)/GV:F=INT(F+.5)
2050 IFF>12THENF$="F ":GOTO2070
2060 GOSUB2420
2070 PRINT#-2,"    "+F$+"    ";
2080 PRINT#-2,USING"##.###";FA(1
1,X)
2090 PRINT#-2,"":PRINT#-2,"":PRI
NT#-2,""
2100 CLS:PRINT"WOULD YOU LIKE AN
OTHER?":GOSUB210
2110 IFZ$="Y"THEN1910
2120 IFZ$="N"THEN2140
2130 IFZ$<>"Y"ORZ$<>"N"THEN2110
2140 CLS:PRINT"THANK YOU":END
2150 PRINT#-2,""
2160 PRINT#-2,"   N          L
 S    L    M    S    S    P    A
    H    W         G        D"
2170 PRINT#-2,"   A          A
 O    I    A    C    P    E    R
    Y    O         R        E"
2180 PRINT#-2,"   M          N
 C    T    T    I    E    N    T
    M    R         A        C"
2190 PRINT#-2,"   E          G
 I    E    H    E    L    M    .
    N    D         D        I"
```



```
2200 PRINT#-2,"   .          U
 A    R    .    N    L    A    .
    O         E        M"
2210 PRINT#-2,"   .          A
 L    A    .    C    I    N    .
    L    O              A"
2220 PRINT#-2,"   .          G
    T    .    E    N    S    .
    O    F    A        L"
2230 PRINT#-2,"   .          E
 S    U    .    .    G    H    .
    G         V"
2240 PRINT#-2,"   .          .
 T    R    .    .    .    I    .
    Y    G         G        A"
2250 PRINT#-2,"   .          .
 .    E    .    .    .    P    .
    .    O         .        V"
2260 PRINT#-2,"   .          .
 .    .    .    .    .    .    .
    .    D         .        G"
2270 PRINT#-2,"   .          .
 .    .    .    .    .    .    .
    .    .         .        ."
2280 PRINT#-2,""
2290 RETURN
2300 PCLEAR1
2310 GOTO120
2320 N$="language":RETURN
2330 N$="social st.":RETURN
2340 N$="literature":RETURN
2350 N$="math":RETURN
2360 N$="science":RETURN
2370 N$="spelling":RETURN
2380 N$="penmanship":RETURN
2390 N$="art":RETURN
2400 N$="hymnology":RETURN
2410 N$="word of god":RETURN
2420 ONF+1GOSUB950,960,970,980,9
90,1000,1010,1020,1030,1040,1050
,1060,1070
2430 RETURN
```

Hint . . .

Speed Up Your CoCo

You can double the speed at which CoCo operates with a simple POKE statement, entered either directly from the keyboard or within a program. The statement is POKE 65495,0. This will speed up your CPU. You can return the computer to normal speed again by POKE 65494,0.

Note that you cannot perform any Input/Output operations, such as saving the program to cassette, when the speedup is in effect. You may also lose temporary keyboard control. If this happens, a simple press of the RESET button will bring things back to normal, too.

DYNAMIC UNO
The New 64K CoCo

By Dan Downard
***Rainbow* Technical Editor**

What do a kernel, tree, pipeline, path, shell and gobbler have in common?

Nothing but *OS-9*, the new Radio Shack operating system for the new 64K CoCo. Sounds like a verse of Old MacDonald, doesn't it? Actually the new CoCo and the operating system are to be sold in separate packages, but they go together to form one of the most dynamic innovations in the personal computer field to date.

Imagine yourself sitting in front of your CoCo running a program, or maybe playing a game, while a friend is accessing files to his program through your RS-232 port. How about running Pascal, C and several other high level languages? All of this on a $199 computer? Well, to be realistic, we are talking in the neighborhood of $800, when you include the disk drive and all. Yes, you can have *the* most sophisticated microcomputer *system* available today for less than half the cost of a bare-bones Apple II.

OS-9 is a multi-user, multi-tasking, operating system developed for Tandy by Motorola and Microware for the Color Computer. It is being marketed simultaneously with a new 64K Color Computer with a "real" keyboard and a few other minor changes. To sum the combination in a short phrase, I would say that Tandy is "far ahead of any competitor." No longer is the CoCo just for games and "a toy" (not that it *ever* was!)

64K CoCo—A New keyboard

By the time you are reading this review the new 64K Color Computers should be on the dealers' shelves. We have been hearing rumors of new CoCo for some time now and finally one has appeared. How much is new? Well, the keyboard for a start. As you can see the 64K CoCo and CoCo 2 have a typewriter-like keyboard with keys—that "travel." There is more to the keyboard than meets the eye. A full RF shield covers the bottom plane of the circuit board supporting the keys. It is grounded by means of a metal clip to the computer board. All keys are spill resistant and have a good "touch." I welcome this addition to the family. We understand that it will be available for $39.95 (plus installation) for upgrading existing CoCos.

Other Changes

The new computer is housed in a fawn-gray enclosure instead of battleship gray, but very similar to the existing CoCo. Obviously a 64K CoCo has to have 64K chips. I wonder how many of you realize that if you have a "285" board with 32K chips you already have 64K. More than that, if you have the 1.2 BASIC and the 1.1 Extended BASIC ROM you have the same thing inside your computer as the 64K CoCo. I guess you could say it was a surprise when we discovered that there had been no changes to the boards for this revision. On the other hand, there have been a few minor

changes to the ROMs. Following is a brief description of the changes.

Addresses Change

	Extended BASIC 1.1
80D0-80DD	Corrects bug with parse counter
80FF	Changes "1.0" to "1.1"
8112	Changes "1980" to "1982"
8C1B-8C51	Corrects bug in DLOAD
9179-917D	Changes to RS-232 out
962C-962D	Changes to PMODE
96A3-96B3	Corrects PCLEAR error
	BASIC 1.2
A[illegible]01	Changes entry address of POLCAT
A114	New Baud constant
A155	Changes "1.1" to "1.2"
A15E	Changes "1980" to "1982"
A1B6	Changes branch to POLCAT
A1C1-A26D	Revised POLCAT routine
A2C3-A2D5	Poll Pin 2 before printing
B3F3-B426	Changes to INTCNV
B[illegible]D6	Correct bug in subtract routine

The only change that should affect anyone is the different address of POLCAT, or the keyboard scan routine. The new entry point is $A1CB instead of $A1C1. I would not advise you to rush out and buy new ROMs as the present versions are compatible. There are changes to the DISK ROM in the works too. Since we did not have the new controller the only thing we know for sure is that the new DISK ROM will have a *DOS* command for booting *OS-9*.

OS-9

The real strength of the new package lies in the operating system. Tandy, along with Motorola and Microware, have adapted their *UNIX* based *OS-9* system to the Color Computer. In today's vernacular I would term it "awesome." *OS-9* is a multi-user, multi-tasking, interrupt driven DOS with multi-level file structures.

Boy, that's a mouthful. By multi-user, we mean literally that one person can operate CoCo from the keyboard while another can access the computer using the RS-232 port via a remote terminal (which can be another CoCo).

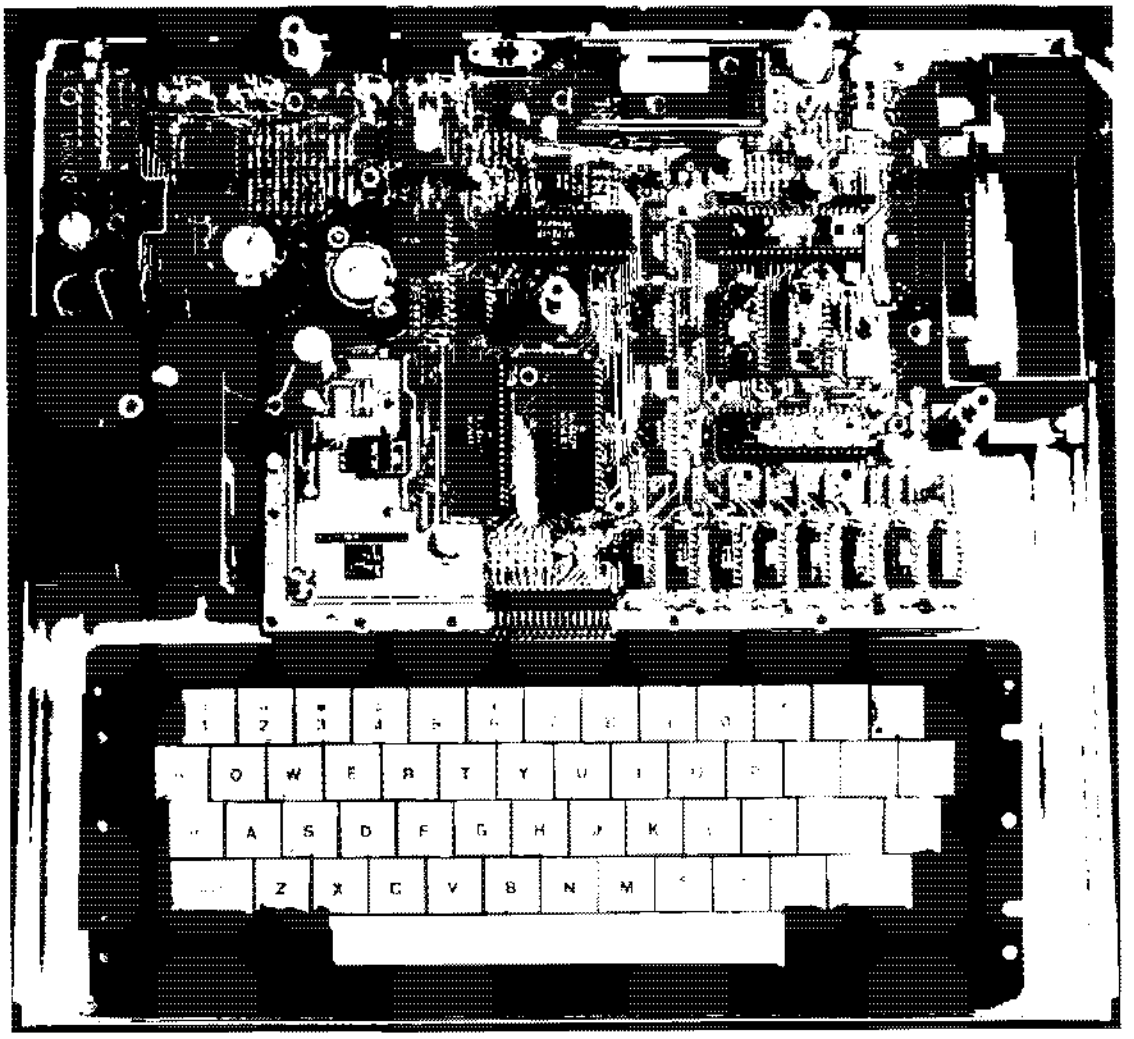
Using multi-tasking involves the processor performing two or more tasks simultaneously such as inputting to one file while outputting another to the printer. Interrupt driven implies a real-time processing environment that is user friendly. In fact, *OS-9* has a real-time clock running continuously that will give you time with a keystroke.

We will discuss multi-level files, but in essence the easiest way to understand it is to visualize a disk with more than one directory. With *OS-9* you can have as many directories as you want and any directory can contain another subdirectory, or files. Tracing this hierarchial structure is called a "path."

As you can see, this system is a little more complex that Disk BASIC, but don't let that scare you. As was mentioned earlier, this DOS opens a wealth of business, scientific and engineering software to the CoCo along with new high-level languages such as Cobol, Pascal, etc.

Peek Inside The New 64K CoCo.

"I wonder how many of you realize that if you have a '285' board with 32K chips you already have 64K. More than that, if you have the 1.2 BASIC and the 1.1 Extended BASIC ROM you have the same thing inside your computer as the 64K CoCo. I guess you could say it was a surprise when we discovered that there had been no changes to the boards for this revision."



Some of the more interesting features of *OS-9* are the utilities provided with the package. Not only do you get the operating system, but excellent text editor, assembler and debug programs. No high resolution screen is provided but upper-lowercase is fully functional for future upgrades. The printer driver supports varied formats and the entire system can be tailored for your needs.

On the other hand BASIC is not supplied with the operating system and will be available as a separate package as will other higher level programming languages. You can still use Disk BASIC on the same computer, but not with *OS-9*.

The documentation is extensive. Four different manuals are provided. True to the CoCo tradition the first manual is named *"Getting Started With OS-9."* An overview of system operation and startup procedures is given in a clear, concise format with several examples. Other documentation includes a User's Manual and a Technical Manual along with a manual for the Editor, Assembler and Debugger. The latter manuals are similar to the Model I and III formats with considerably greater detail with regard to the operating system. I was amazed at the volume of documentation. Tandy probably has to cut down a whole tree for each set.

Booting *OS-9*

To run *OS-9* you need a 64K CoCo with one disk drive. No new hardware is required for the new operating system. The system is supplied on two disks. The first disk is formatted in Disk BASIC and contains two programs, a drive speed check (a real bonus!) and the *OS-9* boot program.

The speed check is very user friendly and recommended before attempting to boot *OS-9*. An error message appears if your drives are not in tolerance, that is, between 298.0 and 303.5 RPM. You are requested to return the drive(s) to a service center for adjustment if they fail the test. You *can* do this yourself. We'll have details next month.

Booting the new system isn't very complex. Two different methods are described, depending on the version of your DISK ROM. With Version 1.0, the boot disk is inserted and you type *RUN"*". You are prompted to insert the system master disk and that's it. With Versions 1.1 and later you just type *DOS*. You are then greeted with the startup message:

OS-9 LEVEL ONE
RS VERSION 01.00.00
COPYRIGHT 1980 BY MOTOROLA, INC.
AND MICROWARE SYSTEMS CORP.
REPRODUCED UNDER LICENSE
TO TANDY CORP.
ALL RIGHTS RESERVED
YY/MM/DD HH:MM:SS

You answer the prompts with the date and time and you're in business. The disks are not copy protected as you will have to make several backups of the system disk if you are using a single drive for your various files and directories. Step-by-step instructions are given on making a backup and are easy to follow. Disks are in a different format than Disk BASIC and have to be formatted with *OS-9* before the backup is made.

The "Shell"

The shell, or command processor, is the heart of *OS-9* that interfaces the user to the OS. Commands may be entered in upper or lower case but must be entered in a structured format. Commands may be the name of a machine language program on disk or in memory, a program or a procedure

file. A typical command would appear as:

dir /D0/BUSINESS

This particular command would list the contents of directory BUSINESS in Drive 0. Commands are separated by spaces. Normally directories are uppercase and files lowercase by convention. Modifiers for I/O are flagged by a /.

Five different I/O devices are contained in the System Service Directory called by the SHELL. P is the printer, D0 is drive 0, TERM is the keyboard/display, D1 is drive 1 and T1 is a terminal connected to the RS-232 port. All input and output can be directed through any of these devices.

To give you an example of the different levels of directories, D0, or drive 0, actually contains another directory called the Root Directory. This directory contains all of the commands in another directory called CMDS plus a file called startup. Startup is called each time the system is booted and normally contains user initialization for such things as printer baud rate, terminal parameters, etc.

One of the disappointments was the lack of a high resolution screen. I don't think it will be long before one appears on the market, but Tandy already has them with other software and I can't understand why it wasn't implemented. At the same time, the screen is scroll protected every 16 lines. No more holding the shift-@ key to see what you want before it disappears. Several keys have been redefined and all control characters are available. The clear key is used as a "control" key for most operations.

Other Features

Sequential and random access files are supported by *OS-9*. Every file has attributes such as read, write and execution by the owner, or the public. Disk format, although different from Disk BASIC, maintains the same number of tracks and sectors on the disk with one important exception. *OS-9* allocates disk space by "clusters." A cluster can be from one to any integral power of two (2, 4, 8, etc.). One sector of each disk is used as a bitmap indicating which clusters are in use.

A technique called "timeslicing" allows all active processes to share CPU time. A real-time clock interrupts the processor 60 times a second. The interruption points are called "ticks." At any tick, the DOS can suspend execution of one program and begin another. Each process is assigned a priority from 0 to 255 with 0 being the highest. The shell has the highest priority due to I/O, etc. By this method, several programs can be executed at the same time.

What this means is that your CoCo now has a *true* time-sharing system—just like a mainframe! No kidding. This truly opens up a whole vista of possibilities in the microcomputer market. It constitutes a *major* innovation by Radio Shack.

All programs for *OS-9* are written in position independent code, allowing advanced memory management. Programs are loaded into memory only when needed but the user maintains the option of making them memory resident. Approximately 48K of user memory remains after the shell, drivers and screen memory. Programs are loaded into the top part of memory and data occupies the bottom part. Both are dynamically allocated and expand toward the center as required. Several programs can be in memory at once and even executed at once. Due to a combination of memory management and multi-tasking, multi-user programs are a reality.

Commands

A list of commands is given in Figure 1 with a short description of each. As you can see, most are standard disk commands you would find in other DOS's. Excellent documentation provides information on the use of each command with examples. I'm sure these commands will be explored in detail in future *Rainbow* articles.

Editor, Assembler and Debug

As *OS-9* is machine language oriented, a cry of joy was detected at the sight of a standard disk-based Editor-Assembler for the Color Computer. Both have full MACRO capability, are user friendly and have excellent documentation. Calls to the *OS-9* system are made by the use of interrupts (SWI2) followed by a descriptor. All of the system calls are contained in a MACRO directory named DEFS. For this reason, an assembler is a necessity. The assembler accepts all of Motorola's directives and pseudo-operations and instructions are given in writing assembly language programs in an *OS-9* format.

The debug module is excellent. In addition to an expression calculator, the program offers breakpoints, register examine and find commands among others.

Summary

As you can tell I am enthusiastic about the possibilities of *OS-9*. Excellent software is already available. Look in the ads this month. I'm sure Radio Shack will support this system especially in the areas of small business and engineering, areas which up until now have been avoided. I like the idea of the DOS being available to everyone along with the keyboard upgrade kit. Awesome!

Command Summary

Attr	Change file attributes
Backup	Make disk backup
Binex	Convert binary to s-record
Build	Build text file
Chd	Change working data directory
Chx	Change working execution directory
Cmp	File comparison utility
Cobbler	Make bootstrap file
Copy	Copy data
Date	Display system date and time
Dcheck	Check disk file structure
Del	Delete a file
Deldir	Delete all files in a directory system
Dir	Display filenames in a directory
Display	Display converted characters
Dsave	Generate procedure file to copy files
Dump	Formatted file dump
Echo	Echo text to output path
Exbin	Convert s-record to binary
Format	Initialize disk media
Free	Display free space on device
Ident	Print OS-9 module identification
Kill	Abort a process
Link	Link module into memory
List	List contents of disk file
Load	Load module(s) into memory
Login	Timesharing system log-in
Makdir	Create directory file
Mdir	Display module directory
Merge	Copy and combine files
Mfree	Display free system RAM memory
OS9Gen	Build and link a bootstrap file
Printerr	Print full-text error messages
Procs	Display processes
Pwd	Print working directory
Pxd	Print execution directory
Rename	Change filename
Save	Save memory module(s) on a file
Setime	Activate and set system clock
Setpr	Set process priority
Sleep	Suspend process for period of time
Shell	OS-9 command interpreter
Tee	Copy standard input to multiple output paths
Tmode	Change terminal operating mode
Tsmon	Timesharing monitor
Unlink	Unlink memory module
Verify	Verify or update module header and CRC
Xmode	Examine or change device initialization mode

Hint . . .

Automatic Lower Case

To move automatically into lower case, type *POKE*282,0. To return to upper case, use *POKE* 282,1.

—Ken Ostrer

Bibbits and Gribbits in *Moptown Hotel*

Ever since the *Moptown Hotel* arrived for review, our family has "adopted" a number of strange but wonderful critters known as the moppets which inhabit Moptown. Moppets, which come in two kinds—Bibbits and Gribbits are a real way of teaching children (ages six-13) the concepts of similarities and differences. *Moptown Hotel* is part of a series of educational programs from the Follet Library Book Co. which includes *Moptown Twin* and *Moptown Parade*—all three packages are available on cassette or disk and require a 16K Extended BASIC Color Computer. The disk version of *Moptown Hotel* is the subject of this review, which we decided to make a family review. Being an educational package, what better way to review the program than to have our 11-year-old program tester (Chris) put the package through its paces.

The *Moptown* series starts with games involving simple comparisons including the recognition of common similarities or differences between moppets and progresses through various games leading to the *Moptown Hotel* which develops higher level problem solving concepts. Thus, *Moptown Hotel* is designed for the latter half of this age group. Those with children six to eight would do best with the first programs mentioned earlier.

Moptown Hotel consists of five separate activities. With the simplier activities, the child is presented with the task of analyzing differences and similarities hetween moppets using defined rules. Later activities require the child to determine the rule based on observations. All moppets can be described by four traits namely: height (short or tall), weight (thin or fat), color of clothes (blue or red), and type (Bibbit or Gribbit). The five activities which range from easy to hard are: (1) Secret Pal; (2) Change Me; (3) Club House; (4) Moptown Map; (5) Moptown Hotel. The first two games are done in low resolution graphics (semi-graphics mode) while the remaining games are in four-color, hi-res graphics. A brief description of each game follows:

1. **Change Me:** This game requires the child to change moppet #1 into moppet #4 by changing traits according to the rule, i.e., change either one or two traits at a time.
2. **Secret Pal:** This game is similar to the popular game Mastermind. The child must guess the secret pal by naming its traits. The computer draws the moppet according to the specified traits and places one yellow mark for each correct trait but does not inform the child which traits are correct. Through a process of deduction, the secret pal can be guessed.
3. **Clubhouse:** The idea of this game is to guess the membership rule for which moppets can enter the clubhouse, e.g. only blue moppets. The child rescribes the traits of a moppet which he thinks belongs in the clubhouse. If the moppet meets the rule it is placed in the clubhouse; otherwise, an X is placed next to the moppet. By observing which moppets enter the clubhouse, theehild soon determines the clubhouse rule.
4. **Moptown Map:** In this game a neighborhood of Moptown is drawn and the child must place moppets into their houses according to the neighborhood rules. For example, eaeh row or eolumn has a rule such as only short blue moppets in a row. Using the arrow keys the child places moppets in each house. By observing which moppets are correctly placed, the child soon learns the neighborhood rules.
5. **Moptown Hotel:** In this final game moppets are assigned to hotel rooms according to floor and eolumn rules. To make it interesting, this activity is a game for two children. Each child earns "money" by placing the correct moppet in a room next to occupied rooms. Before starting the game, the children may decide the hotel rules within the limits of one to three trait differences per row or column.

All of the programs ran smoothly with no hitches. With the disk version, operation is controlled by a master menu. At any time control can be returned to this menu by pressing the CLEAR key; however, in each case one can return to the program by answering "no" to the prompt to continue. This prevents accidentally eanecliing the program by hitting the CLEAR key. Instructions are provided in each program at the beginning and also when the "?" is pressed. This, of course, avoids the need to refer to the manual. It should also be noted that very little typig ability is required of the child. Other than typing his/her name, only one key responses are required by the child, e.g., S or T for short or tall.

From our discussion thus far it is obvious that the Moptown series is designed to teach the conecpts of deductive reasoning. Deductive skills are enforced by using the conerete example of the moppet to deduce an abstract rule. The use of the moppet eharacters rather than abstract geometric shapes, for example, makes the process of learning so much more fun. Children can identify with the moppets in houses or clubhouses or hotels, etc. much more easily than trying to distiguish between abstract shapes.

The documentation is sufficient since the instructions arc included in each program as noted above. However, there might have been more discussion for parents on the concepts of logical thinking. This would help parents to provide guidance to their children. For example, the manual suggests making a deck of moppet cards with some possible games. This is an exeellent idea which allows the learning process to continue while mom and dad use the CoCo to blow up spaee nasties. Overall we rate the *Moptown Series* as excellent—you can tell your child you're buying a game and he/she will never know it's really an educational game.

(*Moptown Hotel* by Follett Library Book Co., 4506 Northwest Highway, Crystal Lake, IL 60014, tape $30, disk $35)

—Pat, Don, and Chris Dollberg

Beyond Words Is Beyond Words

Beyond Words consists of three separate language arts tutors from Computer Island written for children and young adults, grades three through 12. After *CLOADing* this 32K Extended BASIC program, the user must ENTER *PCLEAR 1* before proceeding to *RUN*. At this point the student is asked to enter his/her name and tell if the printer is turned on. The child then chooses one of three subtests. Each is a three-part, menu-driven program and the user can practice one skill and then go on to another at the end of a round. The incorrect questions and answers are displayed, and if the printer is turned on, a list of items to study is generated. If a particular subtest is used for five rounds, a total score is given.

Beyond Words 1, 2 and 3 are written for children in grades three to five, grades six to eight, and grades nine to 12 respectively. All have spelling subtests in which a correct and an incorrect spelling of a word is given. The child then types in and ENTERs the word that he thinks is spelled correctly. Fourth graders who tested it really liked the idea of having to type in a word, rather than keying in the number of the answer. That age groups seems to enjoy a program that allows them to use the computer as much as possible.

A second subtest in each program deals with antonyms and synonyms. In this part, two words are shown. The child has to decide if the meaning of the words are the same (S) or opposite (O).

It is the third subtest that is different in each program. In *Beyond Words* 1, it is short forms. This subtest covers abbreviations and contractions. The long form is given and dashes are shown to indicate the length of the answer. The child is then to type in the short form. In *Beyond Words* 2, there is a subtest in homonyms. A word is given, and again dashes indicate the length of the answer. The child has to type in a word that sounds exactly like the given one.

Beyond Words 3's third subtest is analogies. Two pairs of words are given and the user has to decide if the relationship of the first pair is the same as the relationship of the second pair. In other words, is the analogy true (T) or false (F)?

All three programs have 400 questions each. The questions are randomly selected and in our testing, we found hardly any repeats with each loading.

The words used in the program are all contained in *DATA* statements. The documentation suggests that the program can be modified by the user and gives very complete directions on how to change the *DATA* statements so the program's use really becomes infinite. The modification directions are clearly written and very easy to follow.

We had many children of all different ages use the program that was suited to their particular age group. The reactions and comments about the programs were all very favorable. All of the children, especially the younger ones, enjoyed the amount of interaction that they had with the computer. My daughter, Shari, who is in the eighth grade, found the subtests for her level very challenging. Very often she complains that "educational" software written for her age group is too easy.

(Computer Island, 227 Hampton Green, Staten Island, NY 10312, tape $19.95 each)

—Stephanie Snyder

Phonics I: Teaching Tool With Limitations

Phonics I is an attempt to use the computer, combined with a voice track on tape, to teach the initial consonant blend sounds of words. The tape version, which I received for review, consists of two tapes, one a tutorial, the other a test. The computer is used to synchronize the playing of the cassette recorder with the screen display. I presume the disk version is similar.

In the tutorial, a pleasant male voice first pronounces the blend and asks you to repeat it after him, while the letters which make the sound are shown on the screen in large lower case. The voice says, for example, "Blue starts with bl." This is repeated for 22 separate blends.

Due to the sequential nature of such a presentation it is impossible to isolate only a few sounds for study at any given time, nor can the order be changed. This severely restricts the usefulness of the tutorial as a teaching tool, as the usual practice is to first master a few sounds and then proceed to new material.

The student might be better served in the initial learning stage by a simple voice tape combined with an illustrated booklet showing the letters, and a few words in which the blend occurs. Such tapes are easy to prepare, and most children are familiar with the use of a beep tone to indicate that the page is to be turned. Only after the child is familiar with most of the blends used in this program would the computer tutorial be helpful for review and reinforcement. Some children do have great difficulty associating spoken sounds with the printed word, and for them it is helpful to have a variety of audio visual methods available. The excitement of using the computer for this type of task may provide that extra boost to learning needed.

The testing program is very similar to the tutorial program. The tape voice first explains that the child is to wait for the appearance of a large question mark on the screen before answering. Then the tape voice says, for example, "What are the first two letters in the word black?" (All words used are different from those used in the tutorial portion.)

When the question mark appears, the child is to key in his response. That, for the young child, is easier said than done. It takes the inexperienced child a very long time to find the correct two letters. Unfortunately, he is deprived of the thrill of seeing his work on the screen, for what he keys in is not displayed. No provision is made for him to correct his answer in case he hits the wrong key, or hits two adjacent keys simultaneously. If he does succeed, he is rewarded with a happy face, and if his response is incorrect a sad face is shown and a low tone sounds.

The children who tried the program for me were disappointed that they could not see what they had typed in, but were very happy when they were correct. After the appearance of one of the two faces, the correct two letters are shown. The non-punitive nature of computer testing and drill programs does encourage repetition until success is achieved, but here, where the same words and the same order must be followed, a child who works with it many times may memorize the order of responses, rather than learn the material.

The limitations inherent in this type of program suit it best, in my opinion, to the child who has already learned most of the blends presented, but who needs an extra push. At the present time, when using a computer is, for many, rewarding in itself, this program may help. Although the documentation accompanying the program asserts that it is designed to tutor children in basic reading skills, it might have application in the teaching of spoken English to the foreign born, for whom letter recognition and typing are less problematical.

(Prickly-Pear Software, 9822 E. Stella Road, Tucson, AZ 85701, $24.95)

—Carol Kueppers

Hint . . .

Print Out Disk Directory

If you have a long disk directory and want to see all of it, or if you simply wish to have a hard-copy printout of your directory, one simple command will allow you to do this easily.

Just POKE 111,254:DIR and the entire disk directory will appear on your printer, even it if is too long to be fully displayed on the screen.

VDOS
The Case Of The Hidden Disk

By Frank J. Esser

VDOS stands for "A Virtual Disk Operating System." The word virtual means that an item takes on an appearance of something other than what it really is. The term "virtual memory" has been used to describe the manner in which some of the large mainframe computers handle memory mapping. However, as the 16-bit professional computers become more sophisticated, the term will start to be applied to them also. In the case of the large mainframes, the term virtual memory referred to the fact that the programmer had available to him more memory in which to run his program than was apparent. In the case of a couple of vendors, his program, in essence, could be as large as the disk that stored them and still run. To the programmer it appeared as if he had all the memory he could ever wish for. How does that apply to *VDOS*, you ask? Well, *VDOS* provides the user with a virtual disk system. *VDOS* will take half of user memory for the disk system data storage. This means that in a 32 or 16K system, you will loose half of the available memory to the system. However, if you have a 64K system, *VDOS* will take the 32K of memory under the ROMs and use it. Thus the user will only loose the 2K needed to hold the *VDOS* system itself. If you already have a disk drive, *VDOS* can be used as a supplement to it and is completely compatible with it.

VDOS comes on a cassette tape and is autoloading. That is, you are required to only type in *CLOADM* and wait. *VDOS* requires a 16K, 32K or 64K computer and a cassette recorder. Extended BASIC is not required. The documentation consists of 16 pages of single-sided, typewritten material. The manual is well-written and easy to follow. It took me approximately five minutes to get *VDOS* up and running. The first section of the manual is devoted to explaining just what *VDOS* is and what it will do in very general terms. The second section is the introduction. It is devoted to giving the user a more in-depth description of *VDOS*. The third section is a tutorial on the use of *VDOS* and will walk the user through all its functions. There is an appendix which contains the necessary information which explains how to force *VDOS* to reserve more than half the available memory if one so wishes. Also included is the manner in which *VDOS* can be disconnected if and when the user wishes to do so. The last section is a quick reference section listing the *VDOS* commands and how to use them.

Placing the cassette in the recorder, typing *CLOAM* and waiting will produce the start-up message for *VDOS*. The message is:

VDOS 1.0 COPYRIGHT (C) 1983 BY
DR. PREBLE'S PROGRAMS
TYPE VDOS (ENTER) FOR MENU

END OF USER MEMORY=32642

Typing *VDOS* ENTER gets the main menu which has the following format:

***VDOS* Menu**

1) DIRECTORY
2) SAVE BASIC PROGRAM
3) LOAD BASIC PROGRAM
4) SAVE BINARY FILE
5) LOAD BINARY FILE
6) KILL A BASIC PROGRAM
7) KILL A BINARY FILE
8) EXIT TO BASIC

Let us examine each of these commands individually. The DIRECTORY command works much like Radio Shack's disk BASIC command *DIR*. Exercising this option will display the programs already cataloged (stored) on the memory disk system. The printout has the following format:

VDOS DIRECTORY

TYPE NAME SIZE LOAD EXEC

==========

BAS: CCCALC 7261

30407 BYTES REMAIN FOR STORAGE

PRESS ANY KEY TO CONTINUE

The load and size categories are for machine language programs.

The SAVE BASIC PROGRAM option will store your BASIC programs in the system. The program must first be loaded into regular memory either from tape or disk. It can be *RUN* or just loaded, it doesn't really matter. When you are ready to save the program, type VDOS and press ENTER. *VDOS's* main menu will appear, wherein you select option 2. You are then asked for the filename you wish to use. Upon pressing ENTER, the return is almost instantaneous. In fact, the return is so fast that you may tend to think it did not work, but it did. Welcome to the world of bulk storage. Upon listing the directory contents, you will indeed see that your program has been cataloged.

The LOAD BASIC PROGRAM is just the opposite of the previous command. This option will fetch your program from the *VDOS* storage area and put it in the user memory area, ready to run. The response here is just as fast as the previous command.

SAVE BINARY FILE and LOAD BINARY FILE are the same as their BASIC counterparts, except that you must use the proper load procedures to get them into regular memory from either disk or cassette.

KILL A BASIC PROGRAM will do just that. If there is a BASIC program cataloged on the system, using this option will allow you to delete it.

KILL A BINARY FILE is the same as the above command except that it is used for binary files.

EXIT TO BASIC allows the user to return to the BASIC interpreter and proceed as if *VDOS* does not exist.

VDOS is a 2K virtual disk operating system which will make maximum use of your computer's memory. Although *VDOS* will work with any system, and I think there are many, many applications for the 16K and 32K systems, it really shines on a 64K system. *VDOS* will put to work that hidden 32K of memory that lies under the ROMs. The memory that just sits there doing nothing in most cases. *VDOS* has a command set which will allow you to load and save programs, delete both BASIC and binary programs and list the directory. There is a last command which will return the user to the BASIC interpreter; it remains invisible until you type the command *VDOS*. All BASIC functions perform as before. *VDOS* runs completely independently. It can be exited and reentered at will. It is not removed from memory until you remove it. Which is also a very simple procedure. There is no interaction between it and RS BASIC. *VDOS* is an excellent addition to any Color Computer library. It allows the user to store a number of programs in high memory, to be recalled and used at anytime the user wishes. I have three disk drives on line and use my system for many business applications. Most of these applications I wrote myself and I sure wish that *VDOS* had been available when I was developing and debugging them. It would have saved me countless hours of program swapping time. I see tremendous use for this system if properly applied. If yours is a cassette-based system and you have a number of programs stored on the tape you are working on, you know what a problem the management of those programs can be. Well, *VDOS* can take care of that for you. You can load the programs one at a time into regular memory and catalog them under *VDOS*. Repeat this proeedure as many times as needed, provided there is sufficient memory available. When you have finished, simply reload them to memory from *VDOS* and resave back to tape. What could be easier? If you already have a disk system, *VDOS* will save the time required to load from disk since *VDOS's* response is almost instantaneous. In either case, there are some very definite advantages to using *VDOS*. Couple this with the fact that in a 64K machine it runs almost completely out of your way. You will give up only 2K of user memory to it.

In conclusion, I found *VDOS* to be a program of excellent quality and with tremendous potential. The author of *VDOS* promises that a set of utilities to expand its capabilities will be available in the near future. I have a couple of items which I would like to throw into the pot of future utilities. It would be nice if you could load and run a program under *VDOS* from another program, as can be done with RS Disk BASIC. Also, it would be neat if you could go directly from *VDOS* to tape or disk on a singular program basis. And perhaps a later version would be able to address one of the computer ports for greater storage capabilities. *VDOS* has limited storage, but with a little care, it should do the job very well. After all, 32K of programs is quite a bit. I believe that this program should be in the library of every serious CoCo user, whether you have a cassette or disk-based system.

*(An accompanying utility, **VDUMP**, is now available from the author and will allow you to load or save an entire bank of programs with a single command.)*

(Dr. Preble's Programs, 6540 Outer Loop, Louisville, KY 40228, $49.95 plus $1.50 s/h)

Gathering information for a term paper? Preparing a report or lecture?

Let *Bibliography File* Help With That Term Paper!

By Larry Konecky

Students and teachers, here is a data program written specifically with some of your needs in mind. Bibliography information and related notes can be stored on disk for quick access and recall. Information is filed similar to a 3 x 5 index card record keeping system. Each record contains fields for entering a subject, tag, author, title, publication, volume or address, date, page numbers, and nine lines of notes. The subject, tag, and author fields are stored in a sequential file in alphabetical order by subject and by author within each subject. All other fields are handled through direct access filing.

Included are a variety of formats for accessing, displaying, and printing information contained in your file. Records may be accessed by subject, tag, or author, or any combination of the three. Records may also be accessed by specific title. Typing the word "ALL" in place of a specific title, subject, tag, or author will allow tabbing through all records under specific fields or combination of fields. Records displayed can be sent to a printer in one of two formats (bibliography information only or the complete record).

A listing mode provides various formats for listing accessible fields contained in your file. Subject field headings can be displayed or printed along with a

number indicating the number of records filed under each subject. Tags, authors, or titles under specific subjects; or tags, authors, or titles for all subjects may be listed. Tag and author modes allow selection of either screen or printer for listing while titles can be sent only to a printer. Another listing mode will list to a printer a combination of all subject, tag, and author fields.

This program is written for a 32K Color Computer with disk drive. It will not work as written with cassette as direct access filing is used to handle data. To make full use of the program a printer is necessary. (I use a Line Printer VII.)

A PCLEAR 1 is used to set aside RAM for program use. In the first *RUN* after your system is turned on this statement often causes errors. To alleviate these problems perform a PCLEAR 1 before your first *RUN*.

After *RUN* is entered the following "initial" menu will appear:

1) Continue Existing File
2) Begin New File (Disk)

If you are beginning a new disk, select 2. If you already have a file created on the disk in your drive, select 1. If you select 2 and then save a record to disk on a disk already containing records you will lose access to those records. Only select 2 if you are beginning a new file on a fresh disk or no longer have a need for records presently on a disk. You may want to create separate files (disks) for various subject areas if you expect a large collection of records. There is room for 140 records on each disk. It is helpful to have a copy of the program on each disk being used for a file.

After choosing to continue with an existing file the following "main" menu will appear:

1) ADD
2) SEARCH
3) LIST
4) END

(If you had selected to begin a new disk from the initial menu, you would automatically have been placed in the ADD mode.)

ADD allows addition of new records to your file and consists of four pages for entering information; one page for entering bibliography information and three for entering notes. The first page seen allows entering the following bibliography information by pressing indicated keys:

Key	Bib Info	Length
S	Subject	15
T	Tag	12
A	Author	30
1	Title	80
2	Title continued	80
P	Publication or Publisher	30
V	Volume or Address	30
D	Date	12
G	Page Numbers	15

Each field is limited to a specified amount of characters as shown in the right hand column above.

Notes are entered through each of the other three pages. Up to nine lines of 75 characters each may be used. Each line is entered by accessing its corresponding page and selecting a line number from one through nine (three to a page).

Below each page appears a menu of other selectable functions as shown below:

'C'=CONTINUE 'E'=DELETE 'K'=SAVE
'R'=RETURN 'L'=PRINT 'B'=BACK

Selecting between C and B allows movement between each of the four pages. If you press C on the fourth page you will be asked if you want to save the present record to disk. Press Y to save and then return to the main menu. N will return operation to the main menu without saving the current record to disk. Pressing R on any page will initiate this same action. Press K to save the current record to disk from any page and then return to the main menu. Any time a record is saved, it is first alphabetized by subject and by author within each subject. The subject, tag, and author fields are stored in a sequential file along with an assigned number. This number is used for direct access filing of the remaining fields. Pressing L allows selection of one of two formats for printing records (bibliography only or the complete record). Pressing E in the ADD mode will return operation to the main menu without saving the current record to disk.

The above functions work somewhat differently in the SEARCH mode. Instead of returning to the main menu when completed a check is made for further records under the current field heading(s) being searched and any records found will be displayed. If no records remain under the selected field heading(s) you will then be returned to the main menu. If you press E while in the SEARCH mode the record being displayed will be erased from your disk file.

SEARCH allows accessing records already contained within a file. These records may be displayed on the screen as in the ADD mode or sent to a printer. The records may be accessed individually or by various groupings. The following menu appears in SEARCH mode:

1) Subject-Tag-Author
2) Title
3) Return to Main Menu

If 1 is selected, eight ways of searching are possible. You will be prompted to enter a specific subject, tag, and author one at a time. The word "ALL" may be entered if all records under particular fields are desired. The following combinations are possible:

Subject	Tag	Author
ALL	ALL	ALL
(entry)	ALL	ALL
ALL	(entry)	ALL
ALL	ALL	(entry)
(entry)	(entry)	ALL
(entry)	ALL	(entry)
ALL	(entry)	(entry)
(entry)	(entry)	(entry)

If no records are found under specific entries you will be returned to the main menu.

If you have selected the screen for displaying of records and more than one record is contained under a specific entry you will be able to page through these records one at a time. Once a complete record is paged through and left for another, you can page back to it by pressing B.

If sent to a printer all records under specific fields selected will be printed within one of the two possible formats (bibliography or complete). If more than one record is filed under specific fields, they will be printed continuously until no more are found. When printing more than one record, lines are counted and paper is automatically advanced so that records do not cross over perforations. It is a good idea to set the paper so that printing begins at the top of a sheet. If records are chosen one at a time for printing this automatic advance will not occur.

You may select specific titles to search for, or "ALL" titles. When selecting a specific title you need type in only the first 32 characters. If a title is less than 32 characters you will need to press the space bar enough times to use up a total of 32 characters for a proper search.

When in SEARCH mode and displaying records to the screen, if you have made no changes to records being displayed you need not resave them to disk. If editing has taken place and you want changes set on disk, press K to save, or Y when asked.

Selecting *LIST* from the main menu will display the following menu:

1) Subjects
2) Tags
3) Authors
4) Subjects-Tags-Authors
5) Titles
6) Return

Press 1 to list all subject headings in alphabetical order and also the number of records contained under each. This list may be sent to either screen or printer. Selecting 2 or 3 will list tags or authors by subject. A specific subject may be selected or "ALL" may be selected. Again either screen or printer may be used. Modes 4 and 5 will be sent to the printer only. Pressing 4 will list to a printer all subjects, tags, and authors in alphabetical order by subject and by author within each subject. Titles may be listed by specific subject heading or all of them may be listed by typing "ALL" for the subject entry. Titles will be listed in the same sequential order as in the other listing modes.

Editing is accomplished through either ADD or SEARCH modes. It works in the same manner as entering information. Once you have selected a field to edit and the cursor appears on the screen in that particular field you are committed. If you have made a mistake and selected an incorrect field you will need to retype that field. Just pressing ENTER without retyping will erase any previous entry from RAM. Correcting a character error in any one field necessitates retyping that particular field and then resaving the record to disk.

When completing a session I suggest you END by going to the main menu and pressing 4. This will insure that all files are properly closed. You will find that BREAK is disabled in certain parts of the program to encourage use of the END routine.

(Larry Konecky holds a master's degree in guitar performance and is working on a Doctor of Music Education degree. He is currently Director of CAI in Music at Alcorn State University.)

The listing:

260	0236	3165	1A97
330	041A	3355	1D4C
625	06F8	4020	1FD2
1010	09D0	4110	22B9
1405	0C29	5120	25E8
1590	0F4E	5305	28C7
1830	11DE	5410	2B3A
2081	1532	5690	2DFC
2225	1803	END	3039

```
1 IF PEEK(&H7EB9) <> &H32 THEN C
LEAR 200,&H7EB0:FORI=&H82B9 TO &
H831E:POKEI-&H400,PEEK(I):NEXT E
LSE5
2 FORI=0TO2:POKE&H7EBD+I,18:NEXT
```

```
:I=&H7F1E
3 POKEI,&H26:POKEI+1,3:POKEI+2,&
H7E:POKEI+3,&H83:POKEI+4,&H22:PO
KEI+5,&H7E
4 POKEI+6,&HA4:POKEI+7,&H4C
5 POKE&H19B,&H7E:RUN6
6 PCLEAR1:CLEAR8000:FILES6,1000
10 DIM S$(140),T$(140),A$(140),R
(140)
200 R$=CHR$(255):BC$=STRING$(32,
159):SX$=STRING$(80,"-")
210 SC$=STRING$(64,191):SG$=STRI
NG$(32,"-")
220 '*
221 '* OPEN BUFFERS
222 '*
235 OPEN"D",#1,"TITLE/DAT",160
240 FIELD#1,80AST1$,80AST2$
245 OPEN"D",#2,"PUB/DAT",87
250 FIELD#2,30ASP$,30ASV$,15ASG$
,12ASD$
255 OPEN"D",#3,"INF01/DAT",225
260 FIELD#3,75ASI1$,75ASI2$,75AS
I3$
265 OPEN"D",#4,"INF02/DAT",225
270 FIELD#4,75ASI4$,75ASI5$,75AS
I6$
271 OPEN"D",#5,"INF03/DAT",225
272 FIELD#5,75ASI7$,75ASI8$,75A8
I9$
275 LSETT1$="":LSETT2$=""
280 LSETP$="":LSETV$="":LSETG$="
":LSETD$=""
285 LSETI1$="":LSETI2$="":LSETI3
$="":LSETI4$="":LSETI5$="":LSETI
6$=""
286 LSETI7$="":LSETI8$="":LSETI9
$=""
300 CLS3
305 PRINT@6," *BIBLIOGRAPHY FILE
* ";
310 PRINT@66," SELECT FROM THE F
OLLOWING ";
315 PRINT@162," 1) CONTINUE EXIS
TING FILE ";
320 PRINT@194," 2) BEGIN NEW FIL
E (DISK)  ";
325 I$=INKEY$:IFI$=""THEN325
326 IFI$="1"THEN400
327 IFI$="2"THENMS=1:GOTO605
330 GOTO310
400 OPEN"I",#6,"SUBAUTH":X=1
405 INPUT #6,S$(X)
406 INPUT #6,T$(X)
407 INPUT #6,A$(X)
408 INPUT #6,R(X)
409 IF EOF(6)=-1THEN415
410 X=X+1:GOTO405
415 CLOSE#6
500 D=0:C=0:CLS3:PRINT@6," *BIBL
```

```
IOGRAPHY FILE* ";
501 LSETT1$="":LSETT2$=""
502 LSETP$="":LSETV$="":LSETG$="
":LSETD$=""
503 LSETI1$="":LSETI2$="":LSETI3
$="":LSETI4$="":LSETI5$="":LSETI
6$=""
504 LSETI7$="":LSETI8$="":LSETI9
$="":S$="":T$="":A$=""
505 PRINT@66," SELECT FROM THE F
OLLOWING ";:L=0
510 PRINT@168," 1) ADD      ";
515 PRINT@200," 2) SEARCH   ";
520 PRINT@232," 3) LIST     ";
521 PRINT@264," 4) END      ";
522 I$=INKEY$:IFI$=""THEN522
525 IFI$="1"THEN MS=1:GOTO600
526 IFI$="2"THEN MS=2:GOTO3000
527 IFI$="3"THEN MS=3:GOTO5000
528 IFI$="4"THEN MS=4:GOTO2400
529 GOTO505
600 IFS$(X)="ZZZ"THEN700
605 X=X+1:W=X:IFX=1THENR(1)=1
610 IFX<141THEN700
615 PRINT@352,SC$;:PRINT@384,SC$
;:
620 PRINT@393," FILE FULL ";:X=X
-1
625 GOTO522
```



```
697 '*
698 '* PAGES
699 '*
700 CLS2:PRINT@0,BC$;:PRINT@8,"
*BIBLIOGRAPHY* ";:FORBG=32TO380S
TEP32:PRINT@BG,STRING$(32,143);:
NEXT
705 PRINT@32,"S)"S$;
710 PRINT@49,"(T)"T$;
715 PRINT@64,"A)"A$;
720 PRINT@96,"1)"T1$;
721 PRINT@178,R$;
725 PRINT@192,"2)"T2$;
726 PRINT@274,R$;
730 PRINT@288,"P)"P$;
735 PRINT@320,"V)"V$;
740 PRINT@352,"D)"D$;
745 PRINT@366,"(G)"G$;
750 PRINT@416,"'C'=CONTINUE 'E'=
DELETE 'K'=SAVE"
755 PRINT@448,"'R'=RETURN    'L'=
PRINT  'B'=BACK";
760 I$=INKEY$:IFI$=""THEN760
765 IFI$="S"THEN1200
770 IFI$="T"THEN1220
775 IFI$="A"THEN1240
780 IFI$="1"THEN1260
785 IFI$="2"THEN1280
790 IFI$="P"THEN1300
795 IFI$="V"THEN1320
800 IFI$="G"THEN1360
805 IFI$="D"THEN1340
810 IFI$="C"THEN1400
815 IFI$="E"THEN1100
820 IFI$="K"THEN2000
825 IFI$="R"THEN2200
830 IFI$="L"THEN1900
831 IFI$="B"THEN1000
835 GOTO760
1000 IFMS=1THEN760
1005 IFMS=2 AND Y=1THEN760ELSEY=
Y-2:GOTO3350
1010 GOTO760
1100 PRINT@416,SC$;:PRINT@418,"
DELETE! ARE YOU SURE?   ";
1105 PRINT@450," ( Y OR N ) ";
1110 Q$=INKEY$:IFQ$=""THEN1110
1115 IFQ$="N"THEN700
1120 IFQ$="Y"THEN2200ELSE1100
1200 PRINT@34,;:LINEINPUTN$:S$=L
EFT$(N$,15):C=1
1205 IFT$=""THENPRINT@49,"(T)";E
LSEPRINT@49,"(T)"+T$;
1210 GOTO760
1220 PRINT@52,;:LINEINPUTN$:T$=L
EFT$(N$,12)
1230 GOTO760
1240 PRINT@66,;:LINEINPUTN$:A$=L
EFT$(N$,30):C=1
1250 GOTO760
```

```
1260 PRINT@98,;:LINEINPUTN$:LSET
T1$=N$
1270 PRINT@178,R$;:GOTO760
1280 PRINT@194,;:LINEINPUTN$:LSE
TT2$=N$
1290 PRINT@274,R$;:GOTO760
1300 PRINT@290,;:LINEINPUTN$:LSE
TP$=N$
1310 GOTO760
1320 PRINT@322,;:LINEINPUTN$:LSE
TV$=N$
1330 GOTO760
1340 PRINT@354,;:LINEINPUTN$:LSE
TD$=N$
1345 PRINT@366,"(G)"G$;
1350 GOTO760
1360 PRINT@369,;:LINEINPUTN$:LSE
TG$=N$
1370 GOTO760
1400 CLS6:PRINT@0,SC$;
1405 PRINT@2,S$;:PRINT@19,T$;:PR
INT@34,A$;:PRINT@64,SG$;
1410 PRINT@96,"1)"I1$;
1411 PRINT@173,R$;
1415 PRINT@192,"2)"I2$;
1416 PRINT@269,R$;
1420 PRINT@288,"3)"I3$;
1421 PRINT@365,R$;
```



```
1425 PRINT@416,"'C'=CONTINUE 'E'
=DELETE 'K'=SAVE"
1430 PRINT@448,"'R'=RETURN   'L'
=PRINT  'B'=BACK";
1435 I$=INKEY$:IFI$=""THEN1435
1440 IFI$="1"THEN1600
1445 IFI$="2"THEN1610
1450 IFI$="3"THEN1620
1455 IFI$="C"THEN1500
1460 IFI$="E"THEN1100
1465 IFI$="K"THEN2000
1470 IFI$="R"THEN2200
1475 IFI$="L"THEN1900
1485 IFI$="B"THEN700
1490 GOTO1435
1500 CLS6:PRINT@0,SC$;
1505 PRINT@2,S$;:PRINT@19,T$;:PR
INT@34,A$;:PRINT@64,SG$;
1510 PRINT@96,"4)"I4$;
1511 PRINT@173,R$;
1515 PRINT@192,"5)"I5$;
1516 PRINT@269,R$;
1520 PRINT@288,"6)"I6$;
1521 PRINT@365,R$;
1525 PRINT@416,"'C'=CONTINUE 'E'
=DELETE 'K'=SAVE"
1530 PRINT@448,"'R'=RETURN   'L'
=PRINT  'B'=BACK";
1535 I$=INKEY$:IFI$=""THEN1535
1540 IFI$="4"THEN1630
1545 IFI$="5"THEN1640
1550 IFI$="6"THEN1650
1555 IFI$="C"THEN1800
1560 IFI$="E"THEN1100
1565 IFI$="K"THEN2000
1570 IFI$="R"THEN2200
1580 IFI$="L"THEN1900
1585 IFI$="B"THEN1400
1590 GOTO1535
1600 PRINT@98,;:LINEINPUTN$:LSET
I1$=N$
1605 PRINT@173,R$;:GOTO1435
1610 PRINT@194,;:LINEINPUTN$:LSE
TI2$=N$
1615 PRINT@269,R$;:GOTO1435
1620 PRINT@290,;:LINEINPUTN$:LSE
TI3$=N$
1625 PRINT@365,R$;:GOTO1435
1630 PRINT@98,;:LINEINPUTN$:LSET
I4$=N$
1635 PRINT@173,R$;:GOTO1535
1640 PRINT@194,;:LINEINPUTN$:LSE
TI5$=N$
1645 PRINT@269,R$;:GOTO1535
1650 PRINT@290,;:LINEINPUTN$:LSE
TI6$=N$
1655 PRINT@365,R$;:GOTO1535
1660 PRINT@98,;:LINEINPUTN$:LSET
I7$=N$
1665 PRINT@173,R$;:GOTO1835
```

```
1670 PRINT@194,;:LINEINPUTN$:LSE
TIB$=N$
1675 PRINT@269,R$;:GOTO1835
1680 PRINT@290,;:LINEINPUTN$:LSE
TI9$=N$
1685 PRINT@365,R$;:GOTO1835
1800 CLS6:PRINT@0,BC$;
1805 PRINT@2,S$;:PRINT@19,T$;:PR
INT@34,A$;:PRINT@64,SB$;
1810 PRINT@96,"7)"I7$;
1811 PRINT@173,R$;
1815 PRINT@192,"8)"I8$;
1816 PRINT@269,R$;
1820 PRINT@288,"9)"I9$;
1821 PRINT@365,R$;
1825 PRINT@416,"'C'=CONTINUE 'E'
=DELETE 'K'=SAVE"
1830 PRINT@448,"'R'=RETURN    'L'
=PRINT   'B'=BACK";
1835 I$=INKEY$:IFI$=""THEN1835
1840 IFI$="7"THEN1660
1845 IFI$="8"THEN1670
1850 IFI$="9"THEN1680
1855 IFI$="C"THEN2200
1860 IFI$="E"THEN1100
1865 IFI$="K"THEN2000
1870 IFI$="R"THEN2200
1875 IFI$="L"THEN1900
```



```
1880 IFI$="B"THEN1500
1885 GOTO1835
1900 PRINT@416,SC$;:PRINT@418,"(
B)IBLIOGRAPHY OR (C)OMPLETE";
1910 I$=INKEY$:IFI$=""THEN1910
1920 IFI$="B"THENP=1:GOTO4025
1930 IFI$="C"THENP=2:GOTO4025
1940 GOTO1910
1990 '*
1991 '* ALPHABETIZE AND SAVE
1992 '*
2000 PRINT@416,SC$;:IFMS=1THEN20
20
2005 IFMS=2 ANDC=1THEN2100
2010 IFMS=2AND C=0THENT$(Y)=T$:Z
=Y:GOTO2060
2020 IFS$(X)="ZZZ"THENWW=R(X)ELS
EWW=X
2021 IFX=1THENZ=1:GOTO2045
2025 FORZ=X TO1STEP-1
2030 IFZ=1THEN2045
2035 IFS$<S$(Z-1)THENS$(Z)=S$(Z-
1):T$(Z)=T$(Z-1):A$(Z)=A$(Z-1):R
(Z)=R(Z-1):NEXTZ
2040 IFS$=S$(Z-1)ANDA$<A$(Z-1)TH
ENS$(Z)=S$(Z-1):T$(Z)=T$(Z-1):A$
(Z)=A$(Z-1):R(Z)=R(Z-1):NEXTZ
2045 S$(Z)=S$:T$(Z)=T$:A$(Z)=A$
2050 IFMS=1THENR(Z)=WW ELSER(Z)=
W
2060 PUT#1,R(Z)
2065 PUT#2,R(Z)
2070 PUT#3,R(Z)
2075 PUT#4,R(Z)
2076 PUT#5,R(Z)
2080 OPEN"O",#6,"SUBAUTH"
2081 FORZ=1TOX
2082 WRITE#6,S$(Z)
2083 WRITE#6,T$(Z)
2084 WRITE#6,A$(Z)
2085 WRITE#6,R(Z)
2086 NEXTZ
2087 CLOSE #6
2089 IFD=1THEND=0:Y=Y-1
2090 IFMS=2 AND I$="R"THEN500
2092 IFMS=2THEN3350ELSE500
2100 W=R(Y)
2105 FORV=Y TO(X-1)
2110 S$(V)=S$(V+1):T$(V)=T$(V+1)
:A$(V)=A$(V+1):R(V)=R(V+1)
2120 NEXTV
2121 IFI$="K"THEN2020
2122 IF(I$="C" OR I$="R")AND Q$=
"Y"THEN2020
2124 S$(V)="ZZZ":T$(V)="":A$(V)=
"ZZZ":R(V)=W
2126 LSETP$="":LSETV$="":LSETG$=
"":LSETD$="":LSETT1$="":LSETT2$=
""
2127 LSETI1$="":LSETI2$="":LSETI
```

```
3$="":LSETI4$="":LSETI5$="":LSET
I6$="":LSETI7$="":LSETI8$="":LSE
TI9$=""
2129 IFD=1THEN2080
2130 GOTO2020
2200 IFMS=2 AND I$="E"THEND=1:GO
TO2100
2205 IFMS=1 AND I$="E"THEN X=X-1
:GOTO500
2210 PRINT@416,SC$;:PRINT@418,"
DO YOU WISH TO SAVE THIS  ";:PRI
NT@450," RECORD? (Y OR N)  ";
2215 Q$=INKEY$:IFQ$=""THEN2215
2220 IFQ$="N" AND MS=2 AND I$="C
"THEN 3350
2221 IFQ$="N" AND MS=2 AND I$="R
" THEN500
2225 IFQ$="N" AND MS=1 THENX=X-1
:GOTO500
2230 IFQ$="Y" AND MS=1 THEN2000
2235 IFQ$="Y" AND MS=2THEN 2100
2250 GOTO2215
2400 CLOSE
2410 END
3000 CLS4
3005 PRINT@5," *BIBLIOGRAPHY SEA
RCH* ";
3010 PRINT@66," SELECT FROM THE
FOLLOWING ";
3015 PRINT@163," 1) SUBJECT-TAG-
AUTHOR  ";
3020 PRINT@195," 2) TITLE
      ";
3025 PRINT@227," 3) RETURN TO MA
IN MENU  ";
3030 I$=INKEY$:IFI$=""THEN3030
3031 IFI$="1"THEN3100
3032 IFI$="2"THEN3400
3033 IFI$="3"THEN500
3035 GOTO3010
3100 CLS4
3105 PRINT@5,"*BIBLIOGRAPHY SEAR
CH* ";
3110 PRINT@64,"*INPUT SUBJECT: "
;
3115 PRINT@96,"  ";:LINEINPUTSE$
3120 PRINT@160,"*INPUT TAG: ";
3125 PRINT@192,"  ";:LINEINPUTTE
$
3130 PRINT@256,"*INPUT AUTHOR: "
;
3135 PRINT@288,"  ";:LINEINPUTAE
$
3140 PRINT@356," (S)CREEN OR (P)
RINT? ";
3150 I$=INKEY$:IFI$=""THEN3150
```

```
3155 IFI$="S"THENA=2:GOTO3190
3160 IFI$="P"THENA=1:GOTO3170
3165 GOTO3150
3170 PRINT@353,"(B)IBLIOGRAPHY O
R (C)OMPLETE?";
3175 I$=INKEY$:IFI$=""THEN3175
3177 IFI$="B"THENP=1:GOTO3185
3180 IFI$="C"THENP=2ELSE3175
3185 PRINT@352,SC$;:PRINT@358,"
***  PRINTING  *** ";
3190 IFSE$="ALL" AND TE$="ALL" A
ND AE$="ALL" THEN3200
3191 IFSE$="ALL" AND TE$="ALL" T
HEN3320
3192 IFSE$="ALL" AND AE$="ALL" T
HEN3300
3193 IFTE$="ALL" AND AE$="ALL" T
HEN3280
3194 IFSE$="ALL" THEN3260
3195 IFTE$="ALL" THEN3240
3196 IFAE$="ALL" THEN3220
3199 FORY=1TOX:IFSE$=S$(Y) AND T
E$=T$(Y) AND AE$=A$(Y) THEN4000E
LSE3350
3200 FORY=1TOX:GOTO4000
3220 FORY=1TOX:IFSE$=S$(Y)ANDTE$
=T$(Y)THEN4000ELSE3350
3240 FORY=1TOX:IFSE$=S$(Y) AND A
E$=A$(Y)THEN4000ELSE3350
3260 FORY=1TOX:IFTE$=T$(Y) AND A
E$=A$(Y)THEN4000ELSE3350
3280 FORY=1TOX:IFSE$=S$(Y)THEN40
00ELSE3350
3300 FORY=1TOX:IFTE$=T$(Y)THEN40
00ELSE3350
3320 FORY=1TOX:IFAE$=A$(Y)THEN40
00ELSE3350
3350 IFMS=2THENNEXTY
3355 GOTO500
3400 CLS4
3410 PRINT@1," *BIBLIOGRAPHY TIT
LE SEARCH* ";
3420 PRINT@64,"*INPUT TITLE: ";
3430 PRINT@96,"(ONLY FIRST 32 CH
ARACTERS) ";
3440 PRINT@128,;:LINEINPUTTE$
3450 PRINT@356," (S)CREEN OR (P)
RINT? ";
3455 I$=INKEY$:IFI$=""THEN3455
3460 IFI$="S"THENA=2:GOTO3495
3465 IFI$="P"THENA=1:GOTO3475
3470 GOTO3455
3475 PRINT@353,"(B)IBLIOGRAPHY O
R (C)OMPLETE?";
3480 I$=INKEY$:IFI$=""THEN3480
3481 PRINT@352,SC$;:PRINT@358,"
***  PRINTING  *** ";
3485 IFI$="B"THENP=1:GOTO3495
3490 IFI$="C"THENP=2ELSE3480
3495 IFTE$="ALL"THEN3200
3500 FORY=1TOX:GET#1,R(Y)
3510 IFLEFT$(TE$,32)=LEFT$(T1$,3
2)THEN4000ELSE3350
3990 '*
3991 '* PRINT ROUTINE
3992 '*
4000 IFS$(Y)="ZZZ"THEN500ELSEGET
#1,R(Y)
4005 GET#2,R(Y)
4010 GET#3,R(Y)
4015 GET#4,R(Y)
4016 GET#5,R(Y)
4017 S$=S$(Y):T$=T$(Y):A$=A$(Y)
4020 IFA=2THEN700
4021 IFP=1THENPRINT#-2,TAB(20)S$
(Y);TAB(45)T$(Y)
4025 PRINT#-2,A$(Y)
4030 PRINT#-2,T1$
4035 IFLEFT$(T2$,5)="      "THENP
RINT#-2,T2$ELSEPRINT#-2,T2$
4040 PRINT#-2,TAB(3)P$;TAB(40)V$
4045 PRINT#-2,TAB(3)D$;TAB(40)G$
4046 IFP=1THEN4085
4050 PRINT#-2,TAB(10)STRING$(55,
"-")
4065 PRINT#-2,TAB(15)S$(Y);:PRIN
T#-2,TAB(50)T$(Y)
4070 PRINT#-2,TAB(10)STRING$(55,
"-")
4075 IFLEFT$(I1$,5)="      "THENP
```

```
RINT#-2ELSEPRINT#-2,TAB(3)I1$
4076 IFLEFT$(I2$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I2$
4077 IFLEFT$(I3$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I3$
4078 IFLEFT$(I4$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I4$
4079 IFLEFT$(I5$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I5$
4080 IFLEFT$(I6$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I6$
4081 IFLEFT$(I7$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I7$
4082 IFLEFT$(I8$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I8$
4083 IFLEFT$(I9$,5)="      "THENP
RINT#-2ELSEPRINT#-2,TAB(3)I9$
4085 PRINT#-2,STRING$(80,"-"):IF
P=2THENPRINT#-2
4086 IFMS=1THEN700
4090 IFP=2 AND L=2THENGOSUB4150
4095 IFP=1 AND L=8 THENGOSUB 415
0
4096 L=L+1
4100 IFMS=2THENNEXTY ELSE700
4110 GOTO500
4150 IFP=2THENFORPR=1TO9ELSEFORP
R=1TO3
4160 PRINT#-2:NEXTPR
4170 L=-1:RETURN
5000 PRINT@357," (S)CREEN OR (P)
RINT ";
5005 I$=INKEY$:IFI$=""THEN5005
5010 IFI$="S"THENA=2:GOTO5020
5015 IFI$="P"THENA=1:GOTO5020
5016 GOTO5005
5020 CLS7:PRINT@6," *BIBLIOGRAPH
Y LIST* ";
5025 PRINT@66," SELECT FROM THE
FOLLOWING ";
5030 PRINT@132," 1) SUBJECTS
      ";
5040 PRINT@164," 2) TAGS
      ";
5045 PRINT@196," 3) AUTHORS
      ";
5046 PRINT@228," 4) SUBJECT-TAG-
AUTHOR ";
5050 PRINT@260," 5) TITLES
      ";
5052 PRINT@292," 6) RETURN
      ";
5055 I$=INKEY$:IFI$=""THEN5055
5056 IFI$="1"THENLI=1:GOTO5067
5057 IFI$="2"THENLI=2:GOTO5067
5058 IFI$="3"THENLI=3:GOTO5067
5059 IFI$="4"THENLI=4:GOTO5067
5060 IFI$="5"THENLI=5:GOTO5067
```

```
5061 IFI$="6"THEN500
5067 IFA=2THEN5500
5070 ONLI GOTO5100,5200,5200,530
0,5350,500
5100 CLS5:PRINT@5," *PRINTING SU
BJECTS* ";
5105 PRINT#-2,TAB(5)"*** BIBLIOG
RAPHY SUBJECTS ***":PRINT#-2
5110 CT=0:SS$=S$(1)
5115 FORV=1TOX:S$=S$(V)
5120 IFS$<>SS$THEN5135
5125 CT=CT+1:NEXTV
5135 PRINT#-2,TAB(11)SS$TAB(26)C
T:IFS$="ZZZ"THENPRINT#-2:GOTO500
5140 SS$=S$:CT=1:NEXTV:PRINT#-2,
TAB(11)SS$TAB(26)CT:PRINT#-2:GOT
O500
5200 CLS5:PRINT@5," *BIBLIOGRAPH
Y LIST* ";
5205 PRINT@64," INPUT SUBJECT AR
EA :";
5210 PRINT@128,;:INPUTSU$:CLS5:I
FLI=2THENPRINT@8," PRINTING TAGS
 ";ELSEPRINT@7," PRINTING AUTHOR
S ";
5225 IFLI=2THENPRINT#-2,"*** BIB
LIOGRAPHY TAGS ***"ELSEPRINT#-2,
"*** BIBLIOGRAPHY AUTHORS ***"
5226 PRINT#-2
5230 IFSU$="ALL"THEN5235ELSE5275
5235 SS$=S$(1):PRINT#-2,TAB(5)S$
(1)
5240 FORV=1TOX:S$=S$(V):IFS$="ZZ
Z"THENPRINT#-2:GOTO500
5250 IFS$<>SS$THENPRINT#-2,TAB(5
)S$:SS$=S$
5255 IFLI=2THENPRINT#-2,TAB(9)T$
(V)ELSEPRINT#-2,TAB(9)A$(V)
5260 NEXTV:PRINT#-2:GOTO500
5275 PRINT#-2,TAB(5)SU$
5280 FORV=1TOX:S$=S$(V)
5285 IFSU$=S$THENIFLI=2THENPRINT
#-2,TAB(9)T$(V)ELSEPRINT#-2,TAB(
9)A$(V)
5290 NEXTV:PRINT#-2:GOTO500
5300 CLS5:PRINT@8," PRINTING LIS
T ";:PRINT#-2,TAB(25)"*** BIBLIO
GRAPHY LIST *** ":PRINT#-2
5305 FORV=1TOX:IFS$(V)="ZZZ"THEN
PRINT#-2:GOTO500
5310 PRINT#-2,TAB(13)S$(V)TAB(30
)T$(V)TAB(45)A$(V)
5315 NEXTV:PRINT#-2:GOTO500
5350 CLS4:PRINT@5," *BIBLIOGRAPH
Y TITLES* ";
5355 PRINT@64," INPUT SUBJECT AR
EA: ";
5360 PRINT@128,;:INPUTSU$:CLS5:P
RINT@7," PRINTING TITLES ";
5361 PRINT#-2,STRING$(80,"*"):PR
INT#-2,TAB(24)"*** BIBLIOGRAPHY
TITLES ***":PRINT#-2,STRING$(80,
```

```
"*"):PRINT#-2
5365 IFSU$<>"ALL"THEN5405
5370 SS$=S$(1):PRINT#-2:PRINT#-2
,TAB(15)"*****"TAB(25)S$(1)"
 *****":PRINT#-2,SX$
5375 FORV=1TOX:S$=S$(V):IFS$="ZZ
Z"THENPRINT#-2:GOTO500
5380 IFS$<>SS$THENPRINT#-2:PRINT
#-2,TAB(15)"*****"TAB(25)S$"
 *****":SS$=S$:PRINT#-2,SX$
5385 GET#1,R(V)
5390 PRINT#-2,T1$
5395 IFLEFT$(T2$,5)="      "THENP
RINT#-2 ELSEPRINT#-2,T2$
5400 NEXTV:PRINT#-2:GOTO500
5405 PRINT#-2:PRINT#-2,TAB(15)"*
****"TAB(25)SU$"      *****":PRIN
T#-2,SX$
5410 FORV=1TOX:S$=S$(V):IFS$="ZZ
Z"THENPRINT#-2:GOTO500
5415 IFSU$<>S$THEN5425
5420 GET#1,R(V):PRINT#-2,T1$:IFL
EFT$(T2$,5)="      "THENPRINT#-2
ELSEPRINT#-2,T2$
5425 NEXTV:PRINT#-2:GOTO500
5500 ONLI GOTO 5600,5700,5700,55
10,5510,500
5510 PRINT@354," ALVAILABLE ON P
RINTER ONLY ";
```



```
5520 PRINT@384,"DO YOU WANT TO P
RINT? (Y OR N)"
5530 I$=INKEY$:IFI$=""THEN5530
5540 IFI$="Y" AND LI=4THEN5300
5550 IFI$="Y" AND LI=5THEN5350
5560 IFI$="N"THEN500ELSE5530
5600 CLS5:PRINT@3," *BIBLIOGRAPH
Y SUBJECTS* ";:CO=1
5610 CT=0:SS$=S$(1)
5615 FORV=1TOX:S$=S$(V)
5620 IFS$<>SS$THEN5635
5625 CT=CT+1:NEXTV
5630 PRINT@32*CO+34,CO;TAB(7)SS$
TAB(25)CT;
5631 GOSUB5675:GOTO500
5635 PRINT@32*CO+34,CO;TAB(7)SS$
TAB(25)CT;:CO=CO+1:IFS$="ZZZ"THE
N5631
5640 SS$=S$:CT=1:IFCO=>10THENGOS
UB5675
5645 NEXTV:PRINT@32*CO+34,CO;TAB
(7)SS$TAB(25)CT;
5650 GOSUB5675:GOTO500
5675 PRINT@452," PRESS 'C' TO CO
NTINUE ";:CO=1
5680 I$=INKEY$:IFI$=""THEN5680
5685 IFI$="C"THENCLS5:RETURN
5690 GOTO5680
5700 CLS5:PRINT@5," *BIBLIOGRAPH
Y LIST* ";
5705 PRINT@64," INPUT SUBJECT AR
EA: ";
5710 PRINT@128,;:INPUTSU$:CLS5:C
O=1
5715 IFLI=2THENPRINT@7," TAGS BY
 SUBJECT ";ELSEPRINT@5," AUTHORS
 BY SUBJECT ";
5720 IFSU$="ALL"THEN5725ELSE5775
5725 SS$=S$(1):PRINT@32*CO+32,S$
(1);:CO=CO+1
5730 FORV=1TOX:S$=S$(V):IFS$="ZZ
Z"THEN5820
5735 IFS$<>SS$THENPRINT@32*CO+32
,S$"";:CO=CO+1:SS$=S$
5740 IFLI=2THENPRINT@32*CO+34,T$
(V);ELSEPRINT@32*CO+34,A$(V)
5745 CO=CO+1:IFCO=>11THENGOSUB56
75
5750 NEXTV:GOSUB5675:GOTO500
5775 PRINT@32*CO+32,SU$;
5780 FORV=1TOX:S$=S$(V)
5785 IFSU$=S$THEN5800
5790 NEXTV:GOSUB5675:GOTO500
5800 IFLI=2THENPRINT@32*CO+66,T$
(V);ELSEPRINT@32*CO+66,A$(V);
5805 CO=CO+1:IFCO=>11THENGOSUB56
75
5810 NEXTV
5820 GOSUB5675:GOTO500
```

Greetings!

It's all right to be drawing squares, triangles, decagons, and such, but what is life without circles? I thought about circles for some time, but for the life of me I couldn't figure out how to make the turtle move in a curved line. I have a little instruction booklet about Color Computer turtle graphics, so I looked for "curves" in the index, and didn't find a thing. Finally, I happened one day to be down by the pond, and what do you think I saw? I saw that old turtle that had snuck out of my house a month earlier, and he was swimming. When he left he had taken my pen with him, which I had tied to the middle of his back and let hang down behind him, so he could draw. But by now the pen had slipped around so it was caught on his hind leg. Poor turtle! Maybe Bertha was right to report me to the SPCA the way she did. In any event, that pen was causing the turtle to be off balance, so that instead of swimming in a straight line, he was swimming in circles.

I studied closely, and discovered that his circles weren't really perfect circles. Everytime he stroked with his left foot, the pen at the end acted sort of like an oar and made him turn very slightly to the right. But after that initial turn, he would go straight again. In other words, his "circle" was not really a big closed curve, but a closed, many, many sided figure—a "hundred-agon," or a "thousand-agon." I went back to my Color Computer, determined to try out this principle that so fortuitously appeared to me at the pond. (Anybody believe in fate or astrology?)

The first thing I thought was that in this new many-sided figure, I wanted to make each turning angle as small as possible. That meant each turn would be 1. I decided that each side should be as small as possible, too, so that my turtle might stay on the screen. I started with:

```
FD 1
RT 1
FD 1
RT 1
```

planning to make 360 FDs, and 360 RTs. Pretty soon, however, my fingers got tired. I thought to myself, "There must be an easier way. A computer is a labor-saving device, and I need my labor saved."

Well, there is an easier way, and I won't embarass myself by telling you how long it took me to discover it. There is a

whole 'nother section of the program where you can type in commands to the turtle and store them for later use. This next section is sometimes called "the edit mode." I call it "the edit room," because I like to think of the Color Computer turtle graphics program as a little house with rooms. Well, to get to the *EDIT* room from the *RUN* room, first press the BREAK button. That puts you back out on the house's front porch. Then press E for *EDIT*.

Now you're in the *EDIT* room and there's a little eyebrow at the bottom of the screen.

Before getting back to the circle, let's practice with something simpler. Let's start with a square. In the edit room, it's best to label what you are doing, because you will be storing that "procedure" in the computer's memory. So let's label our square procedure with something like this:

```
TO SQUARE
```

Now, you'll have to play around with the keys a little before you become comfortable with typing in the *EDIT* room, but basically the up and down and right and left arrows will help you move around as you wish. Anyhow, here is how I type out a procedure for squaring:

```
TO SQUARE
FD 40 RT 90
FD 40 RT 90
FD 40 RT 90
FD 40 RT 90
END
```

There are some things you might wish to notice about the way I have typed in the procedure. First (or last), notice that I typed in *END* at the end. It's important to inform the turtle that a procedure has ended. Second, remember that the procedure needs that label (TO SQUARE). Third, notice that I didn't use a separate line for each instruction. In the *RUN* room, I had to *ENTER* each instruction separately, automatically giving each instruction its own private line. Here, I can type several instructions per line, as long as I separate them by a space. It so happens I chose to type in only two instructions per line before starting a new line; and the only reason I did was that I realized my instructions for moving forward and turning were going to be repeated four times, and I just wanted to be able to *see* the repetition easily. So I started a new line each time I repeated the instructions to move and turn. It does make it easier to see, yes? If you wish to try this procedure out on your own, please feel free to experiment. For instance, you could type in the following:

```
TO SQUARE
FD 40 RT 90 FD 40 RT 90 FD 40 RT
90 FD 40 RT 90
END
```

In any case, let's try to run this procedure. Press BREAK, then R, and you're in the *RUN* room. Then type in SQUARE. Does it work? If so, try it a second and a third time. What happens? That's neat! The turtle zooms around the square again and again. He's drawing over and over again on the same spot. We must be doing something right. However, let's try something else. You try what you want; but here is what I want. I go back into the *EDIT* room, and erase the last angle command. I don't really need that to draw the square; all it does is turn the turtle right back into his original direction. But what will happen if he's not returned back to the original direction? Here is what the squaring procedure looks like now:

```
TO SQUARE
FD 40 RT 90
FD 40 RT 90
FD 40 RT 90
FD 40
END
```

I go back to the *RUN* room to try this out. Sure enough, it still works. I still get a square, only the turtle is not set back in his original direction. Now, if I run the squaring procedure several times, I get something more interesting than the turtle going around and around on his original path. Try it.

Here's an idea. I have run a procedure four times in the *RUN* room. How about if I turn *that* into a procedure? I go back to the *EDIT* room and define the following procedure (typing it out underneath—not over—the TO SQUARE procedure):

```
TO SQUAREAGAIN
SQUARE
SQUARE
SQUARE
SQUARE
END
```

I zip back into the *RUN* room, and run SQUAREAGAIN. It works! Now, you try something along that line and see what happens. David, why don't you try doing something similar with triangles? Jonathan, try pentangles.

Thinking about it, we've done some interesting things. First of all, by defining procedures in the *EDIT* room, we've actually added to the simple commands that the turtle understands in the *RUN* room. It used to be that the turtle only understood how to RT and LT and FD and BK and et cetera. Now it understands how to SQUARE and SQUAREAGAIN, and (if you've done it) TRIANGLE and TRIAGAIN and PENTANGLE and PENTAGAIN. Another interesting thing is that we've put procedures inside of procedures. TO SQUAREAGAIN, for instance, is a procedure that contains some TO SQUARE procedures.

Now, if I had my tape recorder hooked up to my Color Computer, I would be able to save these new procedures on tape. They'd be part of my turtle's new repertoire forever. Isn't that neat! The turtle becomes more and more clever. It's as if you're a horse trainer, or a dog disciplinarian. I've got to get my tape recorder in there. Those of you with disks will be able to save procedures on disks.

I haven't forgotten about circles, and I'll get around to them in about two shakes.

Looking back at all my procedures, they look pretty simple. TO SQUAREAGAIN was merely TO SQUARE repeated four times. TO SQUARE was merely FD 40 and RT 90 repeated four times. Is there a shorthand for repeating parts of a procedure? You bet there is! Try this:

```
TO SQUAREAGAIN2
REPEAT 4 (SQUARE)
END
```

And this:

```
TO SQUARE2
REPEAT 4 (FD 60 RT 90)
END
```

If something goes wrong in your experimentation, just remember that the turtle carries out everything you tell it to with absolute precision and literalness. Your instructions have to be perfect. Now, if I can make the turtle automatically repeat instructions for a four-sided figure, surely it will do the same for the 360-sided figure that I want to use to approximate a circle. I try this:

```
TO CIRCLE
REPEAT 360 (FD 1 RT 1)
END
```

Unfortunately, that silly little ninny on the screen draws a stop sign instead of a circle. I can take a hint. I'm going to stop right now. See you next month. Maybe by then you or I will have figured out why the turtle drew a stop sign. Write me a letter, and let me know. I know I've promised to tell you about the pig incident, and I will—but I just don't have time right now. I remain,

Uncle Bert

P.S. You can send your cards and letters to me in care of my good friend Dale Peterson. Just address them like this:

Uncle Bert Woofensburger
c/o Dale Peterson
the Rainbow
9529 U.S. Highway 42
P.O. Box 209
Prospect, KY 40059

LET'S CLEAN UP THAT TEXT SCREEN

By Joseph Kolar
Rainbow Contributing Editor

One of the most pleasing things that one can notice in a program is the neatly centered and located text. How disconcerting it is to look at text that is haphazardly tossed on the sereen without any thought to its appearancc. You've seen it often cnough to recognize sloppy text. An otherwise technically well conceived program is belittled with sorry text presentation. To wit: not centered; jammed up in the upper left corner; not ending a line with a hyphen; the left margin being uneven and much, much more.

Good text presentation is very important. Now, fire up your computer and type *PRINT MEM* and ENTER. Copy it down! Copy the three lines, 50 through 54 from the previous *BASIC Corner*.

You should have:

```
50 PRINT"  THE RAINBOW MAGAZINE IS THE"
52 PRINT"BEST MAGAZINE FOR NEWCOMERS TO"
54 PRINT"THE WORLD OF COMPUTING."
```

First, type *PRINT MEM* and ENTER. Copy it, and *RUN* it. Does it look okay? Did you start line 50 with two spaces? Did you start lines 52 and 54 right after the "?

Note that when writing individual lines, you can't end under the " or you will get a blank row between lines when you *RUN* the program.

HINT: You don't need the final " in the *PRINT* statements. Note that this is *not* true for *INPUT* statements.

Most importantly, it is not necessary to print the sentence above on three program lines. It can be done on one line. Key in line 60:

```
60 PRINT:PRINT"  THE RAINBOW MAGAZINE IS
   THE  BEST MAGAZINE FOR NEWCOMERS TO
   THE WORLD OF COLOR COMPUTING.
```

RUN it and see if you have two identical sentences separated by a blank row. If not, you know what to do. When you have a good copy, type PRINT MEM, ENTER and copy it.

You can calculate how many bytes of memory you saved from the data you collected. The key to this new technique is the " after *PRINT*. Draw an imaginary vertical line to the right of " . Let's start over! Type *NEW* and ENTER. Key in:

```
50 PRINT:PRINT" .
```

Skip two spaces to start a sentence and type: THE RAINBOW MAGAZINE IS THE. Stop and note that you have three spaces to go to reach the imaginary line under the NT". Count the letters in the next word. If the word was a two letter word or if the first syllable had one letter plus a hypen, you could add it. Keep in mind that a space is needed after "THE." There are just two effective spaces before the imaginary line. BEST is a one syllable word with four spaces so it won't fit.

Type in spaces under NT" and you will reach the imaginary line. Start typing BEST MAGAZINE FOR NEWCOMERS TO. Hold it! We are coming close to the imaginary line. We allow one more space after "TO." That leaves us one more spaee before we come to the imaginary line. Checking the next word, we find "THE" to be a three space, single syllable word. No good! We go two spaces until we reach the imaginary line and start typing under the B and continue to type. We finish our sentence before we come to the imaginary line. Good work!

By now you have figured out that the imaginary line is our left hand border.

Key in *NEW* and ENTER. We will go, step by step, through the placement of a paragraph using only one program line. Here is the paragragh:

YOU WILL PLEASANTLY DISCOVER, THAT AS YOU PROGRESS, YOU WILL COMMIT MORE AND MORE TO MEMORY. REMEMBER, WITH AVAILABLE REFERENCE MATERIAL, IT IS NOT WRITTEN IN BLOOD, THAT YOU ARE REQUIRED TO MEMORIZE EVERYTHING. ALL YOU HAVE TO KNOW IS WHERE TO LOOK IT UP.

(Joseph Kolar is a free-lance writer and programmer dedicated to proselytizing for computers in general, and the CoCo specifically.)

One cautionary note. A string of more than 255 characters will just hang up the computer on the line you are keying in. As you will presently see, you will have to back off to the end of the last full line, end the line and start a second line.

Type 10CLS:PRINT:PRINT", skip two spaces to start a sentence and start typing. You have one space to go before the imaginary line. Space it in and cross the imaginary line. Continue keying in "THAT AS YOU PROGRESS, YOU WILL." You have two spaces to get under the T. The first syllable of the next word, COMMIT is three letters, which is not enough. Put in two spaces and begin under the T to type some more.

COMMIT MORE AND MORE TO MEMORY. Only one space to cross over the imaginary line. So, space and under the C keep typing REMEMBER, WITH AVAILABLE REFERENCE. Hold it! We are past the imaginary line. Back up to REF, the first syllable. That leaves us four spaces to get to the imaginary line. The next syllable is ER. REFER leaves us two spaces to go. We need a hyphen so we can type REFER, space, cross the imaginary line and start typing again.

Type ENCE MATERIAL, IT IS NOT WRITTEN. Stop! Right to the imaginary line that time. Good! Key in some more text without skipping a space. We want the next word right after the imaginary line because we know that it is the left margin. IN has to be right at the margin.

Continue typing IN BLOOD, THAT YOU ARE REQUIRED. That leaves one space. Space and begin under the I to key in TO MEMORIZE EVERYTHING. ALL YOU. Right on the button!

Since we want HAVE at the left margin, we type H under the T and keep typing. Uh-oh! We typed HAVE TO KNOW IS and can't depress any more keys. We ran out of string space! That's a fine how do you do!

The solution is to back up to the last word on the previous line. Put a closing " after the last word and then put a ; space, because a ; always allows a leading space.

A new program line must be started, line 20. Key it in and keep typing text: 20 PRINT"HAVE TO KNOW IS WHERE TO LOOK IT. Right on the button again. We cross the imaginary line and since we want the word right up to the left margin, we key in UP under HA.

There are always exceptions to every rule. Remember, I mentioned that you do not need the final " in a print statement? Well, you don't in line 20. But line 10 is an exception because, like an input statement, we have to close the statement with a " if we plan to add something else. So type ;. *RUN* it! How about that? A lot of work but worth it. When you get your breath back, let us review.

"You've seen it often enough to recognize sloppy text. An otherwise technically well-conceived program is belittled with sorry text presentation."

When printing text, type one line at a time. It doesn't matter in what column the " after PRINT is. It could be 20 PRINT" or 20 PRINT:PRINT" or 20 CLS:PRINT:PRINT". The imaginary line is right after the " no matter where it falls, and this is the left margin. Always skip two spaces to start a sentence.

NOTE: Some programmers like to indent two spaces when a new sentence would butt up at the left margin, but some don't.

Once you determine where your imaginary line is located, always start your next syllable, word or indented spaces right after the ". Never start punctuation marks, hyphens, parts of syllables (as opposed to an entire syllable) or a single space after the imaginary line.

Determine how much text you can safely fit on the line. Make sure that the line ends with the last letter in a word; a syllable with a hyphen or a punctuation mark to end the sentence. Fill in any blanks with spaces to the imaginary line.

If a punctuation mark ends a sentence right before the imaginary line, make sure you start the next line with two spaces. If a word ends just before the imaginary line, do not skip a space if you want the next word to butt up to the left margin.

If you run out of string space and you are unable to type any more text, back up to the last full line. Put a closing " just after the imaginary line, then put a ;. Start a new program line and type right after " and continue.

If you typed this in two program lines, you required 275 bytes. If you typed it in the nine individual lines, you needed 319 bytes. That adds up to 44 saved memory bytes.

Nice going, indeed!

Sea Dragon:
Fast Paced Submarine Action

There is much to like about *Sea Dragon*, a new arcade game from Adventure International. Why, then, am I reluctant to tell you to run right out and buy it?

Let me start by extolling the game's virtues—and there are several.

The self starting, machine language game begins with a scoreboard called "The Famous Sailors Hall of Fame." (A bit redundant, but so what?) The top 10 scores will be recorded here.

A very realistic voice comes through the TV speaker saying, "Welcome aboard, Captain!" (Wow! How did they do *that*?)

Your ears are then treated to a beautiful rendition of "Across the Wide Missouri" and a couple of other sailing tunes. Theoretically, you may at any time begin to play the game by pressing 1 or 2 for the number of players or the letter "P" for the Practice Mode. (Did he say "theoretically?" Ah—we've just touched upon one of *Sea Dragon's* faults. I'll explain shortly.)

Then you must choose your level of play—from zero (slowest) to seven (fastest). The instructions claim that zero is "easiest," but, personally, I find a slightly higher level such as 2 to be a bit easier. The screen scrolls by faster, but I can also shoot and jump faster. I agree, however, that level 7 is the "most difficult." In a two player game, each player chooses his own level.

Ready to go? Wait just a second. You've got two more choices to make: Music off, yes or no? Joystick or keyboard?

Now what's this about "music off?"

Each player gets three submarines (game lives). If you said "no" to the "music off" prompt, then when your first sub appears on the screen, it will be accompanied by a rousing chorus of "I'm Popeye the Sailor Man" (toot, toot!). The loss of each game life is signaled by a few sad notes played in a minor key.

I found that cute the first couple of times, but after that I was glad to have the option of doing without.

At last—here we go! Your mission is to sail through an aquatic minefield and navigate treacherous underwater caverns with a limited air supply, shooting at or jumping over a variety of obstacles, on your way to the ultimate target: the Master Mine.

Choose your weapon. If you want to play with the keyboard, press the spacebar to start the screen scrolling from right to left. Move up, down, left (drop back) or right (forward) with the four arrow keys. The spacebar will fire torpedos straight ahead. The *ENTER* key will send a missile upward to a target above you.

"While it might take a bit more practice," says the instruction sheet, "the keyboard will ultimately provide the best maneuverability." Maybe it will if your fingers are better coordinated than mine. I did much better with the joystick.

The fire button on the joystick starts the action and shoots the torpedos—but you still need the *ENTER* key for the missiles. Maybe with some of the newfangled sticks that would pose no problem, but with the standard Radio Shack stick, since it takes one hand to hold the base of the controller and press the firebutton, and the other hand to work the stick, that doesn't leave any hand free for the *ENTER* key. I tried holding the key down with an elbow or wedging it in with a small piece of paper. That had the interesting effect of launching the missiles at random—and some of them even hit what they were supposed to.

You get from 50 to 500 points for each mine, stalactite, battleship or laser shooter you can shoot, but you'll soon find that there are some obstacles you can't shoot. You can only jump, duck, or run. No points for that, but you stay alive.

You can lose a game life by colliding with any obstacle, with the ocean floor, or with the walls and ceilings of the caverns. You can also get killed by deadly rays from the laser shooters, or by fallout from a mine that you've shot.

Although the instructions don't mention it, the game is played in plateaus. If you lose your sub shortly after starting out, your next game life starts again at the beginning. If you can survive as far as the battleship that guards the entrance to the first cavern, you've reached the first plateau. Now if you get killed, you'll start again at that point. Make it through that cavern, and when you come up for air, you're at the second plateau.

How many plateaus and caverns and obstacles there are before you get to the Master Mine, what happens after you shoot it (for 5,000 points) or what the penalty is for running out of air, I can't tell you, because I just never could survive long enough to find out.

Earlier, I mentioned that there's a "Practice Mode." You get an unlimited supply of replacement subs, and the points don't get posted to the scoreboard. Without the Practice Mode, I'd never have seen the second plateau.

Sea Dragon is not just one game. Actually, it's two! What I have been describing till now is only the first game on the tape, the "novice version." There's also a second game called

the "expert version." It is played exactly like the other, but with more and trickier and deadlier obstacles. Even in the Practice Mode, I've never made the first plateau. Better arcade players than I am will be challenged.

As I said, there's a lot to like about *Sea Dragon*. So what could be wrong?

Unfortunately, *Sea Dragon* suffers from a couple of major faults that a $35 program just shouldn't have.

One of them, as I alluded to earlier, is the problem of making the initial selection—one player, two players, or practice. Under the heading of "Beginning the Game," the instruction sheet states: "When the . . . Hall of Fame is displayed on the screen, you may choose the number of players(press 1 or 2), or enterthe Practice Mode by pressing P."

Sorry, but it doesn't work that way. When the music that accompanies the scoreboard gets underway, the keyboard response becomes most unfriendly, and unless you happen tohit the key at exactly the right second, you will just have to sit through an entire chorus of "Across the Wide Missouri." Beware of holding the key down in frustration, or you may find that, without meaning to, you have selected one player at level one, or two players with the first player to play at level two.

If you have selected the "music off" option, then when the scoreboard shows up again at the end of the game, it will sometimes do so silently, and you may select your next game easily at any time. But other times, like it or not, the concert and the frustration will start all over again.

Remember that if you didn't turn the music off, each game life ends with a few notes of music as the action pauses. If you are playing without music, then when each sub is sunk, the screen will jump immediately to the next event—either the next sub, or the scoreboard.

In the early going, that won't make much difference. But what happens after 10 games, or five two player games? Then, of course, it's possible you'll finish with a score that's not one of the 10 best. In that case, since the screen changes instantly when the game ends, you may never see what your score was. That can be very annoying.

I am also not enamored of the packaging. Besides being inconvenient, it may be a contributing factor to the high price. *Sea Dragon* is supplied on one cassette in a bulky, awkward, vinyl book size storage case designed to hold two cassettes. I much prefer the simple packaging used by Spectral, Mark Data and many other software publishers.

Here, then, are my recommendations to the authors and the publisher:

—Reprogram the scoreboard audio so that, when the game is first loaded, the concert will play once only, and that all subsequent appearances of the scoreboard will be silent. Amend the instructions accordingly.

—Reprogram the game action so that when you go down for the third time, the final score will stay up there for a few seconds, with or without the music.

—Sell the game in a simple, convenient, inexpensive package, and cut the price to reflect the cost savings.

If these changes are made, I will then be able to recommend wholeheartedly to my fellow CoCo enthusiasts: Set sail for your nearest software dealer and reel in a prize catch.

(Adventure International, Box 3435, Longwood, FL 32750, 32K ML cassette, $34.95)

—Neil Edward Parks

Book Review

Illustrated Computer Dictionary Makes Good Guide

The *Illustrated Computer Dictionary* from the editors of *Consumer Guide* is like one of those poeket sized language guides that helps you say what you mean so you can avoid telling a waiter in Italy that you'll have the boiled alarm clock with a side order of locomotive wheels.

That is to say, it is a simple guide to common computer terminology that will take a normal human being through the labyrinthine channels of a programming article written by a computer specialist. It probably won't cause the meaning of all the jargon to come clear, but it will give you some idea of what's going on.

The very nature of the computer—something very technical and new to human experience—guaranteed that a jargon would grow up around it. To the uninitiated, that jargon appears to be more like a thieves' cant: a language purposely made confusing to all but the select.

The *Consumer Guide* dictionary (a 180-page, spiral bound paperback) opens up the jargon by patiently and simply plodding through it so that a new user can get a grasp on the terminology. Old hands can learn a thing or two from the dictionary, as well. All too often, we toss around acronyms and jargon without really knowing what we're talking about. It's easy to get deeper into a conversation with Mr. Computer Wizard than we bargained for, then stupidly nod just like we know what's going on. With this guide, at least we may have run across the words before.

As are all *Consumer Guide* books, the dictionary is absolutely non-preferential to any manufacturer. There is very little manufacturer-specific language defined in the dictionary. Most of the terms are general and applicable to almost any system.

The illustrations in the dictionary are simple, but elegant stipple-shaded line drawings, used to clarify written definitions. The terms are listed alphabetically, but in their most common forms (in most cases). Acronyms are alphabetized as they stand, not in order of the words abbreviated. For example, "EDP" comes after "editor" instead of after "electronic bulletin board" as it would if listed as "Electronic Data Processing."

One of the neatest things about the book is the definition and pronouncing guide to acronyms. It defines acronym as "a word formed from the initial letter or letters of the successive parts (or major parts) of a compound term."

For instanee, most of us know that BASIC stands for Beginners All-Purpose Symbolic Instruction Code. All right, class, how many of you knew that "bit" is actually an acronym for *B*inary Dig*IT*? Shucks, I didn't even know that a nibble is half a byte.

My only criticism is that many of the definitions are skeletal, leaving the reader wondering just exactly what is meant. On the other hand, complex information processing isn't this book's purpose. It's a traveler's guide and it's a good one. I'll surely have a copy on hand the next time I try to translate the native speech.

(*Illustrated Computer Dictionary*, Editors of Consumer Guide, Exeter Books, NY, 1983, spiral bound paperback, 180 pages, call a bookstore for price)

—Bruce L. Sublett

Indexer Will Keep Track Of Your Variables

Where is the variable ZQ$?

Indexer is a utility which can answer that question and more! "Which lines in my program make reference to line 275 (or any other line)? Did I ever use the word 'monster' in my program? Where was the keyword '*GOSUB*' used?"

Indexer is a powerful machine language utility which cross references a BASIC program. It cross references variables and line numbers and tells in which line number they are used (as in the case of *GOTO* or *GOSUB*). Additionally, it will search your program for any word that you want!

Indexer is very useful for final documentation of a program or for debugging. A list of all variables used and line numbers referenced within the program is essential when working on a faulty program. *Indexer* will provide such a list in a matter of minutes.

The program is a little over 2K bytes long and loads into the top of user RAM. It is written in position independent code (for 16K or 32K machines with or without disks) and can be relocated anywhere.

Load *Indexer* by typing *CLEAR* 200,&H377F (&H777F for 32K). This is absolutely mandatory. If you forget this step, your computer goes off to never-never land! Next type *CLOADM "INDEXER" (for 32K, CLOADM "INDEXER", &H4000)* then type *EXEC*. *Indexer* is now in the "back of the mind" of the computer. At this time you may start entering a new program or load in a previously saved one.

Indexer effectively adds three new keywords to the BASIC vocabulary, @I, @S and @K. Typing @I causes an indexed list to be created and outputs the result to screen or printer. @S will search the BASIC program for any string up to 32 characters long or search for any BASIC keyword. @K will kill the workings of *Indexer* and reset all "hooks" back to normal operation.

Like all good utilities, you don't even know that *Indexer* is there until you need it. When you do call on *Indexer*, by typing one of the above three commands, the screen clears and a header appears to let you know you are in *Indexer*.

If the command typed was @I, you are then prompted for certain information. To the prompt "program name?" you may respond with up to 12 characters. Then you will be prompted for the date. Here you may type in up to 18 characters. (I found that using additional text information here helps document the program better.) A final prompt of "PRINTER (Y/N/R)?" is asked. Y=YES, N=NO and R=RETURN to BASIC. If the response is "Y" the printer is checked to see if it is online. If not, "PRINTER BUSY" is displayed. The program now goes into action outputting the cross reference to the printer in a very organized and sequential format. What you now have is a listing of every variable used within your program and in what lines that variable is used. You also have a list of every line number that is called on and the calling line numbers.

If you answered "N" to the printer prompt, then all the above information is listed to the screen. The information flies by somewhat quickly (as when giving the *LIST* command), but this can be stopped simply by hitting the spacebar at any time. To resume the output, hit the spacebar again. To abort the output, press "A."

The command @S followed by a string up to 32 characters long, will cause *Indexer* to search for that string. If the string is found, operation goes to the BASIC edit mode. At this time you can edit the string as you like (the way you would in BASIC) or hit *ENTER* to keep the string as is and continue the search for another occurrence of the string. You may type "Q" to quit the search when you are in the edit mode. After the last string has been found, you are told that "SEARCH FAILS" and control is returned back to BASIC.

Indexer resets a few RAM "hooks" for its operation. If you should want to return the system to normal operation, then type @K. This will set the hooks back to normal, and defeat the operation of *Indexer*. To restart, type *EXEC* &H37BF (&H77BF for 32K).

Operational information supplied with the program was good. In addition, memory locations are given for changing the printer output format and baud rate by *POKEing* the proper codes.

If you write long programs and find it to be a nuisance to keep a running list of variables used, then *Indexer* was written for your convenience! I recommend it.

(MI-US'R Software, 115 Rising Sun, Fort Mitchell, KY 41017, $14.95 tape)

—Al Burzynski

Diamond In The Rough Just Needs Polishing

It's a given fact that you can save a lot of coins by finishing furniture yourself. Plus, you can learn a lot in the process. I guess the same philosophy might just bleed over into the software field somewhat because the product I'm talking about is delivered to you cheap and all you have to do is customize it to fit your particular financial planning needs.

The *Household Budget Worksheet*, or *HBW* as it's called by its author, comes as a short, shell of a program with sufficient documentation to allow you to modify the coding to write in your own variable assignments for *your* expenses (both fixed and variable) and your various incomes. *HBW* will then give you a printout (on any printer) of the worksheet. No more flipping through a dozen or more screens trying to correlate this and that. Now you can go sit down in front of the "boob tube" and figure out from your worksheet where you're going to get the money to buy that modem or whatever.

Onee initially set up by you, following the short steps in the manual, and then re*CSAVEd*, *HBW* only requires you to *CLOAD* it once a month or at your convenience, plug in your actual (or estimated) expenses and income from preassembled data, and without even asking if you're ready gives you a neat printout. That's it. Again: from that first program modification session on, all you have to do is enter the current month, your payment and expense figures, including any unbudgeted expenses, as well as income. Sit back and within a few seconds you've got a tangible printout to be mulled over and filed away.

The more I ran this little program, the more I liked it. It does not have a bunch of bells and whistles nor is there anything elaborate about its documentation, but it does give you something you can hold in your hands and analyze. No file saving; no figuring out a long set of sample personal finance situations just to run the darn thing. *HBW* is short and to the point. Its bottom line is printed right out in the open for you to see, giving you both itemized and total expenses, itemized and total income, balances due, savings and just how much you have left over to play with. Even tells you how many more dreaded months you'll owe your soul to the bank, Sears, the encyclopedia salesman, company store or whomever.

The program coding is right there in the manual so you can get an idea of what you have to do. Modifying the program is no big deal at all, even for a "newbie" to programming. *HBW* uses some very unique but simple algorithms to calculate everything for you. It gives you enough room to work with 30 fixed expenses, 20 variable (budgeted and/or charge account) expenses and 20 income itemizations. You ean even enter "exception expenses" that you didn't budget for, like a new pair of sneakers for the wife after that new piece of "gotta have" software showed up in the mail.

I didn't like *HBW* when I first got it. I'm one of those people who'll have the guy at the store assemble the 10-speed before I pick it up. I hate to admit it, but those instructions always seem like so much Japanese to me. So when I read in the *HBW's* eight page manual that the data loaded into the program is demonstration data and that "all" I had to do is delete a block of lines and put in my own, I said whoa! I mean, it wasn't too long after I bought my computer that I discovered how truly lazy I am. Why should I pay someone for a chance to program—and not make a cent for myself? I was ready to chuck the whole shebang out the window. The nerve!

But, I decided to give it a fair chance. And I ended up giving it more and more chances. It soon grew on me, because what I had overlooked from the beginning was its sheer simplicity.

Not too shabby for a wee program that takes up a bit just over 4.2K before running (a minimum of 16K and Extended Color BASIC are required). If you've got disk, you simply transfer it directly to speed up loading with no problem.

If the package had cost much more, maybe I wouldn't recommend it. But at this price, it just might be a gem if you think you can follow instructions and plug this doohickey into that whatchamacallit, and if you still think "budget management" is within your capabilities, just think: you get to do some programming to boot!

So tell, me, what's wrong with unfinished furniture, anyway? The price is right!

(CoCo Data Enterprises, 1215 Emerald Drive, Orlando, FL 32808, cassette only, $6.95 + $.75 S&H)

—Ed Lowe

Number Match Is Ideal For Young Learners

Number Match is a learning program distributed by Micro School Programs. The program requires the user to match a domino pattern with a picture pattern and a numeral. It is intended to help the user develop the concept of numbers, learn to count from one to nine, and recognize the numerals for these numbers. It is geared for the age level of four to six years.

The program is written in BASIC for a 32K Extended BASIC Color Computer, and the first thing that I noticed about it was that it disabled the BREAK key. This is excellent in a children's program because I know from experience that little children love to push that red button.

The program first displays the title and author of the program, and then asks you for your first name. After this, the program asks you to choose a level. (*Easy*, *Medium*, and *Hard*.) At this point, you can choose to have an example shown to you or to continue on to the lesson. The documentation encourages you to do the example if it is your first time using the program.

You can also choose to have the answer choices change automatically at a set speed or manually at your own speed. I found that the three speeds (*Slow*, *Medium*, and *Fast*) were too fast, and I felt that they should be slowed down. The fast speed was too fast even for me, but then again maybe a six-year-old is more dexterous than I am.

After all of the initial entries, I then called my four-year-old daughter to the computer for a test run. My daughter is familiar with using the computer since we have other educational preschool programs for her. At the start of the lesson, the screen is divided into four sections. The upper half of the screen is divided into three rectangles. The first rectangle contains the level you chose and the score. The second rectangle contains the domino pattern, and the third rectangle contains the numeral. The lower half of the screen contains the picture pattern which can be houses, dogs (I think they were dogs), trucks, fishes, etc.

We started out with the manual speed, and she progressed through the three levels rather rapidly. The differences in the three levels are:

Easy:
Given: Domino pattern
1st Match: Picture pattern
2nd Match: Numeral

Medium:
Given: Picture pattern
1st Match: Domino pattern
2nd Match: Numeral

Hard:
Given: Numeral
1st Match: Domino pattern
2nd Match: Picture pattern

I discussed these levels of difficulty with my wife, who used to be a preschool teacher, and she confirmed that they were correct.

As you progress through a lesson in the manual mode, the domino pattern first appears on the screen. Next a picture pattern appears in the lower half of the screen. If the two patterns match in number, then you press ENTER. If the two patterns do not match, then you press the spacebar to change the picture pattern. If the match made is correct, a happy face will appear at the bottom of the screen. If it is incorrect, a sad face will appear. You will then try again until a match is made.

In the automatic mode, the picture pattern changes automatically at the speed you choose. You then press the spacebar when there is a match.

As I stated earlier, I feel that the progression from manual mode, where you have all the time you need, to the *Slow* mode is too severe and it should be slowed down somewhat. One of the good things about this program is that it is written in BASIC. Because of this, I was able to look at the program and modify it so that the *Slow* mode was at the speed I wanted.

After the session is over, the program reports the results. A record of the number of matches made on the first try will be displayed. The program also prints "I'll see you later" and prints your child's name. My daughter really got a kick out of seeing her name printed by the computer.

The documentation for the program is also written quite well. It takes you step by step through the whole program and also gives you tips on using the program. The only complaint that I have about the documentation is that it is put together backwards. They start out by showing you how to use the program, and then in the back they describe how to load the program. Also, their loading instructions do not work. The directions are:

1. Turn the TRS-80 on.
2. Put in the diskette.
3. Press the RESET button.
4. The program will load automatically.

I followed the directions, and it didn't work. I think that these are instructions for loading onto a Model I or III.

In summary, I think that *Number Match* is a worthwhile program to buy. The graphics are very well done, and the program is very well written. If your child goes to preschool, this would be a very good supplement to his or her education. If your child does not go to preschool, this program would be a definite asset in assisting his or her education and preparing for grammar school.

(Micro School Programs from Bertamax Inc., 3647 Stone Way North, Seattle, WA 98103, tape $24.85, disk $29.50)

—Paul Lee

Tidying Up Your *Word Processor* Program

By D. S. Lewandowski
Rainbow Contributing Editor

I would like to thank everyone who has requested "Using An Editor/Assembler." They are being sent out as fast as possible. If you haven't sent for one all you have to do is send a 37 cents stamp, or two 20 cents stamps, to DSL Computer Products, Inc., P.O. Box 1113, Dearborn, Mich., 48121. I have chosen to leave "edit" out of this month's column, there are some rather interesting things we are going to do with Load and Save.

Okay, if you have entered the June text, follow these steps before entering any more. First delete lines 730 and 740, or if you wish just change them to remark statements as I have done. Also rename line 750 to *EDIT*. Next delete lines 1000 and 1010. Start entering text at 1000.

At line 1000 we will place our text, or prompts. Since we are dealing with cassette I/O, we need a filename, a ready cassette prompt, and of course, an error message in case there is no text in buffer. Moving to line 1060 we find the entry point for the load routine. We point to the ready cassette prompt, and wait for a keypress. Once the routine is entered it will load the first file it encounters on the tape. Now here is an interesting little fact; this is the same routine used to load BASIC and machine language programs. At LOOP4 we keep reading in blocks of data until the pointer for the *ROM* routine contains something other than $FF. Once an incomplete block is sensed, we assume we have reached the EOF, or End Of File. At this point we branch to LOUT, or Load OUT. Since the X register is already pointing at the last byte it placed in memory we store it at BUFEN, and jump back to the menu.

Next at line 1320, we find *SAVE*. The first item we need is a filename so we point the X register at MES4. Then we jump to our *PRINT* subroutine way back at line 190. Next we perform a little error checking; if the B register is greater than nine the filename is too long, so we simply ask for it again. So far, so good. We prompt for a cassette ready, at WAIT3. Saving a program involves a bit more than loading one. Pointing X at the input buffer, line 1450, we then WAIT for a keypress. Now we start the cassette and write a leader. At line 1500 we point X directly at the text buffer (we already have the number of bytes to save, back at line 1320). Saving one block at a time we check the VAR after each block to see if it is time to close. Again, if we have anything other than a hex $FF it is time to close the file. Once finished, or should I say almost finished, we branch to SOUT, or Save OUT. Here we place the End Of File marker and save the partial block, turn off the cassette, and rejoin the main menu.

As I said earlier, these routines are the same routines which save and load BASIC and machine language programs. Try it, notice how your BASIC program looks nothing like a listing. Why? To conserve memory all BASIC keywords are tokenized. These tokens show up as graphic symbols on the screen. Well, that's it for now, have fun, and we'll see you next month.

The listing:

```
0E00                    00100          ORG     $E00
                        00110 * A SIMPLE TEXT PROCESSOR IN ASSEMBLY LANGUAGE
0E00 108E 10D6          00120 START    LDY     #BUFF   GET BUFF LOC.
0E04 109F 0E37          00130          STY     BUFST
0E08 109F 0E39          00140          STY     BUFEN
0E0C BD A928            00150          JSR     $A928   CLS
0E0F 8E 0ED1            00160          LDX     #MES1   PRINT INTRO
0E12 BD 02              00170          BSR     PRINT   DISPLAY ON SCREEN
0E14 20 0A              00180          BRA     CONT    GO AROUND ROUTINE
0E16 A6 80              00190 PRINT    LDA     ,X+     GET BYTE
0E18 27 05              00200          BEQ     DONE    IF ZERO EXIT
0E1A BD A30A            00210          JSR     $A30A   OUTPUT A TO SCREEN
0E1D 20 F7              00220          BRA     PRINT   LOOP TILL DONE
0E1F 39                 00230 DONE     RTS     RETURN FROM SUB
0E20 BD A393            00240 CONT     JSR     $A393   ROM INPUT ROUTINE
0E23 10BE 0E39          00250          LDY     BUFEN   TEXT POINTER
0E27 A6 80              00260          LDA     ,X+
0E29 A6 80              00270 LOOP     LDA     ,X+     POINT AT INPUT BUFFER
0E2B 81 5E              00280          CMPA    #$5E    UP ARROW?
0E2D 27 16              00290          BEQ     FIN
0E2F 81 0D              00300          CMPA    #0      END OF LINE?
0E31 27 00              00310          BEQ     MORE    GET ANOTHER LINE
0E33 A7 A0              00320          STA     ,Y+
0E35 20 F2              00330          BRA     LOOP
0E37 0000               00340 BUFST    FDB     0       BUFF POINTER
0E39 0000               00350 BUFEN    FDB     0
0E3B 86 0D              00360 MORE     LDA     #$0D    ENTER BYTE
0E3D A7 A0              00370          STA     ,Y+
0E3F 10BF 0E39          00380          STY     BUFEN   SAVE LOC.
0E43 20 DB              00390          BRA     CONT    GET TEXT
0E45 86 00              00400 FIN      LDA     #$00    END OF TEXT
0E47 A7 A4              00410          STA     ,Y
0E49 109F 0E39          00420          STY     BUFEN   UPDATE POINTER
0E4D BD A928            00430 FIN1     JSR     $A928   CLS
0E50 8E 0F77            00440          LDX     #MES3
0E53 BD 0E16            00450          JSR     PRINT   DISPLAY OPTIONS
0E56 BD A1C1            00460 WAIT     JSR     $A1C1   INKEY$
```

(Dennis Lewandowski, one of the early authors active with the Color Computer, specializes in machine language programming. He and his wife, Rose, founded DSL Computer Products.)

```
0E39 27 FB      00470        BEQ    WAIT
0E3B 81 43      00480        CMPA   #43    ASCII C
0E3D 27 1C      00490        BEQ    REST    RESTART
0E3F 81 45      00500        CMPA   #45    ASCII E
0E41 1027 0038  00510        LBEQ   EDIT
0E45 81 4C      00520        CMPA   #4C    ASCII L
0E47 1027 01AE  00530        LBEQ   LOAD
0E4B 81 50      00540        CMPA   #50    ASCII P
0E4D 27 1E      00550        BED    PAPER
0E4F 81 53      00560        CMPA   #53    ASCII S
0E51 1027 01E5  00570        LBEQ   SAVE
0E55 81 58      00580        CMPA   #58    ASCII X
0E57 27 21      00590        BEQ    EXIT
0E59 20 DB      00600        BRA    WAIT
0E5B BD A928    00610 REST   JSR    $A928   CLS
0E5E 8E 0F20    00620        LDX    #MES2   POINT AT PROMPT
0E61 BD 0E16    00630        JSR    PRINT
0E64 8E 0E37    00640        LDX    BUFST   POINT AT TEXT
0E67 BD 0E16    00650        JSR    PRINT   PRINT TEXT
0E6A 7E 0E20    00660        JMP    CONT    ENTER MORE TEXT
0E6D 10BE 0E37  00670 PAPER  LDY    BUFST   POINT AT START
0E71 A6 A0      00680 LOOP2  LDA    ,Y+     GET TEXT
0E73 27 0B      00690        BEQ    FIN1    ALL TEXT PRINTED
0E75 BD A2BF    00700        JSR    $A2BF   SEND TO PRINTER
0E78 20 F7      00710        BRA    LOOP2
0E7A 7E A927    00720 EXIT   JMP    $A927
                00730 * DELETE  730 TO 740
                00740 *MOVE EDIT TO 750
0E7D 8E 0EA5    00750 EDIT   LDX    #TMES
0E80 BD 0E16    00760        JSR    PRINT
0EA3 20 81      00770        BRA    WAIT
0EA5      4E    00780 TMES   FCC    /NOT AVAILBLE TILL SOFT ISSUE    OF RAINBOW/
0ECF      0D00  00790        FDB    $0D00
0ED1      20    00800 MES1   FCC    /          A SIMPLE TEXT/
0EE8      0D    00810        FCB    $0D
0EE9      20    00820        FCC    /        HANDLING PROGRAM/
0F02      0D    00830        FCB    $0D
0F03      20    00840        FCC    /      by  R.S. LEWANDOWSKI/
0F1E      0D0D  00850        FDB    $0D0D
0F20      20    00860 MES2   FCC    / ENTER TEXT TERMINATE EACH LINE  WITH AN
                               enter, PRESS ^ KEY AND  PRESS ENTER TO STOP./
0F75      0D00  00870        FDB    $0D00
0F77      20    00880 MES3   FCC    / C - CONTINUE/
0FB4      0D0D  00890        FDB    $0D0D
0FB6      20    00900        FCC    / E - EDIT/
0F8F      0D0D  00910        FDB    $0D0D
0F91      20    00920        FCC    / L - LOAD FROM TAPE/
0FA4      0D0D  00930        FDB    $0D0D
0FA6      20    00940        FCC    / P - SEND TEXT TO PRINTER/
0FBF      0D0D  00950        FDB    $0D0D
0FC1      20    00960        FCC    / S - SAVE ON TAPE/
0FD2      0D0D  00970        FDB    $0D0D
0FD4      20    00980        FCC    / X - EXIT TO BASIC/
0FE6      0D00  00990        FDB    $0D00
0FE8      20    01000 MES4   FCC    / FILENAME /
0FF2      0D00  01010        FDB    $0D00
0FF4      20    01020 MES5   FCC    / READY CASSETTE /
1004      0D00  01030        FDB    $0D00
1006      4E    01040 ERMES  FCC    /NO TEXT IN BUFFER /
1018      00    01050        FCB    $
1019 8E 0FF4    01060 LOAD   LDX    #MES5   POINT AT MES5
101C BD 0E16    01070        JSR    PRINT   DISPLAY IT
101F BD A1C1    01080 WAIT2  JSR    $A1C1   INKEY$
1022 27 FB      01090        BEQ    WAIT2
1024 0F 7C      01100        CLR    $7C     BLOCK TYPE
1026 86 0F      01110        LDA    #$F     15 BYTES
1028 97 7D      01120        STA    $7D     # OF BYTES
102A BD A77C    01130        JSR    $A77C   SYNC W/LEADER
102D 8E 0E37    01140        LDX    BUFST   POINT AT BUFFER
1030 9F 7E      01150        STX    $7E     UPDATE ROM POINTER
1032 BD A70B    01160        JSR    $A70B   READ BLOCK
1035 6F 84      01170        CLR    ,X      PUT ZERO AFTER NAME
1037 8E 0E37    01180        LDX    BUFST
103A BD 0E16    01190        JSR    PRINT   PRINT NAME
103D 8E 0E37    01200        LDX    BUFST   READY FOR TEXT
1040 9F 7E      01210        STX    $7E
1042 0F 7D      01220        CLR    $7D
1044 0C 7C      01230        INC    $7C
1046 BD A70B    01240 LOOP4  JSR    $A70B   READ BLOCK
1049 9F 7E      01250        STX    $7E     UPDATE ROM POINTER
104B 0C 7D      01260        INC    $7D     FULL BLOCK?
104D 26 02      01270        BNE    LOUT    NO = DONE
104F 20 F5      01280        BRA    LOOP4   GET NEXT BLOCK
1051 BF 0E39    01290 LOUT   STX    BUFEN   X=END OF TEXT
1054 BD A7EB    01300        JSR    $A7EB   MOTOR OFF
1057 7E 0E40    01310        JMP    FIN1    DONE
105A FC 0E39    01320 SAVE   LDD    BUFEN   GET LENGTH
105D B3 0E37    01330        SUBD   BUFST   MINUS START
1060 27 66      01340        BED    NOTXT   OOPS
1062 FD 10C6    01350        STD    VAR     SAVE BYTE COUNT
1065 8E 0FE8    01360 REDO   LDX    #MES4
1068 BD 0E16    01370        JSR    PRINT
106B BD A393    01380        JSR    $A393   GET NAME
106E C1 09      01390        CMPB   #9      CHECK FOR 8
1070 2E F3      01400        BGT    REDO     TOO MANY
1072 8E 0FE8    01410        LDX    MES4
1075 BD 0E16    01420        JSR    PRINT
1078 BD A1C1    01430 WAIT3  JSR    $A1C1
107B 27 FB      01440        BEQ    WAIT3
107D 8E 02D0    01450        LDX    #$2D0   POINT AT NAME
1080 9F 7E      01460        STX    $7E     ROM POINTER
1082 8E 000F    01470        LDX    #$0F    # OF BYTES
1085 9F 7C      01480        STX    $7C     ROM POINTER
1087 BD 34      01490        BSR    WRITE   LEADER BLOCK
1089 8E 0E37    01500        LDX    BUFST   POINT AT BUFF
108C 9F 7E      01510        STX    $7E     ROM PINTER
108E BD A7D8    01520        JSR    $A7D8   MOTOR+SYNC
1091 0C 7C      01530        INC    $7C     PUT 1 FOR DATA
1093 FC 10C6    01540 COUNT  LDD    VAR     # OF BYTES
1096 1083 00FF  01550        CMPD   #$00FF  EOF?
109A 2B 15      01560        BMI    SOUT    IF SO OUT
109C 83 00FF    01570        SUBD   #$00FF  -255
109F FD 10C6    01580        STD    VAR     BYTES LEFT
10A2 2A 05      01590        BPL    NEXT    BUMP BLOCK
10A4 C3 00FF    01600        ADDD   #$00FF
10A7 20 02      01610        BRA    NXT
10A9 C6 FF      01620 NEXT   LDB    #$FF    255 BYTES
10AB D7 7D      01630 NXT    STB    $7D     ROM POINTER
10AD BD 11      01640        BSR    WRT     TO TAPE
10AF 20 E2      01650        BRA    COUNT   LOOP
10D1 86 FF      01660 SOUT   LDA    #$FF    END OF FILE
10B3 97 7C      01670        STA    $7C     MARKER
10B5 BD 09      01680        BSR    WRT     TO TAPE
10B7 BD A7EB    01690        JSR    $A7EB   MOTOR OFF
10BA 7E 0E4D    01700        JMP    FIN1    DONE
10BD BD A7D8    01710 WRITE  JSR    $A7D8   MOTOR+SYNC
10C0 BD A7F4    01720 WRT    JSR    $A7F4   SAVE BLOCK
10C3 9F 7E      01730        STX    $7E     UPDATE ROM
10C5 39         01740        RTS            BLOCK SAVED
10C6      0000  01750 VAR    FDB    0
10C8 8E 1006    01760 NOTXT  LDX    #ERMES
10CB BD 0E16    01770        JSR    PRINT
10CE BD A1C1    01780 WAIT4  JSR    $A1C1
10D1 27 FB      01790        BEQ    WAIT4
10D3 7E 0E4D    01800        JMP    FIN1
1006            01810 BUFF   *
                0E00 01820   END    START
00000 TOTAL ERRORS
```

Math Game Is A 'Plus'

Everyone knows that two plus two equals four but did you realize that it really depends upon what you want to teach with that equation. In the cse of Bertamax's *Math Games That Teach* (Facts Match—Addition), designed for "personalized instruction on personal computers for both home and school," the author has provided a unique way of reenforcing mathematical knowledge of numbers, words and, in this case, additon facts.

The program comes with a well-documented booklet that explains, quite clearly, all particulars of the program. The instructions suggest that an adult explain the program to the child the first time around and then let the child work on his or her own. The age level recommended for this *Math Game* is six through eight and works with addition of the numbers from zero to 18. "No big deal," you say? Wrong. Here is where the author's experience in education shows her ability in understanding what young children have to master with mathematics.

The program screen is divided up into four segments. A large rectangle (bottom) shows the math problems that the child is to work on (graphic numbers—easy to read). The top three rectangles are, left to right, program choice with score, word name for the numeral shown, and on the right side, the numeral that will provide the answer.

In learning math it is necessary not only to learn what one number plus another equals, but also what the words for the numbers are. It is also helpful to be able to look at a number (i.e., 7) and realize that two numbers can be added to achieve that number (5 + 2, etc.). The program permits the user to choose from three levels that will allow you to decide which system you want, pick the answer to the problem and then the word, pick the answer and figure out the problem, or pick the word and get the problem. Right answers get a "happy face" with sound, wrong answers get a "sad face," also with sound. You also have three options for groups of numbers, 0 to 9, 10 to 18, and 1 to 18. The program will also auto-run an example so you will get the hang of how it works.

The author (Kay L. Schrag, M.Ed.) personalized the program by allowing the user to select not only how fast the program will operate, but choice of "manual" so you can move it at your own speed.

A feature that I personally approve of is that the youngster does not have to search out the keyboard to hit the proper number in order to respond to a problem. The numbers are automatically flashed on the screen, and when the proper number appears the child only has to hit the space bar. If the wrong answer is chosen, the program continues until the child gets it right. After all problems are completed (and correctly done) a report screen shows all the equations done and marks an "X" where the proper response was given on the first try.

I tried out the program on my three official program testers, Jennifer, 8, David, 10, and Mike, 12. All three liked the way the program was designed and enjoyed playing it, although Dave and Mike felt it was too easy for them. They liked the faster speeds. Actually they were right, the age level is six to eight and Jennifer fit right in. She used the slow speed and did fine, thinking out the problems and watching for the happy face.

I feel that the program will help children in the primary grades build up confidence with these introductory math and word problems. I would suggest to parents and teachers that a reward system could be developed for students improving their first-try results.

There are a few minor problems that I encountered. First, the program did not load according to the directions for the TRS-80 diskette. I had to list the directory and run the loader program. Second, I do not like the package the program comes in. Not that it is not colorful, clear and concise, but that there is no way it is going to last, especially in a school setting. I would prefer all school programs to use a vinyl "library" holder for protection. The booklet is fine and should be no problem at all. Thirdly, I had to adjust to TV set color in order to get the program's letters clear enough to read.

These problems can be easily resolved and the most important, I feel, is that the program did not load according to the directions.

Micro School Programs, by Bertamax, Inc., has a series of programs for mathematics, reading, telling time, teachers, and administrators. If they are all as well-thoughtout and designed as "Facts Match-Addition" then Bertamax, Inc., (including Kay L. Schrag) has a winner.

(Bertamax, Inc., 101 Nickerson Street, Suite 202, Seattle, WA 98109, tape $39.50, disk $39.80)

—Michael F. Garozzo

Spelling In Context
Impressive

Spelling In Context, Level2 (Grade 2) is designed to teach spelling in the context of a sentenee. Using this method teaches a student the proper spelling and meaning of a test word. There are 40 lessons and 400 words in this Level 2 program. Common second grade words are used. Users cannot readily ehange words or sentences in *Spelling In Context.*

Each word is flashed on the TV screen for a brief period of time. When the word disappears a sentence is displayed. The sentence has a blank where the test word should appear. The student types in the word and presses ENTER on the keyboard. If the word is correct, a positive reinforcement statement is displayed and the student is allowed to proceed to the next word. On the other hand, if a word is mispelled a corrective message is given. The word is flashed on the screen a second time and the sentence is displayed again. If the user misspells the word a second time, the correct spelling of the word is presented at the bottom of the display. The student will not be allowed to proceed to the next word until the misspelled word is spelled correctly. If a student scores less than 90 percent on the lesson, the misspelled words are reviewed at the end of the lesson.

I play-tested *Spelling In Context, Level 2* on Stacey, the niece of my upstairs neighbor. This was Stacey's first contact with a computer. I'm happy to report Stacey wants her mother to buy a computer and her mother is very responsive after seeing the CoCo in action. With very little tutoring Stacey was running the program like a pro.

Spelling In Context, Level 2 uses the extra color set of *PMODE3:* black, blue, orange, and buff. Text is displayed in upper and lower case. Some text lettering is hard to read. This might be correeted by increasing the size of the letters. I found it hard to read the blue text on the black background. The buff and orange text read OK on the black background.

Everyone involved in testing *Spelling In Context, Lever 2* was impressed with the overall program. It is obvious that much time was spent selecting the spelling words and text used in this program. The program material includes a well written manual. The sound and musical tunes generated by *Spelling In Context* are adequate.

I think more positive reinforcement should be given for high scoring. Additional sound or graphic routines could give the added reinforcement. Stacey was a little let down by the ending routines used with this program.

Bertamax Inc., the producers of *Spelling In Context, Level2,* has a large line of educational software available for the CoCo. If these programs are as well thought out as *Spelling In Context, Level 2* they will make a welcome addition to a program library. Most Bertamax programs are available ou tape or disk.

(Bertamax Inc., 101 Nickerson Street, Suite 202, Seattle, WA 98109, 32K tape $45.50, disk $49.50)

—Gabriel Weaver

PAINT SHOP

By Don Inman
Rainbow Contributing Editor

This month's Using Graphics article will "brush up" on the *PAINT* statement of Extended Color BASIC. In the book "TRS-80 Color Computer Graphics," I used a program that filled a can with paint using the *PAINT* statement. Since there was no top on the can, the paint "spilled" out of the can and filled the screen. That program was used to emphasize two important features of *PAINT*:

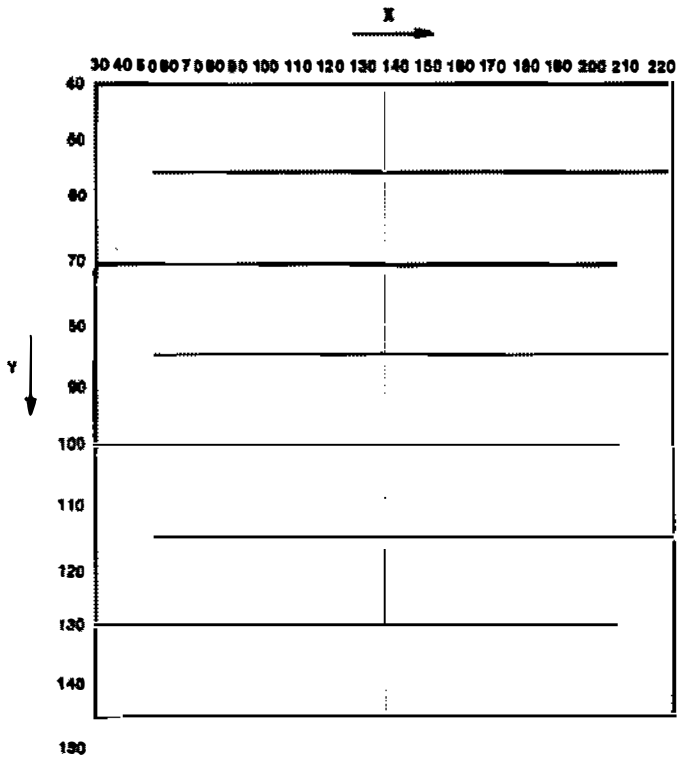
1) If you *PAINT* from inside a figure that is not closed, the paint will "spill" outside the figure.
2) Once you begin to *PAINT* an area, there is no way to stop until the painting has been completed.

Paint Can

The *PAINT* instruction works in deliberate, mysterious ways. Suppose you design a paint can with many baffles distributed about the inside. You also put a top on the can and devise some way to fill the can with paint after it has been closed. The paint can may be represented on a graphic screen as shown in Figure 1.

(Don Inman is the acknowledged master of microcomputer graphics and the author of a large number of books, including TRS-80 Color Computer Graphics and Assembly Language Graphics for the TRS-80 Color Computer with Kurt Inman.)

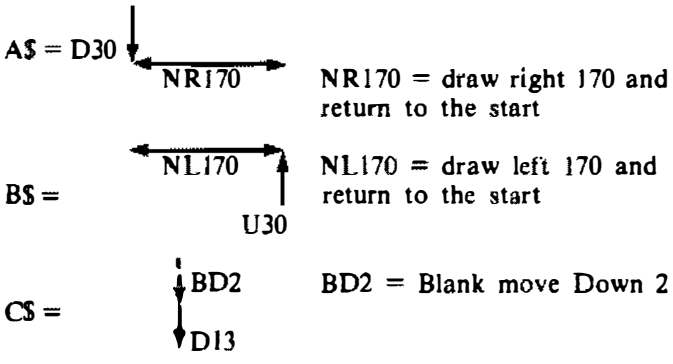
Figure 1 — Closed Can With Baffles



At the top of each baffle, a small space is provided for the paint to move from right to left or from left to right. At the side of each baffle, space is left for the paint to move upward or downward. The can with baffles will be drawn with the following program lines.

```
PMODE 3,1                          'set the mode
PCLS: SCREEN 1,1                   'clear screen — color set 1
A$="D30;NR170"                     'define strings
B$="U30;NL170"                     'left side
C$="BD2;D13"                       'bottom, right, top
DRAW"BM30,40;XA$;XA$;XA$;"         'center
DRAW"D15;R190;XB$;XB$;XB$;U15
;L190"
DRAW"BM125,40;XC$;XC$;XC$;XC$
;XC$;XC$;XC$;"
```

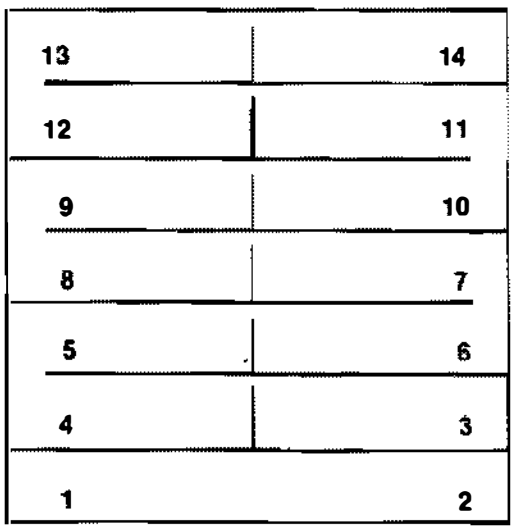
The greatest part of the can is drawn using pieces formed by the substrings:



These substrings are called by the *DRAW* statement by XA$, XB$ and XC$.

The idea behind our baffled can is to investigate how the *PAINT* statement will fill the can. If you start the fill at different points within the can, will the order of fill change?

You might give each section of the can a number. Then start painting from a different section each time. As an example, I numbered the sections as follows:



If you start painting in section 1, will the order of fill be: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14?

The *Baffled Can* program that follows may be used for your investigation. Two inputs are requested. After the can has been drawn, the computer asks:

```
PAINT FROM WHERE (X,Y)?
```

Input the X, Y coordinates for the point at which you wish to start painting. The compute will then ask for a color code.

```
PAINT FROM WHERE (X,Y)?127,144
COLOR(6,7,8)?
```

Input one of the color codes: 6, 7, or 8. Then watch closely as the can fills with paint.

```
99 REM-SET SCREEN AND VARIABLES
100 PMODE 3,1
110 PCLS:CLS
120 A$="D30;NR170"
130 B$="U30;NL170"
140 C$="BD2;D13"
150 '
199 REM-DRAW CAN
200 DRAW"BM30,40;XA$;XA$;XA$;"
210 DRAW"D15;R190;XB$;XB$;XB$;U1
5;L190"
220 DRAW"BM125,40;XC$;XC$;XC$;XC
$;XC$;XC$;XC$;"
230 '
299 REM-GET INPUTS
300 INPUT"PAINT FROM WHERE (X,Y)
";X,Y
310 INPUT"COLOR (6,7,8)";C
320 '
399 REM-PAINT
```

semicolon must follow each substring — even at the end of a line

```
400 SCREEN 1,1
410 PAINT(X,Y),C,B
420 '
499 REM-PAUSE
500 I$=INKEY$
510 IF I$="" THEN 500 ELSE 110
```

press any key to PAINT again

The following inputs are suggested for the beginning X,Y coordinates. Use any color you wish. If you are quick enough, you might tabulate the order in which the sections are filled.

Starting Section	X,Y	Order of Filling
1	123,144	
2	127,144	
3	127,129	
4	123,129	
5	123,114	
6	127,114	
7	127,99	
8	123,99	
9	123,84	
10	127,84	
11	127,69	
12	123,69	
13	123,54	
14	127,54	

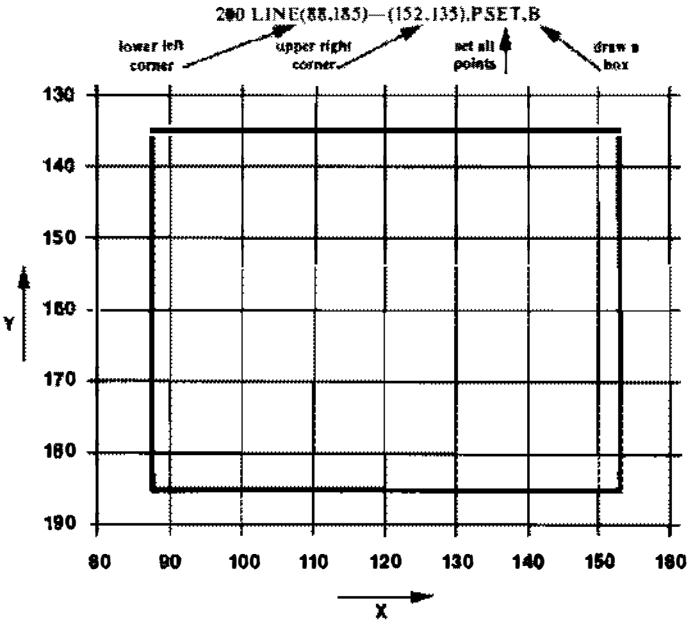
Hint: Widen the gap between the center baffles, and you may be able to see what's going on.

Paint Shop

The next program is a variation of Program 6-2 which appeared in Chapter 6A of "TRS-80 Color Computer Graphics." An automobile paint shop is drawn at the lower center of the screen. An unpainted car drives in from the right side of the screen. A door of the paint shop opens, and the car enters. The car is painted a random color in the paint shop. Then a second door opens, and the painted car leaves the paint shop and exits to the left of the screen.

The rectangular paint shop is drawn (see Figure 2) by the *LINE* statement:

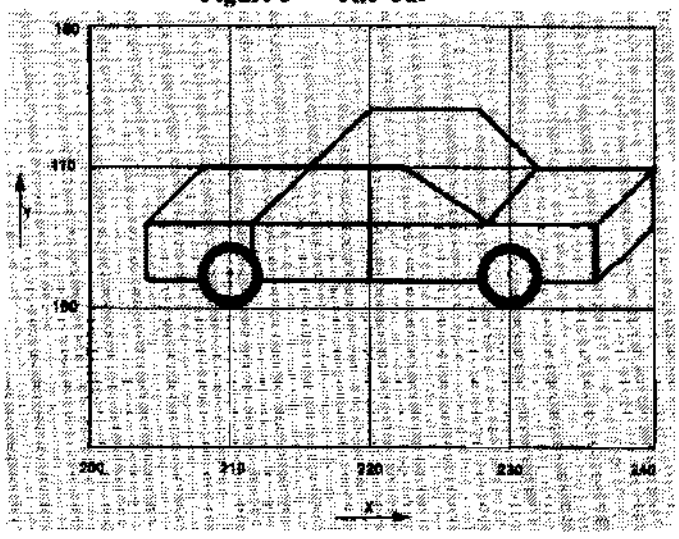
Figure 2 — Paint Shop



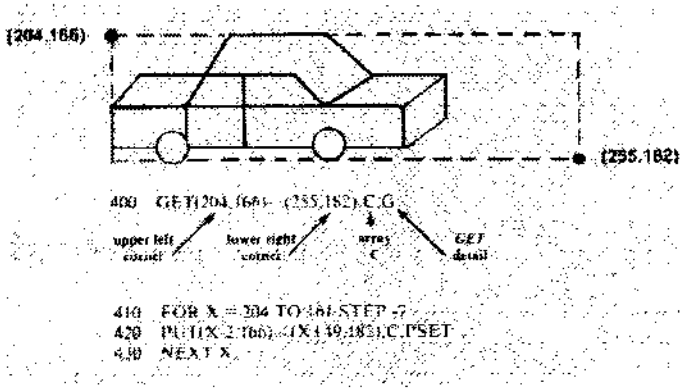
The unpainted car is drawn in the lower right corner of the screen as shown in Figure 3. This is accomplished by one long *DRAW* statement and two *CIRCLE* statements. The drawing starts at position (208,170).

```
300 DRAW"BM 208,170;R8E4R8F4R8G4L8NE4;
    H4L8G4L8NE4D4R4BR4NU4R8NU8;
    R8BR4R4NU4E4U4G4L24"
310 CIRCLE(210,178),5: CIRCLE(230,178),5
```

Figure 3 — The Car



The car is moved to a point in front of the paint shop door by one *GET* statement and a series of *PUT* statements using a *FOR-NEXT* loop. A step of two X values was used to move the car. The *GET* and *PUT* statements include a blank area at the rear of the car so that a separate *PUT* statement would not be needed to erase the "old" car when it is moved to a new position.



Notice that the size of the *PUT* and *GET* arrays are each 51 by 16. The array is dimensioned in line 120 by:

```
120 DIM C(22)
```

A single dimensioned array may be used. The size of the array is roughly calculated from the size of the areas used in the *PUT* and *GET* statements from:

(length * width) / k where k = 40 for PMODES 3 & 4
= 80 for PMODES 1 & 2
= 160 for PMODE 0

In this case: length = 255-204 = 51
width = 182-166 = 16
k = 40
(51*16)/40 = 816/40 = 20.4

22 was actually needed

A series of *PSET* statements (using the background color) opens the garage door.

```
500 FOR Y = 185 TO 160 STEP -1
510 PSET(152,Y,1)      <-- background color
520 NEXT Y
```

The car then moves inside, and the shop door is closed by using the foreground color:

```
700 FOR Y = 160 TO 185
710 PSET(152,Y,4)      <-- foreground color
720 NEXT Y
```

The car is then painted by:

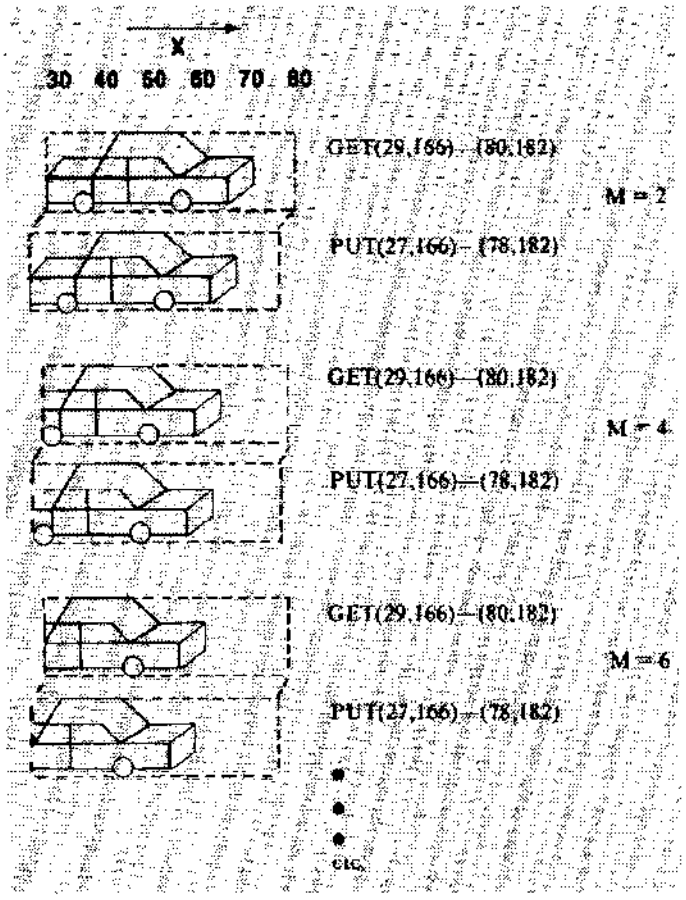
```
730 R = RND(3)+1       <-- colors 2,3,or 4
740 DATA 107,172,119,168,131,172,135,174,131,
    176,121,176,113,176,103,176,107,178,127,178
750 FOR N = TO 10
760 READ A,B
770 IF N = 9 THEN R = 2   <-- wheels always color 2
780 PAINT(A,B),R,4
790 FOR W=1 TO 50; NEXT W   <-- time delay
800 NEXT N
```

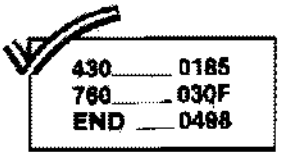
After the car is painted, the opposite paint shop door opens, and the car exits. The car seems to disappear by *GET*ing a fixed area and *PUT*ing it two places to the left.

The next *GET* statement then "loses" two positions from the front of the car as shown in Figure 4. This is done by the lines:

```
970  FOR M = 2 TO 40 STEP 2
980  GET(29,166)—(80,182),C,G
990  PUT(27,166)—(78,182),C,PSET
1000 NEXT M
```

Figure 4 — Disappearing Car





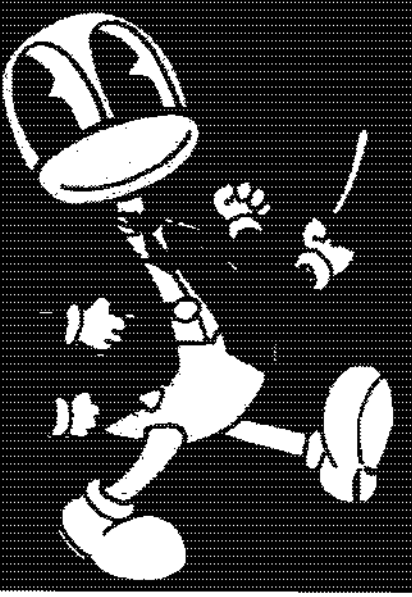
The listing:

```
99 REM-SET SCREEN AND VARIABLES
100 PMODE 3,1:PCLS
110 SCREEN 1,0
120 DIM C(22),E(22)
130 '
199 REM-DRAW PAINT SHOP
200 LINE(88,185)-(152,135),PSET,
B
210 '
299 REM-DRAW CAR
300 DRAW"BM208,170;R8E4R8F4R8B4L
8NE4;H4L8G4L8NE4D4R4BR4NU4R8NU8;
R8BR4R4NU4E4U484L24"
310 CIRCLE(210,178),5:CIRCLE(230
,178),5
320 '
399 REM-GET CAR AND MOVE
400 GET(204,166)-(255,182),C,G
410 FOR X=204 TO 161 STEP-2

420 PUT(X-2,166)-(X+49,182),C,PS
ET
430 NEXT X
440 '
499 REM-OPEN SHOP DOOR
500 FOR Y=185 TO 160 STEP-1
510 PSET(152,Y,1)
520 NEXT Y
530 '
599 REM-MOVE INTO SHOP
600 GET(161,166)-(212,182),C,G
610 FOR X=161 TO 103 STEP-2
620 PUT(X-2,166)-(X+49,182),C,PS
ET
630 NEXT X
640 '
699 REM-CLOSE DOOR AND PAINT CAR
700 FOR Y=160 TO 185
710 PSET(152,Y,4)
720 NEXT Y
730 R=RND(3)+1
740 DATA 107,172,119,168,131,172
,135,174,131,176,121,176,113,176
,103,176,107,178,127,178
750 FOR N=1 TO 10
760 READ A,B
765 IF N=9 THEN R=2
770 PAINT(A,B),R,4
780 FOR W=1 TO 50:NEXT W
790 NEXT N
810 '
899 REM-RAISE DOOR AND MOVE CAR
900 FOR Y=185 TO 160 STEP-1
910 PSET(88,Y,1)
920 NEXT Y
930 GET(101,166)-(152,182),C,G
940 FOR X=101 TO 31 STEP-2
950 PUT(X-2,166)-(X+49,182),C,PS
ET
960 NEXT X
970 FOR M=2 TO 40 STEP 2
980 GET(29,166)-(80,182),C,G
990 PUT(27,166)-(78,182),C,PSET
1000 NEXT M
1099 REM CLOSE DOOR AND RESTORE
1100 FOR Y = 160 TO 185
1110 PSET(88,Y,4)
1120 NEXT Y
1130 RESTORE: GOTO 300
```

RAINBUG V

The conclusion of the Rainbug monitor series by Dan Downard, *Rainbow* technical editor.



For the final installment of *RAINBUG*, the machine language monitor for the CoCo, we will review all of the available commands in detail. A few are new this month such as G, C and B. These commands give you the ability to set breakpoints and jump to other program locations. All registers may be examined and changed using the R command. As you may know, the Motorola ASSIST-09 monitor was used as a guideline for *RAINBUG*, so if more information is necessary find a copy of M6809PM(AD)—MC6809 MC6809E Microprocessor Programming Manual. At the same time you will have to have a programmable timer to implement the trace, or single-step, function and that was outside the scope of this series. If anyone is successful interfacing a MC6840 programmable timer, let's hear from you.

Command Format

RAINBUG accepts three possible command formats depending on the operation. Examples are as follows:

R CR
M SPACE EXPRESSION CR
O SPACE EXPRESSION1 SPACE EXPRESSION2 CR

Notice that expression is used instead of address. An expression can be a combination of addresses. Elementary operators such as "+" and "-" can be used. Leading zeros are unnecessary. There are four special characters:

M—current memory pointer
W—Window value set by W command
P—Current program counter
@—Indirect address

Examples:

M—W	— Current memory pointer minus window value
P—100	— Current program counter minus $100
100@	— Value of two bytes at address $100
1234	— Hex address $1234

As you can see, the expression analyzer is the key to the monitor and can be used to your advantage as you become familiar with it. Any non-hex input other than those noted above will abort a command. A space is used between all commands and expressions and all commands are terminated by a carriage return.

"That about does it for RAINBUG. *It gives you all you need to know to do your own machine language programming."*

Commands

B — Breakpoint	Add or delete up to eight breakpoints
B CR	Prints all breakpoints
B SP - CR	Deletes all breakpoints
B SP EXP CR	Adds a breakpoint at EXP

(Dan Downard is an electrical engineer and has been involved in electronics for 24 years through ham radio (K4KWT). His interest in computers began about five years ago and he has built several 68XX systems.)

B SP - EXP CR	Deletes a break at EXP
C — Call	Call a user subroutine
C CR	Call routine at current program counter
C SP EXP CR	Call routine at address EXP
DR— Disk Read	Read track and sector after prompts at address $2000
DW— Disk Write	Writes 256 bytes at $2000 to track and sector as prompted
E — Encode	Encode a postbyte
E SP H , X CR	Return 5-bit offset from X
E SP HHHH,PCR CR	Return two byte PCR offset
G — Go	Execute user program
G CR	Execute at current program counter
G SP EXP CR	Execute at address EXP
L — Load tape	Same as CLOADM
M — Memory examine	Examine/change memory
M SP EXP CR	Examine address EXP
HHHH /	Examine address HHHH
/	Examine at address of last examine/change
EXP	Replace byte with EXP
SP or , or /	Go to next address
LF	Print next address and byte value
UA	Print previous address and byte value
CR	Terminate command
' TEXT '	Enter ASCII TEXT
O — Offset	Compute branch offsets
O SP FROM SP TO CR	Computes one or two byte offset *FROM-TO*
R — Register	Examine/change registers
R CR	Examine registers
SP	Skip to next register
EX P SP	Change and advance to next register
CR	Terminate command
P — Punch	Same as *CSAVEM* command
P SP xxxx SP yyyy SP zzzz SP file CR	Save machine language program. xxxx-start yyyy-end zzzz-execute file—BASIC filename
V — Verify	Same as BASIC *SKIPF*
@ — Printer	Toggles printer port

Summary

That about does it for *RAINBUG*. It gives you all you need to know to do your own machine language programming. Be extremely careful using the disk commands. All input has to be in *HEX*. Rather than type in the listing I would suggest you obtain a copy of *Rainbow On Tape* as it will have the entire program and save you hours debugging. The program is position independent and can be relocated anywhere in memory. At present it is located at $3000. Addresses immediately below $3000, or wherever you wish to locate the program, are used for the stack. Try to avoid this area. Good luck and happy programming.

```
                        00100 *RAINBUS
                        00110 *DAN DOWNARD     REV 4
                        00111 *LINES 100-2100 AND LINES
                        00112 *2885-4870 APPEARED IN PARTS
                        00113 *1 THRU 4 OF THIS SERIES
3000                    00120         ORG     $3000
                        02105 *COLD START
319E 30  8C FD          02150 RESET   LEAX    RESET,PCR
31A1 9F  9D             02160         STX     $9D
31A3 20  12             02170         BRA     MONTR1
                        02175 *ERROR MESSAGE
31A5     3F04           02180 ERRMSG  FDB     $3F04
31A7 30  8C FB          02190 ERROR   LEAX    ERRMSG,PCR
31AA BD  E2             02200         BSR     PDATA1
31AC 16  0418           02210         LBRA    CMD
                        02215 *MONITOR PROMPT
31AF     52             02220 SIGNON  FCC     /RAINBUG/
31B6     04             02230         FCB     $04
                        02235 *SETUP SOFTWARE INTERRUPT
31B7 BE  3588           02240 MONTR1  LDX     $SWI
31BA BF  0107           02250         STX     $107
31BD 86  7E             02260         LDA     #$7E
31BF B7  0106           02270         STA     $106
                        02275 *SETUP STACK AND START
31C2 32  8D FE0F        02280         LEAS    -22+STACK,PCR
31C6 C6  16             02285         LDB     #$16
31C8 4F                 02290 MONTR2  CLRA
31C9 1F  8B             02300         TFR     A,DP
31CB 8D  02             02310         BSR     MONTR
31CD 20  F9             02320         BRA     MONTR2
31CF 10FF 365B          02330 MONTR   STS     RSTACK
31D3 30  8C D9          02360         LEAX    SIGNON,PCR
31D6 8D  AE             02370         BSR     PDATA
31D8 16  03E4           02380         LBRA    CMD
```

```
31DB                02385          RMB     25
                    02555 *TABLE OF COMMANDS
            31F4    02560 CMDTBL   EQU     *
31F4        04      02565          FCB     4
31F5        42      02570          FCC     /B/
31F6        0211    02580          FDB     BKPT-*
31F8        04      02585          FCB     4
31F9        43      02590          FCC     /C/
31FA        02AF    02600          FDB     CALL-*
31FC        04      02605          FCB     4
31FD        44      02610          FCC     /D/
31FE        018A    02620          FDB     DISK-*
3200        04      02625          FCB     4
3201        45      02630          FCC     /E/
3202        0061    02640          FDB     ENCDE-*
3204        04      02645          FCB     4
3205        47      02650          FCC     /G/
3206        0266    02660          FDB     GO-*
3208        04      02665          FCB     4
3209        4C      02670          FCC     /L/
320A        72F4    02680          FDB     LOAD-*
320C        04      02685          FCB     4
320D        4D      02690          FCC     /M/
320E        FDF2    02700          FDB     CMEM-*
3210        04      02705          FCB     4
3211        40      02710          FCC     /@/
3212        00EE    02720          FDB     PRINT-*
3214        04      02725          FCB     4
3215        4F      02730          FCC     /O/
3216        001F    02740          FDB     OFFS-*
3218        04      02745          FCB     4
3219        50      02750          FCC     /P/
321A        00F4    02760          FDB     PUNCH-*
321C        04      02765          FCB     4
321D        52      02770          FCC     /R/
321E        0296    02780          FDB     REG-*
3220        04      02785          FCB     4
3221        53      02790          FCC     /S/
3222        0330    02800          FDB     STLEV-*
3224        04      02805          FCB     4
3225        54      02810          FCC     /T/
3226        0437    02820          FDB     TRACE-*
3228        04      02825          FCB     4
3229        56      02830          FCC     /V/
322A        73C2    02840          FDB     VER-*
322C        04      02845          FCB     4
322D        57      02850          FCC     /W/
322E        FE5E    02860          FDB     CWINDO-*
3230        04      02865          FCB     4
3231        58      02870          FCC     /X/
3232        002B    02880          FDB     EXIT-*
3234        FE      02882          FCB     -2
                    04875 *SET/RESET BREAKPOINTS
3407 27     25      04880 BKPT     BEQ     BKDSP
3409 17     FC89    04890          LBSR    BLDNUM
340C 27     30      04900          BEQ     BKADD
340E 81     2D      04910          CMPA    #$2D
3410 26     43      04920          BNE     BKERR
3412 17     FCB0    04930          LBSR    BLDNUM
3415 27     04      04940          BEQ     BKDLE
3417 7F     3561    04950          CLR     BKPTCT
341A 39             04960 BKRTS    RTS
341B BD     44      04970 BKDLE    BSR     BKSET
341D 5A             04980 BKDLP    DECB
341E 2B     35      04990          BMI     BKERR
3420 AC     A1      05000          CMPX    ,Y++
3422 26     F9      05010          BNE     BKDLP
3424 AE     A1      05020 BKDLM    LDX     ,Y++
3426 AF     3C      05030          STX     -4,Y
3428 5A             05040          DECB
3429 2A     F9      05050          BPL     BKDLM
342B 7A     3561    05060          DEC     BKPTCT
342E BD     31      05070 BKDSP    BSR     BKSET
3430 27     E8      05080          BEQ     BKRTS
3432 30     A1      05090 BKDSL    LEAX    ,Y++
3434 17     FD2F    05100          LBSR    OUT4HS
3437 5A             05110          DECB
3438 26     FB      05120          BNE     BKDSL
343A 17     FD46    05130          LBSR    PCRLF
343D 39             05140          RTS
343E BD     21      05150 BKADD    BSR     BKSET
3440 C1     08      05160          CMPB    #8
3442 27     11      05170          BEQ     BKERR
3444 A6     84      05180          LDA     ,X
3446 E7     84      05190          STB     ,X
3448 E1     84      05200          CMPB    ,X
344A 26     09      05210          BNE     BKERR
344C A7     84      05220          STA     ,X
```

```
344E 5A          05230 BKADL   DECB
344F 2B 07       05240         BMI     BKADT
3451 AC A1       05250         CMPX    ,Y++
3453 26 F9       05260         BNE     BKADL
3455 16 01C6     05270 BKERR   LBRA    CMDBAD
3458 AF A4       05280 BKADT   STX     ,Y
345A 6F 31       05290         CLR     -B*2+1,Y
345C 7C 3561     05300         INC     BKPTCT
345F 20 CD       05310         BRA     BKDSP
3461 BE 3199     05320 BKSET   LDX     NUMBER
3464 31 8D 00FA  05330 BKLDR   LEAY    BKPTBL,PCR
3468 F6 3561     05340         LDB     BKPTCT
346B 39          05350         RTS
                 05355 *GOTO PROGRAM
346C 8D 01       05360 GO      BSR     GOADDR
346E 3B          05370         RTI
                 05375 *STORE ADDRESS ON STACK
346F 35 30       05380 GOADDR  PULS    Y,X
3471 34 10       05390         PSHS    X
3473 26 1A       05400         BNE     GONDFT
3475 8D ED       05410         BSR     BKLDR
3477 AE E4       05420         LDX     ,S
3479 5A          05430 ARMBLP  DECB
347A 2B 1B       05440         BMI     ARMBK2
347C A6 30       05450         LDA     -8*2,Y
347E AC A1       05460         CMPX    ,Y++
3480 26 F7       05470         BNE     ARMBLP
3482 B1 3F       05480         CMPA    #$3F
3484 26 03       05490         BNE     ARMNSW
3486 B7 3582     05500         STA     SWIBFL
3489 7C 3584     05510 ARMNSW  INC     MISFLG
348C 16 01D4     05520         LBRA    DOT
348F 17 FCAD     05530 GONDFT  LBSR    CDNUM
3492 ED E4       05540         STD     ,S
3494 17 FFC0     05550 ARMBK2  LBSR    BKLDR
3497 70 3561     05560         NEG     BKPTCT
349A 5A          05570 ARMLOP  DECB
349B 102B FF7B   05580         LBMI    BKRTS
349F A6 B4       05590         LDA     [,Y]
34A1 A7 30       05600         STA     -B*2,Y
34A3 B6 3F       05610         LDA     #$3F
34A5 A7 B1       05620         STA     [,Y++]
34A7 20 F1       05630         BRA     ARMLOP
                 05635 *CALL SUBROUTINE
34A9 8D C4       05640 CALL    BSR     GOADDR
34AB 35 7F       05650         PULS    U,Y,X,DP,D,CC
34AD AD F1       05660         JSR     [,S++]
34AF 17 0104     05670 GOBRK   LBSR    BKPNT
34B2 20 FB       05680         BRA     GOBRK
                 05685 *EXAMINE/CHANGE REGISTERS
34B4 8D 23       05690 REG     BSR     REGPRT
34B6 4C          05700         INCA
34B7 8D 21       05710         BSR     REBCHB
34B9 39          05720         RTS
34BA 50          05730 REGNSK  FCC     /P/
34BB 43          05740         FCC     /C/
34BC FF          05750         FCB     -1
34BD 13          05760         FCB     19
34BE 41          05770         FCC     /A/
34BF 00          05780         FCB     0

34C0 0A          05790         FCB     10
34C1 42          05800         FCC     /B/
34C2 00          05810         FCB     0
34C3 0B          05820         FCB     11
34C4 58          05830         FCC     /X/
34C5 FF          05840         FCB     -1
34C6 0D          05850         FCB     13
34C7 59          05860         FCC     /Y/
34C8 FF          05870         FCB     -1
34C9 0F          05880         FCB     15
34CA 55          05890         FCC     /U/
34CB FF          05900         FCB     -1
34CC 11          05910         FCB     17
34CD 53          05920         FCC     /S/
34CE FF          05930         FCB     -1
34CF 01          05940         FCB     1
34D0 43          05950         FCC     /C/
34D1 43          05960         FCC     /C/
34D2 00          05970         FCB     0
34D3 09          05980         FCB     9
34D4 44          05990         FCC     /D/
34D5 50          06000         FCC     /P/
34D6 00          06010         FCB     0
34D7 0C          06020         FCB     12
34D8 00          06030         FCB     0
                 06035 *PRINT REGISTERS
34D9 4F          06040 REBPRT  CLRA
34DA 30 E8 10    06050 REGCHG  LEAX    4+12,S
34DD 34 32       06060         PSHS    Y,X,A
34DF 31 BC D8    06070         LEAY    REGNSK,PCR
34E2 EC A0       06080 REGP1   LDD     ,Y+
34E4 4D          06090         TSTA
34E5 2F 05       06100         BLE     REGP2
34E7 17 FC90     06110         LBSR    OUTCH
34EA 20 F6       06120         BRA     REBP1
34EC B6 2D       06130 REGP2   LDA     #$2D
34EE 17 FCB9     06140         LBSR    OUTCH
34F1 30 E5       06150         LEAX    B,S
34F3 6D E4       06160         TST     ,S
34F5 26 16       06170         BNE     REGCNG
34F7 6D 3F       06180         TST     -1,Y
34F9 27 05       06190         BEQ     REBP3
34FB 17 FC6B     06200         LBSR    OUT4HS
34FE 20 03       06210         BRA     REG4
3500 17 FC65     06220 REGP3   LBSR    OUT2HS
3503 EC A0       06230 REG4    LDD     ,Y+
3505 5D          06240         TSTB
3506 26 DA       06250         BNE     REGP1
3508 17 FC7B     06260         LBSR    PCRLF
350B 35 B2       06270 REGRTN  PULS    PC,Y,X,A
                 06275 *CHANGE REGISTERS
350D 17 FBB3     06280 REGCNG  LBSR    BLDNNB
3510 27 11       06290         BEQ     REGNXC
3512 81 0D       06300         CMPA    #$0D
3514 27 20       06310         BEQ     REGAGN
3516 E6 3F       06320         LDB     -1,Y
3518 5A          06330         DECB
3519 50          06340         NEGB
351A 5B          06350         ASLB
351B 17 FC4C     06360 REGSKP  LBSR    SPACE
```

```
351E 5A          06370          DECD
351F 26  FA      06380          BNE     REGBKP
3521 20  E0      06390          BRA     REG4
3523 A7  E4      06400 REGNXC   STA     ,S
3525 FC  3199    06410          LDD     NUMBER
3528 6D  3F      06420          TST     -1,Y
352A 26  02      06430          BNE     REGTWO
352C A6  82      06440          LDA     ,-X
352E ED  84      06450 REGTWO   STD     ,X
3530 A6  E4      06460          LDA     ,S
3532 81  0D      06470          CMPA    #$0D
3534 26  CD      06480          BNE     REG4
3536 30  8D FAB1 06490 REGAGN   LEAX    TSTACK,PCR
353A C6  15      06500          LDB     #21
353C 35  02      06510 REGTF1   PULS    A
353E A7  80      06520          STA     ,X+
3540 5A          06530          DECB
3541 26  F9      06540          BNE     REGTF1
3543 10EE 88 EC  06550          LDS     -20,X
3547 C6  15      06560          LDB     #21
3549 A6  82      06570 REGTF2   LDA     ,-X
354B 34  02      06580          PSHS    A
354D 5A          06590          DECB
354E 26  F9      06600          BNE     REGTF2
3550 20  B9      06610          BRA     REGRTN
                 06615 *SET TRACE LEVEL
3552 27  07      06620 STLEV    BEQ     STLOFT
3554 17  FBE8    06630          LBSR    CONUM
```



```
3557 FD  3585    06640          STD     SLEVEL
355A 39          06650          RTS
355B 30  6E      06660 STLOFT   LEAX    14,S
355D BF  3585    06670          STX     SLEVEL
3560 39          06680          RTS
                 06685 *VARIABLES
3561     00      06690 BKPTCT   FCB     0
3562             06700 BKPTBL   RMB     16
3572             06710 BKPTOP   RMB     16
3582             06720 SWIBFL   RMB     1
3583             06725 SWICNT   RMB     1
3584             06730 MISFLG   RMB     1
         2FEB    06740 TSTACK   EQU     -21+CMEM
3585             06750 SLEVEL   RMB     2
3587             06760 TRACEC   RMB     2
3589             06770 LASTDP   RMB     2
                 06772 *SOFTWARE INTERRUPT ROUTINE
358B 6A  8C F5   06775 SWI      DEC     SWICNT,PCR
358E 17  00E5    06785          LBSR    LDDP
3591 EE  6A      06790          LDU     10,S
3593 33  5F      06800          LEAU    -1,U
3595 7D  3582    06810          TST     SWIBFL
3598 26  10      06820          BNE     SWIDNE
359A 17  FEC7    06830          LBSR    BKLDR
359D 50          06840          NEGB
359E 5A          06850 SWILP    DECB
359F 2B  09      06860          BMI     SWIDNE
35A1 11A3 A1     06870          CMPU    ,Y++
35A4 26  FB      06880          BNE     SWILP
35A6 EF  6A      06890          STU     10,S
35A8 20  0C      06900          BRA     BKPNT
35AA 7F  3582    06910 SWIDNE   CLR     SWIBFL
35AD 32  62      06911          LEAS    2,S
35AF 30  8D FBF2 06912 ERROR1   LEAX    ERRMSG,PCR
35B3 17  FBD0    06913          LBSR    PDATA
35B6 17  00F2    06920 BKPNT    LBSR    REGPRS
35B9 20  07      06930          BRA     CMDNEP
35BB 17  FF1B    06931 REGPRS   LBSR    REGPRT
35BE 39          06932          RTS
                 06935 *COMMAND HANDLER
35BF 17  FBC1    06940 CMD      LBSR    PCRLF
35C2 17  FE9F    06950 CMDNEP   LBSR    BKLDR
35C5 2A  0D      06960          BPL     CMDNOL
35C7 50          06970          NEGB
35C8 F7  3561    06980          STB     BKPTCT
35CB 5A          06990 CMDDDL   DECB
35CC 2B  06      07000          BMI     CMDNOL
35CE A6  30      07010          LDA     -8*2,Y
35D0 A7  B1      07020          STA     [,Y++]
35D2 20  F7      07030          BRA     CMDDDL
35D4 AE  6A      07040 CMDNOL   LDX     10,S
35D6 BF  3195    07050          STX     PCNTER
35D9 86  23      07060          LDA     #$23
35DB 17  FB9C    07070          LBSR    OUTCH
35DE 33  E4      07080          LEAU    ,S
35E0 FF  3261    07090          STU     PSTACK
35E3 4F          07100          CLRA
35E4 5F          07110          CLRB
35E5 FD  3199    07120          STD     NUMBER
35E8 FD  3584    07130          STD     MISFLG
```

```
35EB C6   02      07140          LDB   #2
35ED 34   07      07150          PSHS  D,CC
35EF 17   F846    07160          LBSR  READ
35F2 30   8D FA6D 07170          LEAX  2+CMPADP,PCR
35F6 81   2F      07180          CMPA  ##2F
35F8 27   4B      07190          BEQ   CMDXQT
35FA 81   20      07200 CMD2     CMPA  ##20
35FC 23   14      07210          BLS   CMDGOT
35FE 34   02      07220          PSHS  A
3600 6C   5F      07230          INC   -1,U
3602 81   2F      07240          CMPA  ##2F
3604 27   4B      07250          BEQ   CMDMEM
3606 17   FB03    07260          LBSR  BLDHXC
3609 27   02      07270          BEQ   CMD3
3608 6A   5E      07280          DEC   -2,U
360D 17   FB2B    07290 CMD3     LBSR  READ
3610 20   E8      07300          BRA   CMD2
3612 80   0D      07310 CMDSQT   SUBA  ##0D
3614 A7   5D      07320          STA   -3,U
3616 30   8D FBDA 07330          LEAX  CMDTBL,PCR
361A E6   80      07340 CMDSCH   LDB   ,X+
361C 2A   07      07350          BPL   CMDSME
361E 10FE 3261    07360 CMDBAD   LDS   PSTACK
3622 16   FB82    07370          LBRA  ERROR
3625 5A           07380 CMDSME   DECB
3626 E1   5F      07390          CMPB  -1,U
3628 24   03      07400          BHS   CMDSIZ
362A 3A           07410 CMDFLS   ABX
362B 20   ED      07420          BRA   CMDSCH
362D 31   5D      07430 CMDSIZ   LEAY  -3,U
362F A6   5F      07440          LDA   -1,U
3631 80   02      07450          SUBA  #2
3633 A7   5E      07460          STA   -2,U
3635 5A           07470 CMDCMP   DECB
3636 A6   80      07480          LDA   ,X+
3638 A1   A2      07490          CMPA  ,-Y
363A 26   EE      07500          BNE   CMDFLS
363C 6A   5E      07510          DEC   -2,U
363E 26   F5      07520          BNE   CMDCMP
3640 3A           07530          ABX
3641 EC   1E      07540          LDD   -2,X
3643 30   8B      07550          LEAX  D,X
3645 6D   5D      07560 CMDXQT   TST   -3,U
3647 32   C4      07570          LEAS  ,U
3649 AD   1E      07580          JSR   -2,X
364B 16   FF86    07590          LBRA  CMDNXL
364E 6D   5E      07600 CMDMEM   TST   -2,U
3650 2B   CC      07610          BMI   CMDBAD
3652 30   89 FFA2 07620          LEAX  CMEMN-CMPADP,X
3656 FC   3199    07630          LDD   NUMBER
3659 20   EA      07640          BRA   CMDXQT
          2FEB    07650 STACK    EQU   -21+CMEM
3658              07660 RSTACK   RMB   2
                  07665 *TRACE FUNCTION
365D 17   FADF    07670 TRACE    LBSR  CONUM
3660 FD   3587    07680          STD   TRACEC
3663 32   62      07690 DOT      LEAS  2,S
3665 EE   F8 0A   07700 TRCE3    LDU   [10,S]
3668 FF   3589    07710          STU   LASTOP
366B FE   3685    07720          LDU   PTM
366E CC   0701    07730          LDD   ##0701
3671 ED   C9 3685 07740          STD   PTM,U
3675 3B           07750          RTI
                  07755 *CLEAR UP STACK
3676 5F           07760 LDDP     CLRB
3677 1F   9B      07770          TFR   B,DP
3679 A1   63      07780          CMPA  3,S
367B 27   07      07790          BEQ   RTS
367D 10FE 3658    07800          LDS   RSTACK
3681 16   FF2B    07810          LBRA  ERROR1
3684 39           07820 RTS      RTS
                  07825 *DUMMY TIMER ADDRESS
          3685    07830 PTM      EQU   *
          3000    07840          END   CMEM
00000 TOTAL ERRORS
```

Hint . . .

Print Out Disk Directory

If you have a long disk directory and want to see all of it, or if you simply wish to have a hard-copy printout of your directory, one simple command will allow you to do this easily.

Just POKE 111,254:DIR and the entire disk directory will appear on your printer, even if it is too long to be fully displayed on your screen.

Save Storage By Packing Your Data

By B. H. Alsop

Even with a 32K CoCo, BASIC program storage space often becomes tight. Every array item, e.g. A(10), uses five bytes of memory. That's pretty expensive. One can conserve this valuable resource by packing multiple data items in one array. Here's how.

Concept

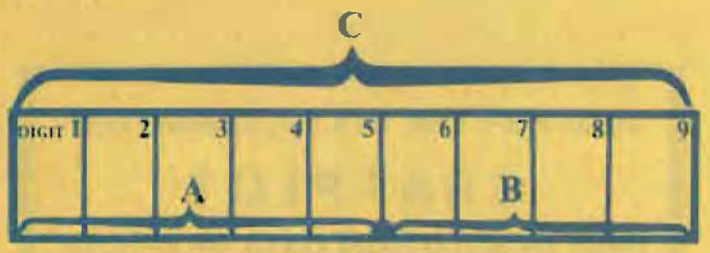


FIGURE 1. PACKING

"C" is the variable in which two other variables (A and B) are stored. In this case, "A" can be up to five digits long and "B" can be up to four digits long. C can't exceed nine digits for a CoCo. Variables A and B can represent integers or floating point numbers.

Sample Program

Listing 1 shows a sample program illustrating packing and unpacking data. Variables A and B each have 1000 items. Normally, one would dimension A and B to 1000 each. This would reserve 10K of memory for these two variables. Here a single variable (C) is dimensioned to 1000. Lines 40 and 50 do the testing to assure that A and B are within range. Line 60 is necessary to move B behind the decimal. Note that a value slightly larger than 0.001 is used to avoid round-off errors. If a value of exactly 0.001 is used, the unpacked value of B can be one too small. Line 70 does the packing. Line 90 unpacks A and line 120 unpacks B. *RUN* the program and you'll see that the input and packed/unpacked values are equal. I had 17508 bytes of memory left when this problem was run.

(Brian Alsop is a Westinghouse engineer, holds amateur and commerical radio licenses and a pilot's license. He has used computers for 15 years to solve large numerical problems.)

Listing 2 shows a similar program where A and B are each dimensioned to 1000. Data is read into each of them. Here the memory remaining is 12644. The net savings in memory is about 5000.

Caveats

One doesn't get something for nothing. If you don't check the input numbers to be within range, you'll get garbage out. It is possible for A to be positive or negative in this packing/unpacking process. The B can only be positive. The reason A can be negative is that INT(-12.7)=-13 and INT(12.7)=12. This is a curiosity I ran into when writing my packing subroutine. I really expected that INT(-12.7)=-12! It appears that the programmers of the INT function defined it with this application in mind.

Extensions

If the range of several variables, say A, B and C, are suitably restricted, there is no reason why three variables can't be packed in one variable. Remember, only one can be plus or minus and it must go before the decimal. The sum of the maximum significant figures can't exceed nine.

If you want to pack floating point numbers, first make them into integers by multiplying by a known power of 10. Truncate with an INT function call, and pack. After unpacking, divide by the same power of 10.

Origin

It seems that in the old days the "big computers" had storage limitations, too. A 16K machine was large back then. Thus, the idea of packing came into being. Where I work, there is a Cray-1 computer which has a 140K fast memory and 3.0 million slower core memory. It generates so much data in a few minutes of running that it can't all be stored on a single three million word tape. Packing is used to conserve tape storage.

It appears that available data to store will always exceed the ability to store it.

Listing 1

```
10 DIM C(1000)
20 FOR I=1 TO 1000
```

```
30 A=RND(99999):B=RND(9999)
40 IF A>99999 OR A<-99999 THEN P
RINT"A>99999 OR A<-99999":STOP
50 IF B>9999 OR B<0 THEN PRINT"B
>9999 OR B<0":STOP
60 B=B*.0001000 1
70 C(I)=A+B
80 PRINT A,
90 A=INT(C(I))
100 PRINT A
110 PRINT INT(B*10000),
120 B=INT(10000*(C(I)-A))
130 PRINT B
140 NEXT I
150 PRINT MEM
```

Listing 2

```
10 DIM A(100),B(100)
20 FOR I=1 TO 100
30 A(I)=RND(99999):B(I)=RND(9999
)
80 PRINT A(I),
100 PRINT A
110 PRINT B(I),
130 PRINT B
140 NEXT I
150 PRINT MEM
```

Back Issue Availability

Back copies of many issues of *the RAINBOW* are still available.

All back issues sell for the single issue cover price—which is $2 for copies of Volume I, Numbers 1-8 (through February, 1982), $2.50 for Volume I, Numbers 9, 10 and 12 (through June except May, 1982) and $2.95 for Volume II, Numbers 9, 10, 11, 12 (March, April, June and July 1983). Also $2.95 for Volume II, Number 2 (September 1983). In addition, there is a $3.50 charge *per order* for postage and handling if sent by United Parcel Service and $6 for orders sent U.S. Mail. UPS *will not* deliver to a postoffice box or to another country. This charge applies whether you want one back issue or all of them.

Most back issues are available on white paper in a reprint form. Issues out of print include May, July, August, September, October, November, and December, 1982 and January, February, April, 1983. VISA, Master Card and American Express accepted. Kentucky residents please add 5 percent state sales tax.

Due to heavy demand, we suggest you order back issues you want now while supplies last.

In addition, copies of the cover *only* of the July, 1982, Anniversary Issue are available separately for $1 each, plus 50 cents shipping and handling. These are suitable for framing.

RAINBOW Info

How To Read Rainbow

Please note that all the BASIC program listings you will find in *the Rainbow* are formatted for a 32-character screen — so they will 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 listings indicates that program is available through our *Rainbow On Tape* service. An order form for this service is on the insert card bound in the magazine.

The Rainbow Seal

The Rainbow Certification Seal is our way of helping you, the consumer. The purpose of the Seal is to certify to you that any product which carries the Seal has been physically seen by us and that it does, indeed, exist.

Manufacturers of products — hardware, software and firmware — are encouraged by us to submit their products to *the Rainbow* for certification. We ascertain that their products are, in actuality, what they purport to be and, upon such determination, award a Seal. This lets you know that we have seen the product and that it does, indeed, exist.

The Seal, however, is not a "guarantee of satisfaction." The certification process is different from the review process. You are encouraged to read our reviews to determine whether the product is right for your needs.

There is absolutely no relationship between advertising in *the Rainbow* and the certification process. Certification is open and available to any product pertaining to CoCo. A Seal will be awarded to *any* commercial product, regardless of whether the firm advertises or not.

We will appreciate knowing of instances of violation of Seal use.

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 that 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 6609 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=&H3F60
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.

What's A CoCo

CoCo is an affectionate name which was first given to the TRS-80 Color Computer by its many fans, users and owners. As such, it is almost a generic term for three computers, all of which are very much alike.

When we use the term CoCo, we refer to the TRS-80 Color Computer, the TDP System-100 Computer and the Dragon-32 Computer. It is easier than using the three "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 TRS-80 Color are, for all purposes, the same computer in a different case, these terms are almost always interchangable.

The Rainbow Check

The small boxes which you see with programs in *the Rainbow* are our *RAINBOW CHECK* program, which is designed to help you type in programs accurately.

The check program will count the number of characters you type in. You can then compare the number the *RAINBOW CHECK* gives you 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 the *RAINBOW CHECK*, type in the program, *CSAVE* it for future use, then type in the command *RUN* and press *ENTER*. Once the program has run, type *NEW* to remove it from that area into which you will be keying programs.

Now, whenever you press the down arrow, CoCo will give you the hexadecimal number of bytes 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 proper BASIC program code.

As the hexadecimal number appears in the upper-left corner of the monitor screen, you may want to clear the screen and press the spacebar five or six times to move the cursor out of the way for easy reading. The *RAINBOW CHECK* counts spaces, too, follow the spacing just as it appears in the magazine.

Here's the program:

```
 10 CLS:IF PEEK(116)=127 THEN
X=32688 ELSE X=16304
 20 CLEAR 25,X-1
 30 IF PEEK(116)=127 THEN X=32688
ELSE X=16304
 40 FOR Z=X TO X+77
 50 READ Y:W=W+Y:PRINT Z,Y;W
 80 POKE Z,Y:NEXT
 70 IF W=5716 THEN 60 ELSE PRINT
"DATA ERROR":STOP
 80 EXEC X:END
 90 DATA 182, 1, 106, 167, 141, 0, 68
 100 DATA 134, 126, 183, 1, 106, 190
 110 DATA 1, 107, 175, 141, 0, 57, 48
 120 DATA 141, 0, 4, 191, 1, 107, 57
 130 DATA 129, 10, 38, 44, 52, 22, 220
 140 DATA 27, 147, 25, 142, 4, 0, 141
 150 DATA 6, 31, 152, 141, 2, 32, 25
 160 DATA 52, 2, 68, 68, 68, 68
 170 DATA 141. 4, 53, 2, 132
 180 DATA 15, 129, 9, 46, 4, 139, 112
 190 DATA 32, 2, 139, 55, 167, 128, 57
 200 DATA 53,22,126,0,0
```