For your TANDY Color Computer

NEW GUINEA K5.95

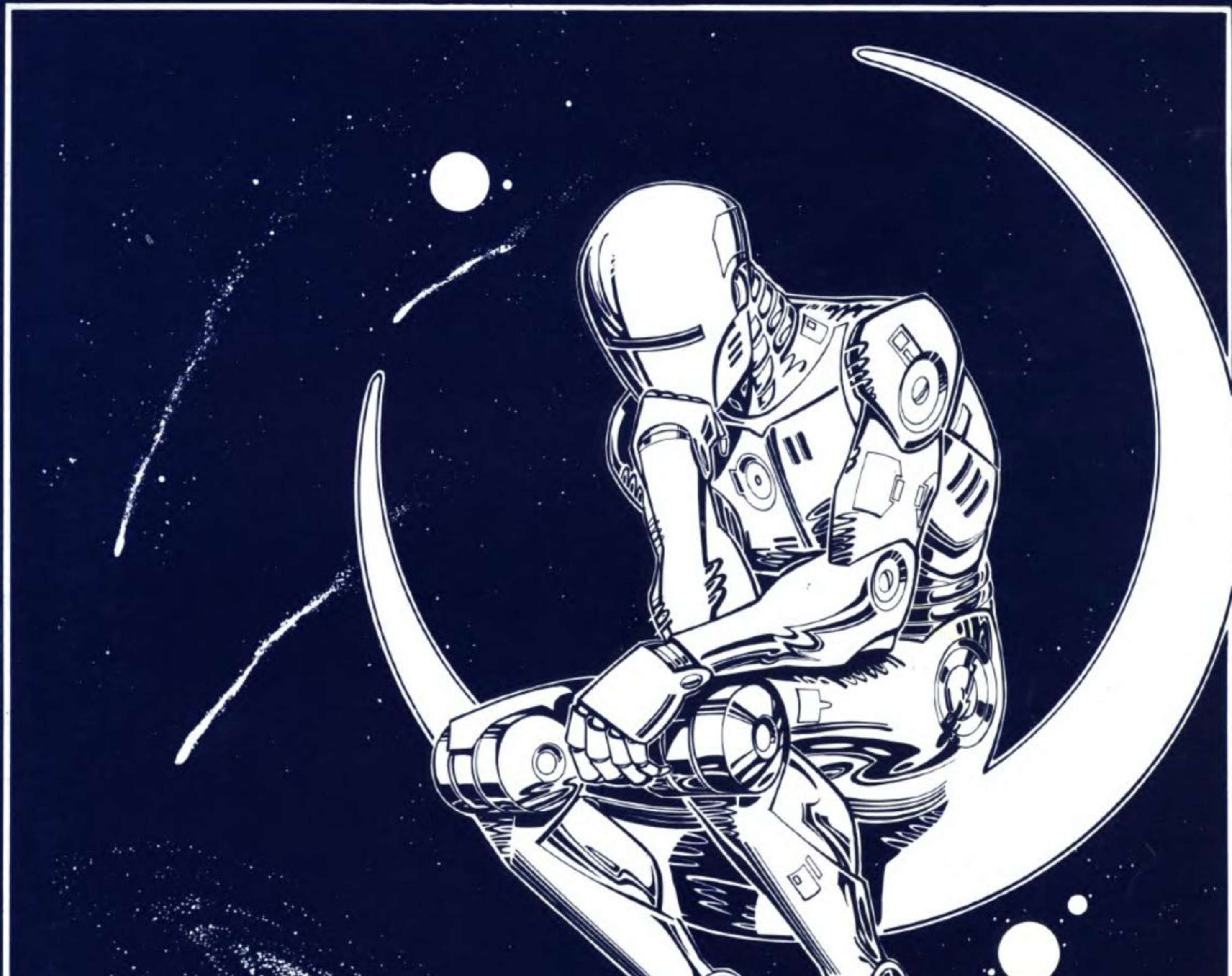
$4.50

AUSTRALIAN RAINBOW

November, 1985

Registered by Australian Post — Publication No. QBG 4009

No.53



AMAZING!!
AMAZING!!
AMAZING!!
CoCo Max®
COLORWARE
File Edit Goodies Font Style
untitled
AMERICAN SCHNOID CO.
'82 '83 '84 '85
Lincl
Saturn
File Edit Goodies Font Style
Pie
AMERICAN SCHNOID CORP.
1985
LABOR 15%
CORP. TAXES 25%
OTHER
File Edit Goodies Font Style
New
Open
Eye4
PARTS
Baby1
AMAZING!!
AMAZING!!
AMAZING!!
$99.95

What is CoCo Max? Simply the most incredible graphic and text creation "system" you've ever seen. You will be generating images like these in minutes.

Who is CoCo Max for? Anyone who has ever held a pencil for fun, school or business will love it. A 6-year old will have fun doodling, a 15-year-old will do class projects, and adults will play with it for hours before starting useful applications *(illustrations, artwork, business graphics, flyers, charts, memos, etc.)* This is one of the rare programs that will be enjoyed by the whole family.

Just point and click to activate *CoCo Max*'s powerful features, including: mirror images, rubber banding, edge tracing, zoom, lasso, sixteen colors, thirty patterns, thirty-two paint brush shapes and textures, undo, rubber stamping, icons, pull down menus, pencil, spray can, eraser, toolbox, and so on, and so on...

The Hi-Res Input Pack is the key to *CoCo Max*'s unmatched power. It gives you direct access to the 49,152 pixels on your screen. That's **12 times** the regular joystick input. It looks like a ROM pack, and plugs into your CoCo or Multipak. Your regular joystick, mouse, or Koala Pad simply plugs into the Hi-Res Pack. Whether you are an artist or an accountant, even if you are the "I can't draw" type, you will be amazed by what you can do with *CoCo Max*.

Price Includes
- Cocomax disk
- HI-RES Input Pack
- Detailed Manual

Disk System: a Multipak or 'Y Adaptor' is needed to plug the disk pack and the HI-RES pack.

SYSTEM REQUIRED:

- Any 64K COCO.
- A standard Joystick, Mouse or Koala Pad.

All these pictures are **unretouched** screen photos or printouts (on an Epson RX-80).

CFM

COMPUTERWARE FOR MICROS.

bankcard welcome here

VISA

Peter & Jillian Collison
11 Grantley Avenue,
Rostrevor S.A. 5073
Phone: (08) 336 6588

AUSTRALIAN RAINBOW

index

UPPER CASE - Article & Program.
Lower Case - Article only.

AUSTRALIAN RAINBOW Publisher and Editor Graham Morphett. Co editor Kevin Mischewski Assistant Editor Sonya Young. With grateful assistance from Brian Dougan, Richard and Judy Bob Thomson, Paul Humphries, Alex Hartmann, Michael Horn, Jim and Sheryl Bentick, Annette Morphett. Cover Art Jim Bentick
ADVERTISING DEADLINES. The 7th of the preceeding month of publication. All advertising is arranged through ToToAdvertising, PO Box 6730, Gold Coast Mail Centre, Qld 4217
OS 9. Kevin Holmes is the contact for os 9 information. He also has access to OS 9 Software from the US. His address is: 39 Pearson St., Narara, NSW, 2250
All programmes in this issue of AUSTRALIAN RAINBOW are available on Rainbow on Tape

Telephone: 075 51 0015 Voice, 075 32 6370 CoCoLink
Printed by: Australian Rainbow Magazine PO Box 1747, Southport, Qld 4215
Founder GREG WILSON

PRINT #-2,

The presses were rolling the magazine out last month as two events took place which will change our CoCo world.

The news of the demise of John Brother's Software Spectrum was one of these events.

Software Spectrum has gone into receivership largely because of piracy.

This company was, apart from ourselves, the largest external organization supporting Tandy products and had about 80% of the software available for CoCo.

Software companies need piracy like they need a hole in the head, and American suppliers are not going to continue to supply to a country which keeps ripping them off.

If you want to see the latest American software continue to come into this country, then BUY it, don't steal it.

The brighter news is that Tandy have seen fit to slash the price of the CoCo.

Our hope is that the influx of new users this action will provide, will learn very quickly about the CoCo and about the friendship and support that surrounds it in this country.

I was in Melbourne during September and amongst other engagements, spent a night with a number of CoCo users at Yarra Valley High School.

I want to thank them for coming so far in such bleak weather to be at the meeting.

We had a wonderful time, and it was terrific to meet so many of the folk who mean so much to the magazine. I lost count of the number of authors who were there - people we especially count as friends.

To you all, thank you. We'll do it again about March, when we'll plan to take a little more time!

(A fuller report can be found in the Club News section of Australian CoCo).

Plans are afoot to hold a BYO Barbeque in the park in Marine Parade, Labardor, Qld, on the 9th of November at 11.30am. All who can make it are invited. A map showing where it will be is to be found on P 51.

Bring the kids - the emphasis will be very much on families and fun!

We have a second phone number for the day time now. It is 075-51-0577. Our old number remains unchanged.

All authors should have heard / be hearing from us about now.

If you have contributed a program or an article to either magazine since August, 1984 and have not received a letter from us, please let me know.

continued on Page 8

REVIEWS

Hardware Review

EARS Pioneers The Way For Spoken Command Response

By Randy Graham

Personal computing is a piece of cake. Just turn it on and start typing; it all appears right there on the screen. If you make a mistake, back up and correct it. Write programs in many languages. Run software for word processing, spreadsheeting — even play games. Print it all out letter-perfect then save it and share it. Nothing to it. Unless you have no hands to type or no eyes to see the screen.

I work with handicapped people. Jonathan was a lineman before his accident. Because he severed his spinal cord, he is confined to a wheelchair. He has very poor use of his arms; he needs help to dress and to eat. When I met him, he was beating everyone in the rehabilitation unit at chess. I thought that a good chess player ought to make a good computer programmer.

Paul has had cerebral palsy since birth. He can get around with crutches although it is difficult; his arms and legs just won't do what his brain tells them. His speech is difficult for most of us to understand. He is a college graduate and would like to be a writer, but trying to type can be a tearful frustration for him.

As a result of a disease, everything looks blurred to Betsy. She can manage to read with a magnifying glass, holding the material close to her face; she has not been able to find a computer screen she can read.

For Jonathan and Paul and Betsy, the keyboard and screen which give us access to all the marvels of personal computers are just as formidable barriers to computing careers as the six-inch curb which locks them out of a shopping mall or an office building. We can cut notches in the curbs and build ramps around the steps. What can we do to open up our favorite activity to some very fine people?

It was for these friends and others that I flew to the RAINBOWfest in Chicago to find out what Rich Parry and Lester Hands were up to at Speech Systems. It was fun meeting them and I became so excited about their work that I brought an EARS home with me so I could join the adventure. EARS (Electronic Audio Recognition System) is a device and program through which a Color Computer responds to spoken commands.

Yes, I said spoken commands! No other computer has it — just CoCo. We are coming along with voice synthesizers which can electronically vocalize text in memory, but going the other way? Little more than a year ago I attended a conference in which an expert in the field said that speech recognition was not even on the horizon. He was wrong. Say "hello" to the expert, EARS.

What You Get

In a nice plastic envelope you get two pieces of hardware, a piece of software and a very good manual. One piece of hardware is a ROM pack which contains the magic and the other piece is a headset. There are no earphones in the device; it is just there to hold the microphone close to your mouth. The headset plugs into the ROM pack and you know where that goes.

The software (tape or disk) contains a machine language program, a utility and several demonstration programs. Load and execute the EARS program and you are in business. To the user, the effect is that all of your Extended Color BASIC ROM still works but some new commands have been added. Time to start learning.

How it Works

LISN is the first new command you meet. Like all BASIC commands, it can be used in direct mode or within a BASIC program. Type `LISN "1"` and say "one." You have taught CoCo/EARS its first word. Let's note that you have also taught it your accent — and language.

EARS is a "speaker-dependent" system. The user teaches the machine and thereby sets up a unique relationship. There is nothing romantic about this; it seems to be the easiest way to do it. A "speaker-independent" system which can recognize any voice in any accent, pitch and inflection must be much more complex. This way, whether I say "one" or "uno" or "ein," when I say it again, EARS will respond with "one." Space is reserved in memory for a speech synthesizer like *Voice*. If you tell EARS to SAY "one," it will speak it from the speech system.

OK, I have taught EARS one word. I can now repeat that process. In its present shape and form, EARS can "learn" 64 strings. Apparently, the program sets up a two-dimension array. One element is a string of up to 16 characters which I type in. The other element is a voice print of what I say into the microphone. EARS calls this pair a "template." Let's see what I can do with this array.

MATCH lets me use the vocabulary list. After this command, I say "one"; there is a noticeable pause, then the screen prints '1'. EARS has searched the array of voice prints, found a match and printed the character string.

Let's note again that the relationship between the input string and the spoken part is what I want it to be. "Gadzooks" can make the computer type out "My name is Randy" if that is the way I made the template.

What if I forget my own list? TLIST will produce the string part of the list of templates for me one screen at a time. TSAVE will save the whole list of strings and voice prints to tape or disk (EARS checks to see which you are using); TLOAD will load a list so I do not have to do it over every time.

Vocal Programming

Now for the great part. I have taught EARS the digits, alphabet and as many BASIC commands as it will hold. Time for another command.

DRECT instructs EARS to accept BASIC statements from me. I can say "one-zero-a-equals-one-two-three-enter" and a BASIC line appears on the screen. Then, "two-zero-print-a-enter" and the second line is in place. I can say "list" and it will (because I added the "enter" to my string); when I say "run," it does it! Primitive? Yes, but I wrote and ran a BASIC program without touching the keyboard! I feel like I have the first horseless carriage in town.

While in DRECT mode, the computer will accept keyboard input, but it will continue to respond to voice commands until turned off with BREAK.

A Fairy Tale

Jonathan is up and dressed and has finished breakfast. It is time to go to work. He rolls down the hall to the den where CoCo/EARS is waiting. He activates a remote control switch by sound or however and the computer comes on and loads EARS which loads the template list. (This last part will require an enhancement that EARS does not have. Right now, Jonathan would have to leave his computer on and only turn his monitor and peripherals on and off.)

He says "Terminal" and the screen says RUN "TERM .BIN",R (ENTER). Before starting to work, like many businesspeople, he says, "Get Dow." "Get" tells the terminal program to go to its autodial/autologon section. "Dow" calls and logs on to DJNS. At the menu he says "news" and the screen shows //NEWS (ENTER). He goes through the news by menu selections, reads the Wall Street Journal online and logs off. Now he says, "Get office." Autodial and logon gets him into the office computer. Perhaps he will first read "Mail" and "Memo" before calling for the file on which he is working. It is slow and tedious, having to give all digits and spelling out words. But, he is working!

This is only just barely a fairy tale. If you devote 36 templates to the letters and digits, you only have 28 left for words. But TCLEAR will empty the template buffer so you can load another list tailored to the specific application you are running. And needless to say, he will need the help of a typist to construct his template lists.

Come on Along

It sometimes seems like the pioneering days of personal computing are over. Nobody builds his own any more — what did I do with that wirewrapping tool? Hardly anybody writes programs — software is too readily available. Everybody is into graphics. Well, if you sometimes look longingly at the old covered wagon gathering dust in the shed, hitch up. A whole new land mass has been opened for exploration.

Someday there will be listening typewriters that write it all out as you dictate. Other control applications will become as common as digital microwave ovens and calculator watches. And perhaps we will have Hal, the conversational computer, by 2001. But you can still get ahead of the parade. EARS is the place to start, and you can only do it with a CoCo!

I have told you about my excitement with this new tool. I would like to hear from others who are experimenting with EARS for handicapped applications. My complete address is 2115 Buford Road, Richmond, VA 23235. Or, maybe we can share our experiences in the "Letters to Rainbow" department.

I did not write this review with EARS — I am glad I did not have to. It would have taken forever, spelling out each word. If your template list is full, there is a pause of a couple of seconds before the computer responds. And, let's face it, EARS can be quirky. Rich tells you to imagine you are training your puppy. That's the way you must speak. Sometimes I say a couple of words that are not on the template list and get up and walk away. When I do that, I remember the old proverb: "I had no shoes, and complained, until I met a man who had no feet."

(Speech Systems, 38W 255 Deerpath Rd., Batavia, IL 60510, $59.95)

Software Review

CINC PAC — Battle of Midway For Victory At Sea

By H. Larry Elman

A few months ago I single-handedly sank the entire Royal Navy in the English Channel; shortly after that, I found a way to mishandle even U.S. Marines into frequent surrender. So the RAINBOW editor, reacting the way any normal government ruler would, gave me an extra star or two and sent me off to command at Midway. Of course, one of the first ridiculous things I succeeded in doing was landing an entire Squadron of B-17 "Flying Fortresses" on the deck of the USS Yorktown. (This is comparable to driving a Boeing 747 airliner through your local car wash.)

Midway was one of the more crucial battles of this century and has become the focus of tremendous study by tacticians, strategists and war-gamers. There are a number of computer games available based on this battle; one of the best is Ark Royal's *CINC PAC — Battle of Midway*. (For the civilians among us, CINC PAC stands for Commander-in-Chief, Pacific Fleet. It is pronounced as if spelled sink-pack.)

In this game, the computer commands the Imperial Japanese Navy, while you command three aircraft carriers, a few squadrons of aircraft and three shore bases (two of which are almost totally useless). Your biggest advantage is the fact (as in the actual battle) that the Navy can read the Japanese code, enabling you to set an ambush.

Believe me, if you fight this battle without using that asset, your Color Computer will cream you! I know; it has beaten me more times than I will publicly admit.

If you have not tried war games (or "Battle Simulations" as some of their authors prefer to call them), then you have probably missed one of home computing's more challenging intellectual pursuits. You need not be either a militarist or a history nut to enjoy these descendants of chess.

War games can be broken into a number of subcategories, of which two are worth a short digression. Some war games are based on a particular historic event; *CINC PAC — Battle of Midway* is of this type, while chess is not. All war games involve some sort of map for a board. Most computer war games use a low resolution, semi-graphics screen. Again, the comparison with chess is very appropriate.

A small number, *CINC PAC* *Battle of Midway* among them, utilize the highest resolution available on the monitor to provide a map board which looks like it came from a geography text. Of course, the selection of that type of display implies dedication of much larger computer storage just for the map.

Most semi-graphics war games can be programmed in BASIC; *CINC PAC — Battle of Midway* uses BASIC for input/output and logic, but has machine language control over the displays. Another difference between semi-graphics and high resolution games is that the rules for navigation and movement can be more generalized in high resolution. The price paid for this is the user no longer simply moves pieces with arrow keys, but must think through the angular directions for his movement and/or his fire.

I have seen semi-graphics simulate the tactics of a particular time quite well, but I was unprepared for how well Steve Berry was able to model the subtleties of naval and air warfare at Midway and "shoe horn" it into the Color Computer's 32K. In this game, the user can feel as if he is there — the displays are that well done, and even the pauses between moves have the suspense of "being there."

As with most such games, I found the instructions too brief. There is no way to enter your initial game later than a week before the battle; the instructions suggest you use that time to practice fleet maneuvers. I did, but became a bit miffed because by the time I finished the practice and had gotten acceptably proficient I had destroyed almost a third of my fleet in operational accidents! The only thing to do was start over at the NEW GAME point and sit watching the idiot box while pressing SKIP MOVE until I had passed the wasted week. This process left my forces intact. After that, of course, I simply saved the new starting situation and worked from it.

One of the reasons I lost so many assets during set-up was a few bugs in the program. I phoned Ark Royal and requested software support. During most of the initial calls they were unaware that I was reviewing the game for RAINBOW, so I can state quite confidently that their software support to the general customer is excellent.

There seem to be two main problems. The lesser one is the lack of a logic trap to prevent overloading a platform (carrier or land base); if more than nine squadrons of planes are landed on any platform it is possible to hang the program and watch your fleet sail off to never-never land. Because the normally prudent player is unlikely to put more than five squadrons on any one platform, this bug is considered very minor.

However, a somewhat related bug is the ability to lose control over a carrier by attempting to exit the Launch Planes sequence improperly. To correct that, simply replace Line 1186 of the main program with:

```
1186 IF A$="S" THEN 1196 ELSE IF A$>"1" OR A$>"9"
     OR IF VAL (A$)>(UC+1) THEN 1185
```

This patch is simplicity itself on the disk version but somewhat inconvenient on the tape version. Ark Royal says if you get a copy that is *not* marked with a red 'X' you should contact them for replacement; copies with a red 'X' will already have had the patch inserted. With the patch inserted, you can abort Launch Planes by typing 'S' and the keyboard is locked against the non-numeric input which is known to cause the bug.

Another minor difference likely to occur in the patched version concerns Ship's Bells. In all versions, the program calls the Admiral (you) back from coffee breaks by sounding the time with Ship's Bells. Later versions sound somewhat more like "Victory at Sea" than earlier ones; the do-it-yourselfer can improve the early version by a higher pitched tone in lines 1357 and 1359 and by a smaller time delay in Line 1358. I mention this not as a bug, but as a small example of the mood-setting touches which make this game so stimulating.

There is another anomaly in the program; whether to call it a "bug" is a judgment call. Every U.S. unit on the screen (possibly even some enemy units) is a tiny interrupt-driven blinking light. Sometimes after combat, a downed unit will continue to blink; a ghost which can be battle debris. I considered it a radar sighting for my search units. The screen shows about 3,000 miles of Pacific Ocean, but combat sightings are at distances more like 100 miles. Accurate navigation improves your score. On the other hand, in one game I wasted three days' use of two B-17 squadrons chasing these ghosts. Personally, I feel the presence of the ghosts provided the touch of uncertainty needed in a combat game.

If you want a game more challenging than ones which rely totally on the arrow keys or the joystick, consider *CINC PAC — Battle of Midway*. It is good enough to call itself a battle Simulation, and it is fun enough to call itself a game.

(Ark Royal Games, P.O. Box 14806, Jacksonville, FL 32238; requires 32K, tape $27.95, disk $30.95)

Software Review

Telepatch — An Excellent Enhancer For *Telewriter*

By John R. Curl

When I first got *Telewriter-64* last year, I was in heaven. I couldn't believe the program's versatility and ease of use. I didn't have to study the manual and several other books for a month before I could start using it. I didn't have to remember complicated command strings. Most of my documents were less than six pages long. This was the word processor I had been looking for.

As time went on, though, I began making a mental wish list of things *Telewriter-64* didn't have that it needed. This list included key click, block move, key repeat, easier disk I/O and more. *Telewriter-64* was still the word processor for me; I just wished it had some little improvements.

Enter Spectrum Projects' *Telepatch* by Bob Van Der Poel. As the name implies, *Telepatch* is a *Telewriter-64* enhancer. It adds all of the above improvements and more. After using it for a week, I can't believe *Telewriter* never had the enhancements that *Telepatch* provides.

Telepatch requires a 64K Color Computer with at least one disk drive. You must also have the disk version of *Telewriter-64*. Even with all of the enhancements, the basic functions of *Telewriter-64* are still the same.

Installation of *Telepatch* is very simple. First, make a backup of your *Telewriter-64* and *Telepatch* diskettes. (Never use your originals!) Next, format a blank diskette (known as the destination diskette). Insert the *Telepatch* diskette and type RUN "PATCHER" and press ENTER. After this, all you have to do is follow the prompts — it's as simple as that.

When finished, your new systems diskette should have the following programs on it: *TW64/NEW, T/BAS, */*

*** and *TSPOOL/BAS*. These four programs handle everything including disk I/O. *TW64/BIN* is the new patched *TW64/BIN*. *T/BAS* replaces *U/BAS* as the boot program. */*** is the new disk I/O routine replacing *S/XXX*, *S/BIN* and *S/ASC*. I'll discuss *TSPOOL/BAS* later in the review.

To boot your patched *Telewriter-64*, just type `RUN "T"` and press ENTER. You will be greeted by a new title screen. It states that *Telewriter-64* is "modified, improved and made better in many ways" (it's the truth). You won't notice anything different on the main menu or the format menu. However, the Disk I/O menu has been totally replaced.

The new features added to the editor are: True Block Move, Visible Carriage Returns, Key Beep, Key Repeat, Lowercase, Faster Cursor Movement and Overstrike Mode.

The new boot program is the best improvement. With regular *Telewriter-64*, I had to set all of the print parameters in the format menu before I could print a document. This aggravated me very much! With the new boot program, you can set almost all of these parameters beforehand and not have to worry about them. This is made possible because the boot program now uses the POKE command to set the parameters.

You can also determine whether or not the new features are on or off. All you have to do to change a parameter or to turn a feature on is use BASIC's EDIT command to change different values in the boot program. The manual explains the procedure and also gives the different values that you need.

The format menu's printing parameters that can be set are: page numbering, spacing, queue function, printer type, one-page function, where to put the page number, characters per line, upper margin, bottom margin, left margin, lines per page and printer Baud rate. As you can see, almost all of the format features can be set to any default value needed.

The editor features which can be set in the boot program include: editor screen's background color (green or buff), key click on or off, key click length, key click tone, key-repeat rate, repeat delay, shift-repeat delay, lowercase/uppercase toggle and overstrike/insert toggle.

There are also several disk I/O parameters that can be changed in the boot program. These are: verify on or off, default drive number, binary or ASCII I/O default and drive stepping rate (for those who have drives which can run faster than 30 ms).

While I'm on the subject of disk I/O, I will explain the features of the new disk I/O driver. With the original *Telewriter-64* if you wanted to switch between binary and ASCII I/O, you had to exit the program and load either *S/BIN* or *S/ASC* and then save this as *S/XXX*. That's quite a hassle. With the new driver, */***, all you have to do to toggle between binary or ASCII I/O is press the 'I' key at the disk I/O menu. No more problems. When the "Kill" function is selected, you will be asked if you are sure you want to kill the file. You will also be asked if you are sure when you try to read in a file and there is already text in the buffer.

The program will also remember the last filename used. Thereafter, this will be the default filename for any I/O functions. If you would like to change the default filename, press the '@' key at the disk I/O menu. You will then be prompted for the new default filename. All other disk I/O funtions still operate the same.

I will now explain the *TSPOOL/BAS* program. In order to run this utility, you must exit to BASIC from the disk I/O menu, and type `RUN "TSPOOL"` and press ENTER. This utility program allows you to print a file to the disk instead of a printer. This means the file will be printed to disk in the exact same format as if you had printed it on paper: margins and spacing will be observed, centered text will be centered and pages will be numbered.

This enables you to print the file at a later time using a simple little BASIC program with the LINE INPUT command. You don't have to have *Telewriter-64* to print the file on a printer. Just OPEN the file on disk, LINE INPUT the text from the disk and then `PRINT#-2,` the text. You would have to actually use this feature to appreciate it.

The only limitations of this utility are that it will only work with files that have been saved in binary format and it will not work with files that have embedded 'Q' (queue) commands.

Telepatch is an excellent program. It adds several outstanding features to *Telewriter-64*. In my opinion, whether you already own *Telewriter-64* or are thinking of purchasing it, you must purchase *Telepatch* in order to get the most benefit from *Telewriter-64*.

(Spectrum Projects, Inc., P.O. Box 21272, 93-15 86th Drive, Woodhaven, NY 11421, $19.95 plus $3 S/H)

Book Review

CoCo Gains Power And Speed With *500 Pokes, Peeks 'N Execs*

This comprehensive book, *500 Pokes, Peeks 'N Execs* by Kishore M. Santwani, will provide you with hours of PEEKing, POKEing and EXECuting.

The soft cover book contains 68 pages, all chock-full of useful information for the CoCo user.

The use of the various commands listed will allow you to enhance the capability of your BASIC programs without having to resort to complicated machine language routines.

We are all familiar with the high-speed POKE and other often-used PEEKs and POKEs frequently found in the pages of RAINBOW. This book contains a wealth of others that will amaze and amuse you.

The book is well structured into various sections which deal with cassette, disk, printer, etc., and each command is followed by the expected result and an appropriate remark as to what it can be used to accomplish. Here in one easy-to-use source are all the various PEEKs and POKEs you will likely ever need. No more searching through scraps of paper and past issues of RAINBOW to find that "handy POKE I can't remember."

The book contains many useful commands and routines too numerous to list here, but suffice it to say that almost anything from *Auto-Start* to *Zaxxon* plays is covered. Other useful commands to provide key beeps and screen print are included as well as key disables, memory bank switching and recovering lost BASIC programs.

In summary, this is a unique book which contains information which took a great deal of research to collect.

Many of the commands listed have been, until now, relatively guarded by those "in the know." It will help unravel the many POKE, PEEK and EXEC commands of the CoCo to help you gain the power and speed of assembly language through BASIC. It will help you reach greater heights in successful programming and professionalize your work. I believe you will find *500 Pokes, Peeks 'N Execs* worth the price.

(Spectrum Projects, Inc., P.O. Box 21272, Woodhaven, NY 11421, $16.95 plus $3 S/H)

— Jerry Semones

Book Review

Practical Programming In Pascal

Makes Learning Pascal Easy

PASCAL is much more than a programming language. It was conceived by Niklaus Wirth as an academic tool to teach structured programming techniques. As such, most texts on the subject are written with the computer science major in mind, and are extremely academic and technical. For a novice programmer, it can be a strenuous if not insurmountable task to try to learn the language for practical purposes from such a text. Kent Porter's *Practical Programming In Pascal* bridges the gap.

Inside the flashy red, soft cover of this book (which will make it easy to find in the pile of documentation that always accompanies computer ownership) is a very well organized text. Also, due to Mr. Porter's smooth writing style and thorough knowledge of PASCAL, all 266 pages are incredibly easy to read and understand.

In the first four chapters you will be introduced to the history of PASCAL and the benefits of structured programming, then be taken from a description of the bare basics through the step-by-step development of a program using PASCAL's excellent looping and decision making abilities.

The development of the program from statement of purpose to final product is, of course, accompanied by a technical discussion of the parts of the program as they are constructed. As the book progresses, this process is repeated, developing more and more complex programs while readers learn more and more features of this powerful programming language. By the time you finish it, you will have a practical knowledge of PASCAL and have learned structured programming techniques which you can apply regardless of the language you are using.

This brings me to BASIC09. This book was written by Mr. Porter as "An Introduction to Computer Programming" and has much general programming information demonstrated in PASCAL. In that BASIC09 is an enhanced BASIC with added PASCAL-like features, they are very similar. If you are having some difficulty understanding some of the features of BASIC09 like parameter passing, defining data types and complex data structures, this book will help. It is a PASCAL study guide, however, and it should be studied along with the use of a PASCAL compiler.

In summary, if you are just beginning, this book will get you started in the right direction. Get yourself one of the available PASCAL compilers and use *Practical Programming In Pascal* as your study guide. You'll be writing PASCAL programs in no time.

If you already program in PASCAL at an intermediate level, the book will be useful for review purposes. Since it is structured as a study guide, it is not particularly suited for use as a reference manual, but it is fully indexed.

(New American Library, 1633 Broadway, New York, NY 10019, $14.95)

— James F. Taylor

continued from page 3..

The Best of CoCoOz#3 "Utilities", is now available. Price is the same as for #'s 1 and 2. This Tape should be of special interest to readers of Rainbow, who tend to use utilities more often. There are spoolers, disk directory handlers and all manner of beaut things on this Tape (or disk).

The Best of CoCoOz #1, 2, and 3 are now available on order from your Tandy shop.

Ron Wright's excellent Viatel program is now available from Paris Radio and soon from your Tandy store. You'll need a 1200/75 baud modem. Tandy will have one in late November / early December.

CoCoBug Magazine is to cease production. We are going to honour any current subscriptions with a commensurate sub to Australian CoCo. Barry Cawley, CoCoBug's editior now joins our team, and will head up the "softgold" magazine.

Finally, you may have noticed the increased size this issue.

From the recent survey, it seems the demand for more information is ever present. So we are expanding the size of Rainbow first to 72 pages, and if that proves successful, then we'll go on to 80 pages.

This is a magazine for experienced users and is meant to challenge you. If you find Rainbow heavy going, you may like to switch to Australian CoCo.

Once more I find from the surveys, I must also remind users of Tandy MC-10's, T1000's, T2000's, T100's & T200's that the magazine we produce which is of relevence to you, is again Aussi CoCo.

EDUCATION PAGE

With this edition, this column goes into recess till next year.

I want to thank all those who have contributed to it during the year. Especially to Bob Horne, I say thank you sir for the fine brace of programs you supplied.

The work of all the authors of this column has been appreciated very much. The readers' letters and surveys confirm it.

And what a way to go out!

Tandy has dropped the price of the CoCo to an amazing low, thus giving schools the absolute opportunity to get those little fingers on a keyboard.

This is what computing in schools should really be about. It's all very well to have expensive units in schools, but if the kids don't get to use them, it's not going to do them much good is it!

If that were not enough, then the news of Austsat and the changes about to happen for isolated children is really something to get excited about.

Imagine sitting at your kitchen table somewhere west of Port Hedland in WA and being able to submit your work through your computer to your teacher, whom you can see on your TV, and who will react to your inputs immediately!

Well this is not a thing for imagining any longer, it is about to happen, and I am very pleased to say that we, along with Tandy, are to be a part of it.

The system offers all sorts of possibilities. We will be discussing and expanding upon them in future issues.

Next year, we won't have that rich reserve of programs we had from Bob Horne this past year. (Unless he sends some more!)

So we have two needs.

The first and most urgent, is for educational programs for use in Australian CoCo in February - the "Back to School" issue.

The second need is for quality programs, hopefully road tested, for inclusion in Australian Rainbow.

1986 looks like being a bumper year for Tandy and for those who use Tandy computers - the prices will be the best ever, Tandy has finally organised their own excellent customer support unit, our magazines are stronger that ever, and the new software, which just keeps streaming in, is SO good! And to cap it all, the new satellite system promises some very exciting possibilities for the future.

whatzit

by Randall Smith and Bob Horne

Randall Smith created a game called "Whatzit" for July 1983's Rainbow.

Bob Horne took this game and made some changes to it, and has since used it in the classroom to assist in the teaching of spelling.

THE LISTING:

16K ECB

```
1 '***WHATZIT5 BY RANDALL SMITH*
2 GOTO10
3 SAVE"WHATZIT5:2":DIR2:STOP
10 'RAINBOW JULY 1983 PAGE24
15 'ADAPTED BY BOB HORNE, IPSWIC
H
20 CLS:SOUND200,2:PCLEAR4:PMODE3
,1:PCLSRND(4):SCREEN1,RND(2)-1:C
LEAR550:DIMI(515),X$(515),CH$(36
),U(10),B(12),SC$(2),W(15),W$(15
):T=RND(-TIMER)
30 FORX=0TO35:READCH$(X):NEXT:FO
RX=1TO515:READX$(X):NEXT
40 CLS:SOUND200,2:FORX=1024TO105
5:POKEX,134:NEXT:FORX=1056TO1472
STEP32:POKEX,134:POKEX+31,134:NE
XT:FORX=1504TO1535:POKEX,134:NEX
T
50 PRINT@68,"WHICH UNIT DO YOU W
ISH TO";:PRINT@100,"PRACTISE (TY
PE A NUMBER";:PRINT@132,"FROM 1
TO 40.)";:INPUTXX:PRINT@159,CHR$
(134);
60 IFXX<1 OR XX>40 THEN40
70 FORX=1TOXX:READW1,W2:NEXTX
80 FORX=2TO6:PRINT@X*32+2,STRING
$(27,32);:NEXTX
90 PRINT@225,"DO YOU WANT INSTRU
CTIONS(Y/N)";
100 PRINT@170,"w h a t z i t";
110 IN$=INKEY$
120 IN$=INKEY$:IFIN$="Y"THENGOSU
B930 ELSEIFIN$=""THEN120
130 PRINT@225,"  HOW MANY PLAYER
S (1 OR 2)?  ";
140 IN$=INKEY$
150 PL$=INKEY$:IFPL$<>"1"ANDPL$<
>"2" THEN150 ELSEPL=VAL(PL$):SOU
ND200,1
160 PCLS:DRAW"BM0,12;C8;S8"+CH$(
30)+CH$(25):GET(0,0)-(28,14),U,G
:GET(128,96)-(156,110),B,G
170 PCLS:DRAW"BM83,12S8C7"+CH$(3
2)+CH$(17)+CH$(10)
180 DRAWCH$(29)+CH$(35)+CH$(18)+
CH$(29)
190 DRAW"BM0,30"+CH$(25)+CH$(21)
+CH$(10)
200 DRAWCH$(34)+CH$(14)+CH$(27)+
CH$(1)
210 DRAW"BM0,48"+CH$(28)+CH$(12)
+CH$(24)+CH$(27)+CH$(14):COLOR8,
5:LINE(69,33)-(126,51),PSET,B
220 IFPL=1 THEN250 ELSECOLOR7,5:
DRAW"BM163,30"+CH$(25)+CH$(21)+C
H$(10)
230 DRAWCH$(34)+CH$(14)+CH$(27)+
CH$(2)
240 DRAW"BM191,48"+CH$(28)+CH$(1
2)+CH$(24)+CH$(27)+CH$(14):COLOR
8,5:LINE(128,33)-(185,51),PSET,B
250 LINE(42,171)-(210,191),PSET,
B
260 IFPL=2 THENLP=2:PL=PL-1
270 PUT(0,52)-(28,66),U,PSET:SCR
EEN1,1
280 FORWO=1TO W1
290 CW=RND(W1)+W2:IF I(CW)=1 THE
N290 ELSE I(CW)=1
300 W$=X$(CW)
310 L=LEN(W$)
320 A$=LEFT$(W$,INT(L/3+.5)):B$=
MID$(W$,INT(L/3+1.5),INT(L-2*(IN
T(L/3+.5)))):C$=RIGHT$(W$,INT(L/
3+.5))
330 J$="":X=RND(5):ON X GOTO340,
350,360,370,380
340 J$=C$+A$+B$:GOTO390
350 J$=C$+B$+A$:GOTO390
360 J$=B$+C$+A$:GOTO390
370 J$=B$+A$+C$:GOTO390
380 J$=A$+C$+B$:GOTO390
390 DRAW"BM65,187C6S8"+CH$(17)+C
H$(18)+CH$(29)
400 DRAW"BM+10,+0"+CH$(14)+CH$(2
3)+CH$(29)+CH$(14)+CH$(27)
410 IN$=INKEY$
420 IFINKEY$<>CHR$(13) THEN420 E
LSEPAINT(128,175),5,8
430 L=LEN(J$):M=INT(132-(21*(LEN
(J$)/2))-21)
440 FORX=1TO L:M=M+21:FORS=1TO10
STEP3
450 DRAW"BM"+STR$(M)+",100C6S"+S
TR$(S)+CH$(ASC(MID$(J$,X,1))-55)
460 PLAY"V31T230L23003A":DRAW"BM
"+STR$(M)+",100C5S"+STR$(S)+CH$(
ASC(MID$(J$,X,1))-55)
470 NEXTS:DRAW"BM"+STR$(M)+",100
C6S12"+CH$(ASC(MID$(J$,X,1))-55)
:NEXTX
480 T=1700+50*L
490 GOSUB640
500 IN$=INKEY$
510 T=T-1:IFT=0 THEN T1=1:GOSUB5
30:GOTO390
520 GOTO660
530 PLAY"T3L3V3101BL2A"
540 PAINT(128,175),5,8
550 IFT>0 THENGOSUB640:RETURN
560 LINE(0,79)-(255,100),PRESET,
BF
570 M=INT(132-(21*(LEN(J$)/2))-2
1)
580 DRAW"S8BM40,90"+CH$(29)+CH$(
17)+CH$(14)+"BR10"+CH$(32)+CH$(2
4)+CH$(27)+CH$(13)+"BR10"+CH$(32
)+CH$(10)+CH$(28)
590 FORX=1TO LEN(W$):M=M+21
600 DRAW"BM"+STR$(M)+",115C8S12"
+CH$(ASC(MID$(W$,X,2))-55):NEXTX
610 DRAW"C7S8BM60,135"+CH$(29)+C
H$(27)+CH$(34):DRAW"BR10"+CH$(10
)+CH$(16)+CH$(10)+CH$(18)+CH$(23
)
620 FORX=1TO1000+30*L:NEXT:LINE(
0,69)-(255,135),PRESET,BF
630 RETURN
640 WW$="":M=INT(130-(14*LEN(W$)
/2))
650 DRAW"BM"+STR$(M)+",187C6S8":
RETURN
660 IN$=INKEY$:IFIN$=""THEN510
670 IFASC(IN$)=8 THENWW$="":GOTO
690 ELSEIFASC(IN$)=13 THEN700 EL
SEIFASC(IN$)<65 ORASC(IN$)>90 TH
EN510
680 IFLEN(WW$)=LEN(W$) THEN510 E
LSEDRAWCH$(ASC(IN$)-55):WW$=WW$+
IN$:GOTO510
690 PAINT(128,175),5,8:GOSUB640:
GOTO510
700 IFWW$=W$ THENPLAY"V31L10T100
03ABCDEFG" ELSEGOSUB530:IFT>0 TH
EN510 ELSE750
710 DRAW"S12C8BM80,125"+CH$(27)+
CH$(18)+CH$(16)+CH$(17)+CH$(29)
720 IF T1=1 THEN T1=0:GOTO740
730 SC$(PL)=STR$((INT((LEN(W$)*5
)+T/6.86))+VAL(SC$(PL))):GOSUB86
0
740 FORX=1TO500:NEXT:LINE(0,79)-
(255,125),PRESET,BF:PAINT(128,17
5),5,8
750 NEXTWO:FORX=W2 TO W2+W1:I(X)
=0:NEXTX:IFLP=2 THENPL=PL+1:LP=0
:PUT(0,52)-(28,66),B:PUT(227,52)
-(255,66),U,PSET:PLAY"T3L303V31G
L302G":GOTO280
760 IFPL=2 THENPUT(227,52)-(255,
66),B ELSEPUT(0,52)-(28,66),B
770 DRAW"BM13,100C7S16"+CH$(16)+
CH$(10)+CH$(22)+CH$(14)
780 DRAW"BM145,100"+CH$(24)+CH$(
31)+CH$(14)+CH$(27)
790 DRAW"BM65,187C6S8"+CH$(17)+C
H$(18)+CH$(29)
800 DRAW"BM+10,+0"+CH$(14)+CH$(2
3)+CH$(29)+CH$(14)+CH$(27)
810 IFVAL(SC$(1))>VAL(SC$(2)) TH
ENDRAW"BM107,25" ELSEDRAW"BM149,
25"
820 DRAW"S4C8NU5NE5NR5NF5ND5NG5N
L5NH5":FORX=1TO100:NEXT
830 DRAW"C5NU5NE5NR5NF5ND5NG5NL5
NH5":FORX=1TO100:NEXT
840 IFINKEY$<>CHR$(13) THEN820
850 SC$(1)="0":SC$(2)="0":SCREEN
0,0:GOTO130
860 L=LEN(SC$(PL))-1:ON PL GOTO8
70,880
870 PAINT(98,36),5,8:M=INT(101-(
14*L/2)-14):GOTO890
880 PAINT(158,36),5,8:M=INT(160-
(14*L/2)-14)
890 FORSC=2 TOL+1:M=M+14:DRAW"BM
"+STR$(M)+",48C6S8"
900 DRAWCH$(ASC(MID$(SC$(PL),SC,
1))-48)
910 NEXTSC
920 RETURN
930 CLS:PRINT"IN whatzit A SCRAM
BLED GROUP OF LETTERS WILL APPEA
R.YOU HAVE    ABOUT 30 SECONDS T
O ENTER THE   RIGHT ANSWER."
940 PRINT"IF YOU MAKE AN ERROR P
RESS THE  LEFT ARROW KEY AND STA
RT OVER.  IF YOU ENTER THE WRONG
 ANSWER   YOU MAY RETYPE IT IF T
HERE IS   TIME.":PRINT:PRINT"PRE
SS ANY KEY TO CONTINUE."
950 IFINKEY$=""THEN950
960 CLS:PRINT" EACH PLAYER GETS
10 WORDS.THE  SCORE IS BASED ON
THE TIME TAKENAND THE LENGTH OF
THE WORD.*******NOTE:YOU MUST PR
ESS <ENTER>   WHEN YOU ARE FINIS
HED TYPING THEWORD."
970 PRINT:PRINT"PRESS ANY KEY TO
 START THE GAME."
980 IFINKEY$="" THEN980 ELSE CLS
RND(8):GOTO130
990 DATA"BU1U4E1R2F1D4G1L2H1BD1;
M+4,-6;BD6BR3"
1000 DATA"BU5BR1E1D6NL1R1BM+4,+0
"

1010 DATA"NR4U1BU3U1E1R2F1D1G1L1
G2BM+7,+1"
1020 DATA"BU5E1R2F1D1G1NL2F1D1G1
L2NH1BM+7,+0"
1030 DATA"BU3NR4M+3,-3;D6BM+4,+0
"

1040 DATA"BU4NR3U2R4BD2BL1F1D2G1
L2H1BM+7,+1"
1050 DATA"BU6BR3NF1L2G1D2NR3D2F1
```

```
R2E1U1H1BM+4,+3"
1060 DATA"U1BU5R4D1M-4,+4BM+7,+1
"
1070 DATA"BU1U1E1R2E1U1H1L2G1D1F
1R2F1D1G1L2H1;BM+7,+1"
1080 DATA"BU1F1R2E1U4H1L2G1D1F1R
3;BM+3,+3"
1090 DATA"U5E1R2F1D3NL4D2BR3"
1100 DATA"U6R3F1D1G1NL3F1D1G1L3B
R7"
1110 DATA"BU1U4E1R2F1BD4G1L2H1BR
7BD1"
1120 DATA"R1U6NL1R2F1D4G1L2BR6"
1130 DATA"U3NR3U3R4BD6L4BR7"
1140 DATA"U3NR3U3R4BD6BR3"
1150 DATA"BU1U4E1R3BD4NL1D2L3H1B
R7BD1"
1160 DATA"U3NR4U3BR4D6BR3"
1170 DATA"BU6BR1R1ND6R1BR1BD6BL1
L2BR6"
1180 DATA"BU2D1F1R2E1U5BD6BR3"
1190 DATA"U6BR4;M-4,+3;M+4,+3;BR
3"
1200 DATA"NU6R4BR3"
1210 DATA"U6;M+2,+3;M+2,-3;D6BR3
"
1220 DATA"U6;M+4,+6;U6BD6BR3"
1230 DATA"BR1HU4ER2FD4GL2BR6"
1240 DATA"U6R3F1D1G1L3D3BR7"
1250 DATA"BU1U4E1R2F1D4G1L2H1BE1
BR1F2BR3"
1260 DATA"U6R3F1D1G1L3R1;M+3,+3;
BR3"
1270 DATA"BU1F1R2E1U1BU3H1L2G1D1
;M+4,+2;BR3BD2"
1280 DATA"BU6R2ND6R2BD6BR3"
1290 DATA"BU1U5BR4D5G1L2H1BD1BR7
"
1300 DATA"BU6;M+2,+6;M+2,-6;BD6B
R3"
1310 DATA"U6BR4D6;M-2,-2;NU1;M-2
,+2;BR7"
1320 DATA"U1;M+4,-4;U1BL4D1;M+4,
+4;D1BR3"
1330 DATA"BR2U3M-2,-2U1BR4D1M-2,
+2BD3BR5"
1340 DATA"BU6R4M-4,+6R4BR3"
1360 REM***YEAR 5 LISTS*********
1370 REM***UNIT 1 * 13 WORDS***
1380 DATABAMBOO,COCKATOO,WOOLLEN
,BOOMERANG,SALOON,KANGAROO,BALLO
ON,SMOOTH,JACKAROO,LAGOON,SCHOON
ER,CARTOON,OOZING
1390 REM***UNIT 2 * 14 WORDS***
1400 DATAJOURNEY,VALLEY,JOCKEY,C
HIMNEY,QUAY,ABBEY,TROLLEY,BUOY,D
ONKEY,PULLEY,KIDNEY,MONKEY,GALLE
Y,STOREY
1410 REM***UNIT 3 * 13 WORDS***
1420 DATAAGREEABLE,EXCEED,REFERE
E,FREEDOM,CANTEEN,SQUEEZE,SUCCEE
D,GREETINGS,BEEHIVE,CHEERFUL,NEE
DLEWORK,DEGREE,EERIE
1430 REM***UNIT 4 * 13 WORDS
1440 DATAACCEPT,AFFORD,ABRIDGE,A
CCOUNT,ALLOW,ACCUSE,ANNUAL,ACCID
ENT,ANNOY,ADDRESS,ANNOUNCE,ADDIT
ION,APPEAR
1450 REM***UNIT 5 * 14 WORDS
1460 DATANOISY,REJOICE,DESTROY,N
OISLY,APPOINT,ENJOYMENT,CHOICE,D
ISAPPOINT,AVOID,EMPLOY,LOITER,AN
NOY,TOILET,MOISTURE
1470 REM***UNIT 6 * 14 WORDS
1480 DATAACTOR,HOST,COUNT,HEIR,W
AITER,EMPEROR,DUKE,NEGRO,GOD,MAY
OR,AUTHOR,SHEPHERD,CONDUCTOR,TIG
ER
1490 REM***UNIT 7 * 13 WORDS
1500 DATAFORTUNE,HISTORY,FORGIVE
,FORTNIGHT,GLORIOUS,PORTION,FORC
E,IMPORTANT,BORING,MEMORY,ORCHAR
D,VICTORY,RESTORE
1510 REM***UNIT 8 * 10 WORDS
1520 DATATROOP,SQUAD,BALLET,CREW
,CHOIR,ARMY,FAMILY,JURY,CROWD,ST
AFF
1530 REM***UNIT 9 * 15 WORDS
1540 DATAQUAINT,ENTERTAIN,GAILY,
FAINTLY,FAITHFUL,DAILY,PAINFUL,W
AIST,RAILWAY,RAINBOW,PRAISE,FAIL
URE,DAINTY,WAITING,STRAIGHTEN
1550 REM***UNIT 10 * 9 WORDS
1560 DATAOBJECT,INJECT,COLLECT,C
ORRECT,SELECT,INSPECT,CONNECT,DI
RECT,PROTECT
1570 REM***UNIT 11 * 14 WORDS
1580 DATASPARKLING,ARMY,COLLAR,P
ARDON,GARMENT,BEGGAR,REMARK,FARC
E,CIGAR,ORCHARD,MARVEL,SUGAR,DEP
ART,HEART
1590 REM***UNIT 12 * 12 WORDS
1600 DATABUSY,UNTIDY,READY,WARY,
CLUMSY,STEALTHY,WEARY,HEAVY,STEA
DY,DROWSY,HUNGRY,SPEEDY
1610 REM***UNIT 13 * 19 WORDS
1620 DATAREADY,HEALTHY,TREASURE,
FEATHER,HEAVEN,BREAST,STEADY,WEA
LTHY,MEASURE,WEATHER,HEAVY,INSTE
AD,STEALTHY,LEATHER,HEAVILY,DREA
DFUL,BREADTH,MEADOW,HOMESTEAD
1630 REM***UNIT 14 * 7 WORDS
1640 DATASTUDY,CARRY,HURRY,COPY,
REPLY,BURY,WORRY
1650 REM***UNIT 15 * 14 WORDS
1660 DATANERVOUS,CERTAIN,GROCERI
ES,PERFECT,PERMIT,SERMON,EXERCIS
E,SERVANT,DIFFERENT,SHEPHERD,SER
VICE,SEVERAL,HANDKERCHIEF,NURSER
Y
1670 REM***UNIT 16 * 12 WORDS
1680 DATACAREFUL,HOPEFUL,TRUTHFU
L,BEAUTIFUL,USUAL,SKILFUL,JOYFUL
,CHEERFUL,PAINFUL,MOURNFUL,GRACE
FUL,FAITHFUL
1690 REM***UNIT 17 * 14 WORDS
1700 DATAFOUNTAIN,POUNCE,AROUSE,
LOUNGE,FLOUNDER,MOUNTAIN,ANNOUNC
E,TROUSERS,ALOUD,ROUNDERS,ACCOUN
T,PRONOUNCE,DOUBT,BOUNDARY
1710 REM***UNIT 18 * 8 WORDS
1720 DATASHOW,FLOW,SNOW,ROW,FALL
OW,OVERFLOW,NARROW,OWE
1730 REM***UNIT 19 * 11 WORDS
1740 DATACAUSE,AUTHOR,HAUL,BECAU
SE,AUTUMN,GAUDY,PAUSE,TAUT,SAUCE
,APPLAUSE,CAUTIOUS
1750 REM***UNIT 20 * 6 WORDS
1760 DATASMUGGLE,BELIEVE,MANAGE,
ARRANGE,SETTLE,PRODUCE
1770 REM***UNIT 21 * 17 WORDS
1780 DATASPIRE,POLITE,OFFICE,SUR
PRISE,DIVIDE,DESCRIBE,SHIRE,INVI
TE,ADVICE,REVISE,COLLIDE,ENTIRE,
CHASTISE,TIRED,ADVISE,PRIZE,CAPS
IZE
1790 REM***UNIT 22 * 12 WORDS
1800 DATATAX,STITCH,OSTRICH,DUCH
ESS,SPEECH,ADDRESS,WALTZ,PRINCES
S,CHORUS,BROOCH,SANDWICH,WATCH
1810 REM***UNIT 23 * 13 WORDS
1820 DATABURDEN,DISTURB,FURTHER,
ABSURD,FURNISH,GURGLE,SATURDAY,N
URSERY,PURPOSE,MURDER,CURTAIN,UR
GE,URGENT
1830 REM***UNIT 24 * 12 WORDS
1840 DATACANARY,DAIRY,FACTORY,CO
MEDY,LIBRARY,VICTORY,COUNTRY,HIS
TORY,MEMORY,TRAGEDY,INJURY,DIARY
1850 REM***UNIT 25 * 13 WORDS
1860 DATAROUGH,DOUBLE,COURAGE,CO
UPLE,FLOURISH,TOUGH,TROUBLE,ENCO
URAGE,TOUCH,NOURISH,ENOUGH,COUNT
RY,COUSIN
1870 REM***UNIT 26 * 7 WORDS
1880 DATATRAVEL,QUARREL,MODEL,CA
NCEL,LABEL,EQUAL,PROPEL
1890 REM***UNIT 27 * 12 WORDS
1900 DATAQUESTION,QUARREL,QUARTE
R,EQUATOR,SQUATTER,LIQUID,INQUIR
E,SQUEAL,SQUARE,SQUASH,QUAINT,QU
IET
1910 REM***UNIT 28 * 12 WORDS
1920 DATATIMID,HURRIED,EXCITED,S
UDDEN,DISTINCT,INSTANT,QUIET,HON
EST,SECRET,PLEASANT,VALIANT,PATI
ENT
1930 REM***UNIT 29 * 17 WORDS
1940 DATAPIECE,CHIEF,FIERCE,BELI
EVE,FIELD,RECEIVE,NIECE,CHIEFTAI
```

```
N,PIERCE,SHRIEK,SHIELD,CEILING,T
HIEF,YIELD,GRIEF,HANDKERCHIEF,BR
IEF
1950 REM***UNIT 30 * 12 WORDS
1960 DATATOMATO,ECHO,NEGRO,BANJO
,MOSQUITO,CARGO,DINGO,VOLCANO,BU
FFALO,POTATO,HERO,MOTTO
1970 REM***UNIT 31 * 16 WORDS
1980 DATAVASE,REFUSE,CLOSE,EXERC
ISE,RESULT,PAUSE,AMUSE,ENCLOSE,S
URPRISE,DESIGN,PRAISE,ACCUSE,PRO
POSE,PHRASE,ABUSE,CONFUSE
1990 REM***UNIT 32 * 6 WORDS
2000 DATAVISIT,CREATE,ENJOY,EMPL
OY,INVENT,ATTRACT
2010 REM***UNIT 33 * 9 WORDS
2020 DATAGUIDE,GUILTY,GUESS,GUIN
EA,GUARD,GUITAR,GUEST,GUILLOTINE
,DISGUISE
2030 REM***UNIT 34 * 12 WORDS
2040 DATAPHOTO,RADIO,SOLO,ESKIMO
,CAMEO,GROTTO,PIANO,BANJO,MERINO
,STUDIO,FOLIO,MEMENTO
2050 REM***UNIT 35 * 17 WORDS
2060 DATAHINGE,ENGINE,GINGER,FRI
NGE,TRAGIC,GIANT,PLUNGE,MAGIC,GE
NIUS,LOUNGE,ENRAGE,GENERAL,SPONG
E,AVERAGE,GIRAFFE,ANGEL,PIGEON
2070 REM***UNIT 36 * 15 WORDS
2080 DATAFISHERMAN,POSTMAN,CLERG
YMAN,TRADESMAN,FOREMAN,MILKMAN,P
OLICEMAN,GENTLEMAN,PLOUGHMAN,CRA
FTSMAN,MONSTER,YOUNGSTER,SPINSTE
R,GANGSTER,TRICKSTER
2090 REM***UNIT 37 * 20 WQRDS
2100 DATATOMB,GNOME,HONEST,KNIGH
T,ISLAND,WHISTLE,WRAPPER,BOMBER,
HONOUR,KNAVE,ISLE,GLISTEN,WRIGGL
E,PLUMBER,GHOST,KNAPSACK,HASTEN,
WRETCHED,WREATH,WRESTLE
2110 REM***UNIT 38 * 15 WORDS
2120 DATASKILFUL,WELFARE,ENROL,B
ELFRY,ALREADY,UNTIL,EXCEL,BEAUTI
FUL,WELCOME,ALSO,PROPEL,ALMOST,E
XPEL,ALTHOUGH,ALTOGETHER
2130 REM***UNIT 39 * 21 WORDS
2140 DATADESERT,PROPERTY,ENERGY,
MYSTERY,VERANDAH,OBSERVE,GOVERN,
DANGER,DESSERT,LIBERTY,ENERGETIC
,NURSERY,INTERESTING,OBSERVANT,C
ONCERT,DANGEROUS,EXPERT,REVERSE,
DESERVE,PATTERN,LANTERN
2150 REM***UNIT 40 * 12 WORDS
2160 DATAPREPARE,CREATE,BELIEVE,
BREATHE,ADMIRE,PAUSE,BATHE,CLOTH
E,NOTICE,IMPROVE,INCREASE,ESCAPE
2170 DATA13,0,14,13,13,27,13,40,
14,53,14,67,13,81,10,94,15,104,9
,119,14,128,12,142,19,154,7,173,
14,180,12,194,14,206,8,220,11,22
8,6,239,17,245,12,262,13,274,12,
287,13,299,7,312,12,319,12,331,1
7,343,12,360,16,372,6,388,9,394,
12,403,17,415,15,432
2180 DATA20,447,15,467,21,482,12
,503
```

EDUCATION NOTES

16K ECB

Learning The Art Of Written Expression With Proper Punctuation

By Steve Blyn

"Holy cow, Batman! Watch out for the Penguin!" exclaimed Robin.

There are quite a few punctuation marks in the preceding paragraph. Using punctuation marks correctly was almost becoming a lost art until recently. The "back to basics" movement occurring in many school systems has included stressing the teaching of written expression. The proper use of punctuation marks is an essential part of this skill.

Punctuation Quiz reviews the uses of punctuation marks. A chart is displayed listing major punctuation marks and shows their keyboard symbols. The student is then quizzed on when to use each of the marks illustrated on the chart. The student's task is to type in the answer to the computer's questions.

Rather than having the student insert the actual mark as the answer, we felt it proper, in this case, to require the correct spelling of names of the punctuation marks. This is always a difficult decision to make when writing any educational program. It presents a double-edged sword: On one hand, we always prefer to teach correct spelling; on the other hand, a child often feels cheated if he knows the correct answer but misspells it and gets marked as wrong.

We compromised on the issue this time. Exact spelling is the only way to get a correct answer in this program. The pressure is eased, however, by omitting any scoring in the program. While we do want the children to learn to spell words correctly, we do not want, at the same time, to penalize them too severely for minor spelling mistakes. After each group of five questions, the student may press 'C' to continue or 'E' to end the program. The program may be played continuously until the student learns all of the marks in the program.

Punctuation Quiz also helps to familiarize children with the keyboard/typewriter symbols. Punctuation marks are often confusing to beginners because the SHIFT key must be used to produce some of them. Fortunately, the CoCo's keyboard is very similar to that of a standard typewriter. More than half of the punctuation marks are in similar positions. Many other computers have keyboards that are quite different from a standard typewriter keyboard.

Beginners should be encouraged to gain familiarity with using all of the CoCo's keyboard symbols. The only odd punctuation marks we have are the brackets. The left bracket is made by simultaneously pressing the SHIFT and down-arrow keys and the right bracket is made by pressing the SHIFT and right-arrow keys. (I have trouble remembering this myself and always have to check it out first.)

Line 50 reads the DATA statements necessary for the chart. Line 60 reads the DATA statements for the questions and answers. Lines 70-100 print out the chart of quotation marks and their symbols.

Lines 190-320 contain the quiz. Line 210 randomly selects one of the 11 possible question and answer sets. Line 220 prints the chosen question. Lines 260-270 check to see whether the answer is correct. If it is incorrect, the correct answer will be displayed.

After five questions, the student may begin again or end the program by pressing either the letter 'C' or 'E'. The routine on lines 320-340 takes care of this task. A record of the student's score could be added between lines 320-330 if you desire one. This can easily be achieved by incrementing a counter on Line 260. To achieve this, add CR=CR+1 at the end of Line 260.

Each time a question is answered correctly, the counter will increase by one. You could then create a line such as 325 PRINT CR:CR=0. This would indicate the score and reset the count to zero for the next round.

As discussed earlier, we did not feel the need for a scorecard in this particular program. The choice, of course, is yours. As always, please feel free to alter this program as you see fit to help your children and students benefit the most from it.

```
180 ......194
380 .......48
END ......93
```

The listing: PUNCQUIZ

```
1Ø REM"PUNCTUATION QUIZ"
2Ø REM"STEVE BLYN,COMPUTER ISLAN
D,NY,1985
3Ø DIM A$(11),B$(11),C$(11),D$(1
1)
4Ø CLS
5Ø FOR I=1 TO 11:READ A$(I),B$(I
):NEXT I
6Ø FOR I=1 TO 11: READ C$(I),D$(
I):NEXT I
7Ø PRINT"         PUNCTUATION  MARK
S"
8Ø PRINT"  MARK           NAME"
9Ø PRINT"  ----           --------"
1ØØ FOR A=1 TO 11:PRINT A TAB(2)
A$(A) TAB(13)B$(A): NEXT A
11Ø REM"CHOOSE A DECORATIVE PATT
ERN"
12Ø NN=128+RND(127)
13Ø PR$=STRING$(32,NN)
14Ø PRINTPR$;
15Ø PRINT@484,"PRESS <ENTER> TO
GO ON";
16Ø EN$=INKEY$
17Ø IF EN$=CHR$(13) THEN 18Ø ELS
E 16Ø
18Ø FOR T= 1 TO 5
19Ø CLS
2ØØ PRINT"        PUNCTUATION   QU
IZ":PRINT PR$;
21Ø R=RND(11)
22Ø PRINT@96,C$(R)
23Ø PRINT@192," ":PRINT@192,"ANS
WER";
24Ø INPUT E$
25Ø PRINT@288,PR$;
26Ø IF E$=D$(R) THEN SOUND 2ØØ,2
:PRINT@ 364,"CORRECT"
27Ø IF E$<>D$(R) THEN SOUND 2Ø,3
:PRINT@362,D$(R):PRINT"IS THE CO
RRECT ANSWER THIS TIME."
28Ø PRINT@ 484,"PRESS <ENTER> TO
 GO ON";
29Ø EN$=INKEY$
3ØØ IF EN$=CHR$(13) THEN CLS ELS
E 29Ø
31Ø NEXT T
32Ø CLS RND(8):PRINT@96,"     PRE
SS <C> TO CONTINUE          OR <E
> TO END THE PROGRAM."
33Ø EN$=INKEY$
34Ø IF EN$="C" THEN RUN ELSE IF
EN$="E" THEN END ELSE 33Ø
35Ø DATA ",",COMMA
36Ø DATA .,PERIOD,?,QUESTION MAR
K,!,EXCLAMATION POINT,;,SEMICOLO
N,":",COLON,-,HYPHEN
37Ø DATA (  ),PARENTHESES,[  ],B
RACKETS,'' '',QUOTATION MARKS,',
APOSTROPHE
38Ø DATA THIS COMES AT THE END O
F A      DECLARATIVE SENTENCE.,P
ERIOD
39Ø DATA THIS IS USED TO SHOW ST
RONG     FEELINGS OR EMOTIONS.,E
XCLAMATION POINT
4ØØ DATA  THIS IS USED IN PLACES
 WHERE YOUWOULD PAUSE IN SPEAKIN
G.,COMMA
41Ø DATA THIS IS USED BETWEEN TW
O MAIN   CLAUSES NOT JOINED BY A
         CONJUNCTION.,SEMICOLON
42Ø DATA THIS IS USED BEFORE A L
IST OF   ITEMS.,COLON
43Ø DATA THIS IS USED TO SHOW PO
SSESSION OR OWNERSHIP.,APOSTROPH
E
44Ø DATA THIS IS USED TO DIVIDE
A WORD   INTO PARTS AT THE END O
F A LINE.,HYPHEN
45Ø DATA THESE ARE USED TO ENCLO
SE AN    EXTRA WORD OR WORDS.,PA
RENTHESES
46Ø DATA THESE ARE USED TO ENCLO
SE ITEMS FOUND WITHIN PARENTHESE
S.,BRACKETS
47Ø DATA THESE ARE USED TO SHOW
SOMEONE'SEXACT WORDS.,QUOTATION
MARKS
48Ø DATA THIS IS USED AT THE END
 OF AN   INTERROGATORY SENTENCE.
,QUESTION MARK
```

EDUCATION OVERVIEW

Computers Can Play An Important Role In Art Education

By Michael Plog, Ph.D.

When dealing with computers and education, the subject matter area getting the short end of the electronic stick is the arts. This is understandable in some ways, but really doesn't make much sense — for the arts or for education.

In general, elementary schools use computers most frequently with math, followed by language arts, reading, computer programming, social studies, science and then art. (This is a finding from the National Commission for Employment Policy, "Uses of Computers in Education," conducted by Education Turnkey Systems, Inc.) We can easily understand why computers are most frequently used with mathematics. After all, computers are mathematical calculating machines, and math teachers have been trained in such things as calculating machines. Art teachers are creative people who deal with such things as paint, clay or other products.

Of course, the concept of "art" covers a wide range of activities. There are the performing arts, such as dance, theater, mime, magic, music and so on. Commercial art covers such things as photography and drafting. Some schools even include film making in the curriculum.

The amazing thing is that in most fields of art the computer has been used by professional artists, but not in the schools, where students are usually introduced to the arts. Much of the past "computer art" depended on the ability of the computer to produce random numbers. Then, a song or painting or some other artistic product was created using the random numbers.

The random effect has been used in

the past with some degree of success. Mozart (in 1777) composed a piece of music by using dice to help pick a scale. One hundred and eighty years later (1957), Lejaren Hiller and Leonard Isaacson composed the "Iliac Suite for String Quartet" using random numbers generated by a computer. Generally, however, the random number approach to art results in a product that is appreciated only by the artist (if that much).

There is much more to computer-assisted art (and art education) than random numbers. We listen to a lot of computer music. Moog synthesizer and electronic guitars are commonplace today. The computer provides not only the music, but in some cases, provides the structure of a piece and relieves the artist of many mundane tasks, from writing musical notes to reading them. Vangelis, who provided the score for the movie *Chariots of Fire*, cannot read music. But, he can generate art.

Computers are used to help in all kinds of animation, from Saturday morning cartoons to commercials. In fact, were it not for computers, we would have much less animation today than in the past. Salaries have increased a lot since Walt Disney directed a team of animators.

Computer-generated graphics are not only used to present subatomic particles and the structure of DNA (which looks like art to me), but pieces that hang in museums. Computers have been used to analyze the movements of dancers and runners for training purposes. John Swartzwelder has even used a CoCo to create one-line jokes (example: Scientists discover new moon orbiting Kate Smith). Computers have been used to mathematically analyze prose and poetry of great masters and then mimic that style to produce (some not so great) literary efforts.

So, if artists use computers, why should art students not be using them? There are several reasons. First, we do not expect students, especially elementary students, to produce great works of art. We expect them to learn about art instead of attempting to compete with professionals. Second, most of the computer applications mentioned here involve rather sophisticated software, which artists are unwilling to share with a group of elementary students. Also, there are few software publishers who provide computer-assisted instruction packages (still the most frequent use of computers in schools) for art. Finally, art teachers may not realize what students can do with the help of a computer.

I hold the position that students — even in primary grades — can use the computer to help them learn about the arts. I believe students should learn art by actually doing it. They should learn about painting by organizing colors and shapes; about music by organizing sound in meaningful patterns; about dance by studying movement.

Consider the easiest one first: music. My nephew recently showed me a music program for the Color Computer. The screen shows the music staffs, and by movement of the cursor and command keys, notes are placed on the staff. It seems to make sense that students would learn about music by writing it and listening to their compositions being played. The teacher could introduce such concepts as chords and harmony by showing students how to improve their own music. The students have a stake in their creations. Music by other people becomes more "real" to one who has written music.

It is difficult to convince a band to play a piece of music written by an elementary student. Especially when there may be 30 such students, each wanting to hear their own creation 100 times. It is not so difficult, on the other hand, to convince the Color Computer to play a musical composition written by a student. It is easy for the student to make changes in the music stored in a Color Computer and hear the results. It is hard for a student to change a few notes in a piece of music and expect a "live" band to immediately play again the altered piece. Thus, this piece of software seems to be an ideal teaching tool.

The initial barriers keeping the teacher away from the student are more easily overcome. The student learns about music by actively engaging in the process of organizing sound. It seems reasonable that such an approach would increase the motivation to learn how to play an instrument.

How about other arts? The computer can be used to help the student learn about painting as well as music. The Color Computer has some wonderful graphics packages. Students can first create, then print their masterpiece on paper to be displayed on the refrigerator door.

Beginning students make simple mistakes that can ruin an art project. With a graphics package, it is easy to correct these little problems before the final product is ready. This makes the final product easier to accept, as well as giving the student a valuable lesson that art is a dynamic, changing expression rather than a "first time only" event.

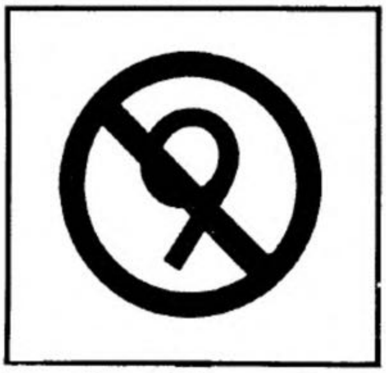
In kindergarten classes, a pre-art activity is to learn colors and shapes. The Color Computer is an ideal machine for computer-assisted instruction with kindergarten students. It is easy to imagine a student running a program which draws a triangle on the screen while the cassette recorder is talking about triangles. The triangle is colored red while the cassette provides a verbal explanation of the color red. As with all computer-assisted instruction packages, a major advantage is that the student can participate in the lesson as many times as needed.

Now, I do not know if such a program is already on the market. I doubt that it would require a master programmer to put together a unit on colors and shapes. If you have written one, please let me know about it and I will share that information through these pages.

So far, we have presented examples of computer-assisted art lessons for primary students. The same graphics packages mentioned could be used as a supplemental activity in a drafting class in a high school. The student could design the initial drafting project using a commercial graphics package, then complete the paper version based on changes and corrections made to the electronic rough sketch. Dance students in high school could use the screen as a model of a stage for choreographing movement, then translate the mental image to physical movement.

There is probably no end to the opportunities offered students with this blend of the computer and the arts. Before leaving this idea, a pratical point should be raised. The arts are a very important part of education and one of the key subject matter areas for students to learn about our cultural heritage. Yet, when budget crunches happen in schools, the arts often suffer the most. The use of computers can help continue this crucial part of education while still allowing for budget reductions.

I must mention one of the most creative arts, computer programming. Just think, by your knowledge and skill, you can use a medium of expression (your Color Computer) to generate a work of beauty. Your programs can be poetry, even if you are the only reader. I welcome your thoughts on this subject, and would appreciate reading anything you want to send to me at 829 Evergreen, Chatham, IL 62629.



NO NINES ALLOWED!

By Richard Ramella

No-niner is an innocent little listing. It's not much trouble to type in and run. But beyond that, it's a tricky test of your thinking ability. Some players have been Known to never beat it. If you want to consider that as a dare, then do.

My best score is 39, so that's the score you should equal or try to beat. I think a score of 40 or more is impossible, but I hope someone proves me wrong. (If so, I'd like to hear about it).

When you run the program you're presented with an eight-by-eight gric comprised of all letter O's, except for the 'X' in the northwest corner. To start, move the 'X' to the position where you want to begin. Tap the 'A' Key for north, 'Z' Key for south, comma Key for west and period Key for east. When ready to start the test, press 'P' for play. A numeral '1' appears in place of the 'X' and you've scored one point.

From this start, directional moves count upward to'8', then start over at '1'. Every move you make scores a point.

Of course, there's a catch. At any time during play, the game ends if any two numbers in a vertical row total nine or more. For example: At the start you press 'P' for play when the 'X' is in the northwest corner. Then, pressing the period Key seven times counts from '2' to '8', ending at the northeast corner.

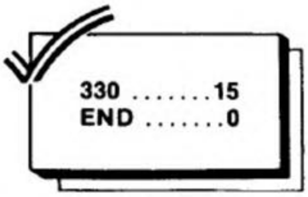
The only possible next move is south. You press the 'Z' Key, a '1' is set beneath the '8' and the game ends because the two cells' numbers total nine.

No-niner Keeps score, won't let you move off the playing board or into previously set numbers, and ends the game when you have exceeded the legal vertical total of eight.

In Keying in the game, remember the material within quotes in lines 160 and 390-430 is comprised of the letter 'O' and not a zero. Similarly, the material within quotes in Line 180 is the uppercase word for "OX" (as in strong as an ox).

This is a game for naturally mellow folKs who accept success as the reward for perseverance. Keep the desK pounders away from it; they might smite your CoCo to bits when they score a measly 16 points or so for the 20th time in a row!

(Any questions about No-niner may be directed to Mr. Ramella at 1493 Mt. View Avenue, Chico, CA95926. Please include an SASE.)



330 15
END 0

The listing: NONINER

```
1ØØ REM * NO-NINER * TRS-8Ø EXTE
NDED COLOR BASIC 16K
11Ø REM * BY RICHARD RAMELLA
12Ø CLEAR 5ØØ
13Ø DIM A$(1Ø)
14Ø CLS: PRINT "CAN YOU SCORE 39
?"
15Ø FOR C=1 TO 1Ø
16Ø A$(C)="ØØØØØØØØØØ"
17Ø NEXT C
18Ø A$(2)="ØX"+RIGHT$(A$(2),8)
19Ø X=2: Y=2: B$="12345678"
2ØØ GOSUB 55Ø
21Ø A$=INKEY$
22Ø IF A$="" THEN 21Ø
23Ø GOSUB 5ØØ: GOSUB 39Ø
24Ø IF A$="P" THEN MID$(A$(Y),X,
1)="1": GOSUB 55Ø: GOTO 27Ø
25Ø GOSUB 55Ø
26Ø GOTO 21Ø
27Ø Q=1: S=1: C=2
28Ø A$=INKEY$
29Ø PRINT @ 36Ø,"SCORE"S;
3ØØ IF A$="" THEN 28Ø
31Ø GOSUB 5ØØ
32Ø IF INSTR("AZ,.",A$)=Ø OR A$=
"" THEN 28Ø
33Ø GOSUB 39Ø: GOSUB 55Ø
34Ø U=VAL(MID$(A$(Y-1),X,1))
35Ø M=VAL(MID$(A$(Y),X,1))
36Ø P=VAL(MID$(A$(Y+1),X,1))
37Ø IF U+M>8 OR P+M>8 THEN XX=1Ø
Ø: GOTO 5ØØ
38Ø GOTO 28Ø
39Ø IF Q=Ø THEN MID$(A$(Y),X,1)=
"Ø"
4ØØ IF A$="A" AND Y>2 AND MID$(A
$(Y-1),X,1)="Ø" THEN Y=Y-1: L=1
41Ø IF A$="Z" AND Y<9 AND MID$(A
$(Y+1),X,1)="Ø" THEN Y=Y+1: L=1
42Ø IF A$="," AND X>2 AND MID$(A
$(Y),X-1,1)="Ø" THEN X=X-1: L=1
43Ø IF A$="." AND X<9 AND MID$(A
$(Y),X+1,1)="Ø" THEN X=X+1: L=1
44Ø IF Q=Ø THEN MID$(A$(Y),X,1)=
"X": RETURN
45Ø IF L=Ø THEN 28Ø ELSE L=Ø
46Ø MID$(A$(Y),X,1)=MID$(B$,C,1)
47Ø S=S+1: C=C+1
48Ø IF C=9 THEN C=1
49Ø RETURN
5ØØ IF XX<>1ØØ THEN RETURN
51Ø PRINT @ 416,"END SCORE:"S
52Ø FOR T=1 TO 1ØØØ
53Ø NEXT T
54Ø END
55Ø GH=76
56Ø FOR D=2 TO 9
57Ø PRINT @ GH,MID$(A$(D),2,8);:
 GH=GH+32
58Ø NEXT D
59Ø RETURN
6ØØ END
```

This program helps make the 'ups and downs' of learning easy

ELEVATOR

By Bill Bernico

Sesame Street teaches fundamentals to youngsters, helping them understand simple, everyday words like large or small, near or far, alike or different. Here's a program that helps them comprehend up and down.

This program, Elevator, at first glance, may appear to be strictly for young kids. Don't believe it! There are some pretty slick programming tricks integrated into this one - tricks you may want to use in your own programs even if you aren't overly entertained by an elevator making the rounds.

Constructing the seven floors of the building is done in lines 80-220, building being topped off by the word "ELEVATOR" in Line 230. Lines 240-250 ask if the user wants to go up or down. Lines 260-290 accept user input. Line 270 keeps the words "GOING UP" on the screen and erases the "OR". Line 280 keeps the word "GOING" on the first line, erasing the rest of that line. It also keeps the word "DOWN" on the second line, erasing "(U/D)."

The basis of the elevator action is held in lines 300 through 420. Lines 300-350 are GOSUBed if the user chooses 'D' and lines 360-420 are GOSUBed if the user chooses 'U'.

In either case, 'S' equals the value of the sound,with the top floor having the highest sound. 'N' equals the number of the floor displayed in the right window and is poked into value 'V'. I used POKE instead of PRINT @ because PRINT @ left unwanted spaces before and after the number.

As the elevator passes each floor a character string that's half black and half yellow appears in the window, which gives the impression of the elevator being halfway between floors. I chose the variable BY$ to represent the "Bottom Yellow" color block. TY$ is used when the "Top Yellow" block appears.

These actions are merely repeated in a FOR/NEXT loop until you've covered all seven floors. I'd be interested in hearing from anyone who can use this premise and build a more complex program around it. My address is 708 Michigan Avenue, Sheboygan, WI 53081, phone (414) 459-7350.

250 225
END 2

The listing: ELEVATOR

```
10 'ELEVATOR
20 'BY BILL BERNICO
30 '708 MICHIGAN AVE.
40 'SHEBOYGAN, WI 53081
50 '(414) 459-7350
60 '
70 L$=CHR$(128):A$=STRING$(10,19
1):B$=CHR$(191):C$=B$+B$+B$+L$:D
$=CHR$(159):E$=STRING$(2,191):TY
$=CHR$(156):BY$=CHR$(147)
80 CLS3:PRINT@498,A$;
90 PRINT@466,C$;C$;E$;
100 PRINT@434,A$;
110 PRINT@402,C$;C$;E$;
120 PRINT@370,A$;
130 PRINT@338,C$;C$;E$;
140 PRINT@306,A$;
150 PRINT@274,C$;C$;E$;
160 PRINT@242,A$;
170 PRINT@210,C$;C$E$;
180 PRINT@178,A$;
190 PRINT@146,C$;C$;E$;
200 PRINT@114,A$;
210 PRINT@82,C$;C$;E$;
220 PRINT@50,A$;
230 PRINT@18,B$;"elevator";B$;
240 PRINT@0,"GOING UP OR";
250 PRINT@32,"DOWN (U/D)?";
260 I$=INKEY$:IFI$=""THEN260
270 IF I$="U"THEN PRINT@8,"   ";
:PRINT@32,STRING$(11,143);:GOSUB
 360
280 IF I$="D"THEN PRINT@6,"
 ";:PRINT@37,"          ";:GOSUB 30
0
290 GOTO 80
300 S=180:N=55:V=1113:FOR Y=85 T
O 469 STEP 64
310 PRINT@Y,TY$;:GOSUB420
320 PRINT@Y,D$;:SOUNDS,1:POKEV,N
:GOSUB420
330 PRINT@Y,BY$;:GOSUB420
340 PRINT@Y,L$;:PRINT@Y+4,L$;:GO
SUB420
350 S=S-5:N=N-1:V=V+64:NEXTY:RET
URN
360 S=150:N=49:V=1497:FOR Y=469
TO 85 STEP-64
370 PRINT@Y,BY$;:GOSUB420
380 PRINT@Y,D$;:SOUNDS,1:POKEV,N
:GOSUB420
390 PRINT@Y,TY$;:GOSUB420
400 PRINT@Y,L$;:PRINT@Y+4,L$;:GO
SUB420
410 S=S+5:N=N+1:V=V-64:NEXT Y:RE
TURN
420 FORK=1TO200:NEXTK:RETURN
```

ZONX

by David Billen

Want great sound without sacrificing fast graphics? *Zonx* provides you with both! It proves that, without a doubt, the CoCo can produce sound without slowing the action, and as an added bonus, *Zonx* is fun!

To play *Zonx*, enter and run the program listing. If you have Extended BASIC, begin by typing PCLEAR 1. After you type RUN, the program begins reading the DATA statements and poking the numbers into memory. These numbers make up a machine language program. After all the numbers are poked, the BASIC programs will save the machine language program to tape. You can load this program by typing CLOADM and run it by typing EXEC. Before testing *Zonx*, be sure to save the BASIC program that created it just in case you make a typing error and it does not work.

(David Billen is a self-employeed programmer. His hobby is music synthesis, ranging from his CoCo to Syncho drums.)

Once the program is up and running the first thing you see is a title screen. At this point, *Zonx* also plays a little heavy metal rock-and-roll music while waiting for you to press the right joystick button.

The game begins and your joystick controls a ship that moves right or left near the bottom of the screen. The ship will not stay still — something to do with gravitational photon intervention.

Two different types of characters will fly down the screen, "Zonxes" and "Zugs." Zonxes are the ones that flash; Zugs are the sluggy-looking things. A Zonx will not harm you, but if you can touch it with the very tip of your ship, the game will have a brief "spasm" and 100 points are rewarded. Conflict with a Zug is always fatal and, since the goal is to earn points, I suggest avoiding them.

You will encounter a wave of Zonxes, a wave of Zugs and a wave of both. If you survive this far, bonus points are awarded and the level of difficulty increases. After the fourth increase in difficulty the game never gets any harder. If you make it that far, you are doing incredibly!

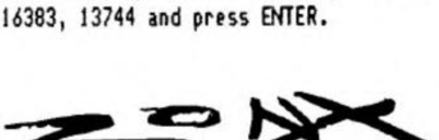
I welcome correspondence from those who have questions or comments. My address is P.O. Box 832, Edmond, OK 73083. Please include an SASE.

Good luck, it's all in the wrist! □

Editor's Note: As we were going to press, David Billen advised us that four POKEs will be needed to correct a problem that has arisen on some CoCos. You can type in the following line and CSAVE it on tape before the Zonx ML program; to play the game, simply load this program (leave the play button down on your recorder) and run it. After the four POKEs have been made, the ML program will be loaded and executed automatically.

```
10 CLS:PRINT@264,"LOADING zonx..
.":POKE359,127:POKE360,255:POKE3
61,34:POKE362,57:CLOADM"ZONX":EX
EC
```

If you want to use the Zonx program on disk, change CLOADM in this loader program to LOADM. Since the following listing will not run a disk system, run it with the controller unplugged and save the machine language on tape, then CLOADM it into your disk system later and type SAVEM "ZONX", 13740, 16383, 13744 and press ENTER.

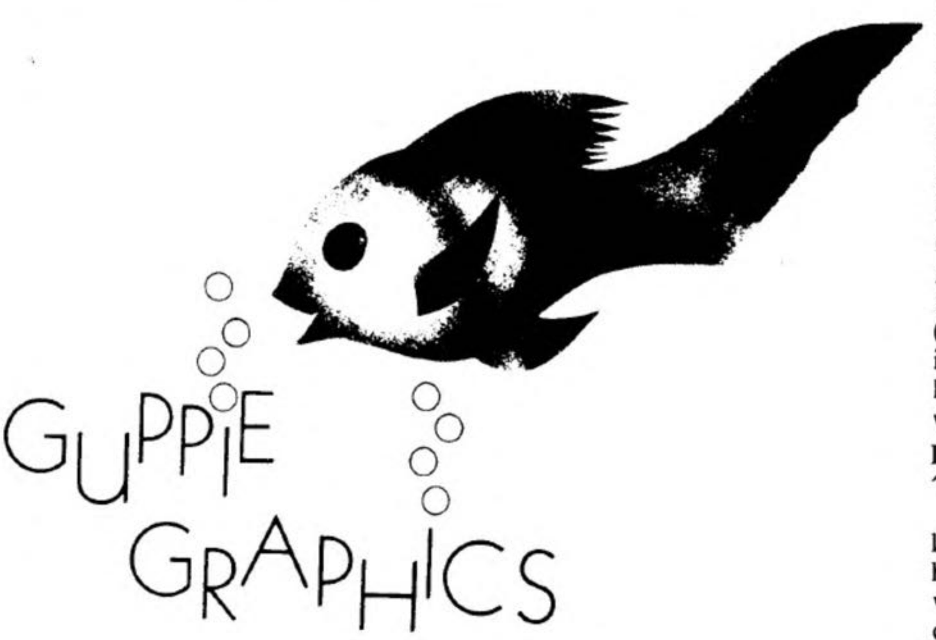


150	104	890	19
310	74	1030	41
440	106	1170	123
600	255	1300	166
750	26	END	238

The listing: ZONX

```
10 CLEAR 200,13700
15 CLS:PRINT:PRINT"LOADING MACHI
NE CODE..."
20 X=13744
30 READ A:IF A<>999 THEN POKE X,
A:X=X+1:GOTO 30
35 FOR X=15872 TO 16383:POKE X,0
:NEXT
50 T=6:FOR X=15891 TO 15905 STEP
 2:POKE X,T:POKE X+1,RND(255):T=
T+1:NEXT X:POKE 15907,13:POKE 15
908,RND(255)
60 PRINT:INPUT "READY RECORDER/P
RESS ENTER";A$
70 CSAVEM "ZONX",13740,16383,137
44
80 END
100 DATA 134,85,183,0,113,142,53
,187,191,0,114,18,134,62,31,139
110 DATA 142,53,218,191,1,13,134
,53,183,255,3,28,239,189,169,118
120 DATA 16,223,10,16,206,63,255
,126,56,162,182,255,2,12,0,59
130 DATA 13,12,16,38,1,191,182,2
55,32,31,137,132,3,138,128,183
140 DATA 255,32,182,255,0,247,25
5,32,214,18,132,128,38,3,90,42
150 DATA 7,92,193,62,35,2,214,18
,215,1,142,12,64,214,18,84
160 DATA 58,236,132,221,14,134,8
,16,142,0,0,16,175,132,48,136
170 DATA 32,74,38,247,134,255,15
1,8,142,12,64,214,1,215,18,84
180 DATA 58,37,46,15,15,134,24,1
48,8,167,136,32,134,60,148,8
190 DATA 167,136,64,167,136,96,1
34,219,148,8,167,137,0,128,167,1
37
200 DATA 0,160,167,137,0,192,134
,195,148,8,167,137,0,224,126,54
210 DATA 214,204,1,128,148,8,212
,8,237,136,32,204,3,192,148,8
220 DATA 212,8,237,136,64,237,13
6,96,204,13,176,148,8,212,8,237
230 DATA 137,0,128,237,137,0,160
,237,137,0,192,204,12,48,148,8
240 DATA 212,8,237,137,0,224,220
,14,132,15,196,240,221,14,126,54
250 DATA 214,222,4,166,192,17,13
1,192,0,37,3,206,160,0,223,4
260 DATA 57,142,62,19,16,174,132
,111,164,49,168,32,16,140,14,0
270 DATA 37,11,189,54,161,132,31
,16,142,4,0,49,166,16,175,129
280 DATA 140,62,37,37,223,57,142
,62,19,134,3,167,145,140,62,37
290 DATA 37,249,13,13,16,38,5,98
,57,13,16,38,15,142,62,37
300 DATA 111,129,140,62,53,37,24
9,12,16,15,73,57,142,62,37,189
310 DATA 55,82,38,42,167,164,167
,168,32,49,168,64,16,140,13,64
320 DATA 37,11,167,164,167,168,3
2,16,142,0,0,32,17,134,60,167
330 DATA 164,148,7,167,168,96,15
0,7,167,168,32,167,168,64,16,175
340 DATA 129,156,53,16,37,255,20
0,150,6,76,132,3,151,6,38,17
350 DATA 150,7,129,255,38,4,134,
85,32,5,72,36,2,134,255,151
360 DATA 7,57,16,174,132,16,140,
1,0,34,23,189,54,161,132,127
370 DATA 145,81,44,16,189,54,161
,132,31,16,142,4,0,49,166,16
380 DATA 175,132,79,57,134,255,5
7,134,255,151,0,150,73,42,2,19
390 DATA 57,16,38,0,95,214,74,92
,196,63,215,74,150,77,92,141
400 DATA 7,79,214,74,141,2,32,23
7,183,255,32,13,0,42,4,90
410 DATA 38,249,57,53,192,150,12
,129,1,38,17,151,73,76,151,12
420 DATA 134,232,183,255,34,19,1
34,248,183,255,34,57,76,38,8,16
430 DATA 206,63,255,126,56,162,5
7,151,12,134,248,183,255,34,142,
12
440 DATA 64,214,18,189,54,161,12
9,5,34,5,134,232,183,255,34,151
450 DATA 8,126,54,47,129,1,16,38
,0,143,150,12,31,137,64,151
460 DATA 1,189,54,161,148,1,189,
55,152,32,239,13,17,38,18,142
470 DATA 62,55,111,129,156,71,37
,250,12,17,134,2,151,73,151,76
480 DATA 57,142,62,55,189,55,82,
38,86,167,164,167,168,32,49,168
490 DATA 64,16,140,13,64,37,25,1
67,164,167,168,32,167,168,64,167
500 DATA 168,96,167,169,0,128,16
7,169,0,160,16,142,0,0,32,47
510 DATA 206,56,148,150,9,76,132
,15,151,9,150,17,13,9,38,7
520 DATA 76,129,5,37,2,134,1,151
,17,74,72,51,198,16,159,1
530 DATA 198,8,166,192,167,164,4
9,168,32,90,38,246,16,158,1,16
540 DATA 175,129,156,71,16,37,25
5,156,57,129,2,16,38,1,211,214
550 DATA 74,90,90,196,127,215,74
,150,77,189,55,152,79,214,74,189
560 DATA 55,152,32,235,24,36,24,
66,24,129,24,66,24,36,24,66
570 DATA 24,129,189,56,222,142,6
2,82,134,4,111,134,74,42,251,151
580 DATA 7,151,73,151,77,142,62,
12,76,167,128,140,62,19,37,249
590 DATA 142,160,0,159,4,126,58,
21,79,31,139,16,191,0,136,32
600 DATA 4,173,159,160,2,166,128
,38,248,134,62,31,139,57,142,255
610 DATA 192,111,132,111,2,111,4
,142,62,82,16,142,62,87,95,166
620 DATA 165,161,133,34,16,37,5,
92,193,5,37,243,198,5,166,133
630 DATA 167,165,90,42,249,142,4
,0,134,96,167,128,140,6,0,37
640 DATA 249,16,142,4,32,142,58,
126,189,56,200,16,142,4,96,189
650 DATA 56,200,16,142,4,203,189
```

```
,56,200,142,62,82,16,142,4,237
660 DATA 141,25,142,62,87,16,142
,5,45,141,16,142,4,0,166,132
670 DATA 136,64,167,128,140,6,0,
37,245,32,31,95,16,191,0,136
680 DATA 31,155,166,133,139,48,1
73,159,160,2,92,193,5,37,243,134
690 DATA 48,173,159,160,2,134,62
,31,139,57,182,255,0,132,1,39
700 DATA 249,126,59,75,142,54,23
3,159,79,15,16,79,151,81,151,12
710 DATA 167,226,68,151,77,189,5
5,119,189,54,177,189,53,224,173,
159
720 DATA 62,79,166,224,76,38,233
,142,6,0,150,78,151,81,175,227
730 DATA 189,55,119,189,54,177,1
89,53,224,13,12,38,20,220,14,39
740 DATA 16,132,102,196,102,16,1
31,0,0,39,4,12,13,32,2,12
750 DATA 12,173,159,62,79,174,22
5,48,31,38,211,79,151,81,167,226
760 DATA 68,151,77,189,55,119,18
9,54,177,189,53,224,173,159,62,7
9
770 DATA 166,224,74,38,233,57,14
2,55,251,159,79,15,17,126,57,123
780 DATA 142,58,0,159,79,15,17,1
5,16,15,74,15,75,126,57,123
790 DATA 189,54,233,189,55,251,1
3,12,38,10,13,13,38,6,134,3
800 DATA 151,73,151,76,57,134,1,
151,78,189,61,21,189,61,138,142
810 DATA 62,37,150,78,139,3,72,4
8,134,159,53,142,62,55,48,134
820 DATA 159,71,189,57,116,189,5
7,230,189,57,240,189,59,217,134,
255
830 DATA 151,7,189,61,138,150,78
,76,151,78,129,5,37,209,10,78
840 DATA 32,205,129,3,16,38,2,38
,214,74,92,196,63,215,74,38
850 DATA 9,214,75,90,39,4,196,12
7,215,75,214,75,219,74,150,77
860 DATA 72,189,55,152,214,74,21
9,75,79,189,55,152,32,218,32,32
870 DATA 32,32,32,32,32,32,32,32
,60,60,40,40,90,79,78,88
880 DATA 41,41,62,62,0,32,32,32,
32,32,32,32,32,66,89,32
890 DATA 68,65,86,73,68,32,66,73
,76,76,69,78,13,32,32,32
900 DATA 32,32,80,82,69,83,83,32
,66,85,84,84,79,78,32,84
910 DATA 79,32,66,69,71,73,78,0,
76,65,83,84,32,83,67,79
920 DATA 82,69,13,13,32,32,32,32
,32,32,32,32,32,32,32,72
930 DATA 73,71,72,32,83,67,79,82
,69,0,151,1,151,2,215,3
940 DATA 134,252,151,77,15,0,134
,4,167,226,150,77,214,1,141,64
950 DATA 79,214,1,141,59,106,228
,38,241,134,4,167,228,150,77,214
960 DATA 2,141,45,79,214,2,141,4
0,106,228,38,241,166,224,150,0
970 DATA 145,3,36,38,150,77,39,3
,74,151,77,150,1,129,255,39
980 DATA 3,76,151,1,150,2,129,1,
39,3,74,151,2,126,58,246
990 DATA 183,255,32,90,18,18,18,
18,38,249,57,142,59,133,166,128
1000 DATA 39,249,230,128,129,1,3
8,18,16,142,4,46,166,164,138,64
1010 DATA 167,160,16,140,4,50,37
,244,134,1,189,58,234,16,142,4
1020 DATA 46,166,164,132,191,167
,160,16,140,4,50,37,244,182,255,
0
1030 DATA 133,1,38,202,57,80,16,
16,16,1,16,32,32,16,16,1
1040 DATA 16,80,16,80,16,16,16,1
,16,32,32,16,16,1,16,80
1050 DATA 16,64,32,1,16,64,16,64
,32,1,16,48,16,64,32,1
1060 DATA 16,64,16,64,32,1,16,48
,16,80,16,16,16,1,16,32
1070 DATA 32,16,16,1,16,80,16,80
,16,16,16,1,16,16,16,16
1080 DATA 16,16,16,1,16,1,16,0,0
,134,10,214,78,61,231,226
1090 DATA 166,228,198,8,189,58,2
34,189,60,162,189,60,182,189,55,
64
1100 DATA 189,61,138,106,228,38,
233,53,2,142,60,20,166,128,16,39
1110 DATA 255,72,68,230,128,189,
58,234,52,16,189,55,64,189,61,13
8
1120 DATA 53,16,32,232,1,255,144
,32,144,32,144,32,144,32,144,16
1130 DATA 128,48,144,16,128,48,1
44,16,128,16,80,16,96,32,80,16
1140 DATA 80,16,80,16,144,16,128
,16,2,32,144,16,128,16,2,32
1150 DATA 144,16,128,16,80,16,96
,144,0,0,150,13,129,1,38,21
1160 DATA 150,73,151,76,134,4,15
1,73,151,13,134,232,183,255,34,1
89
1170 DATA 60,162,126,60,182,150,
13,139,64,151,13,36,16,15,13,204
1180 DATA 0,0,221,14,150,76,151,
73,134,248,183,255,34,57,129,4
1190 DATA 16,38,250,251,150,13,6
4,72,138,128,214,13,189,55,152,7
9
1200 DATA 214,13,189,55,152,32,2
37,150,86,76,151,86,129,10,37,21
1210 DATA 15,86,142,62,82,198,3,
166,133,76,167,133,129,10,37,5
1220 DATA 111,133,90,42,242,57,1
42,62,82,198,4,16,142,15,112,166
1230 DATA 133,206,60,227,72,72,1
71,133,51,198,134,5,151,1,16,159
1240 DATA 2,166,192,167,165,49,1
68,32,10,1,38,245,16,158,2,90
1250 DATA 42,221,57,3,51,51,51,3
,15,207,207,207,3,3,243,3
1260 DATA 63,3,3,243,195,243,3,5
1,51,3,243,243,3,63,3,243
1270 DATA 3,3,63,3,51,3,3,243,24
3,243,243,3,51,3,51,3
1280 DATA 3,51,3,243,243,142,4,0
,111,128,140,14,0,37,249,134
1290 DATA 255,167,128,140,16,0,3
7,249,142,255,192,111,0,111,2,11
1
1300 DATA 5,134,248,183,255,34,1
6,142,15,113,142,62,82,198,4,189
1310 DATA 60,191,189,60,182,142,
61,113,16,142,15,111,16,159,1,16
1320 DATA 142,15,106,189,61,87,5
7,52,32,198,5,166,128,152,7,167
1330 DATA 164,49,168,32,90,38,24
4,53,32,49,33,16,156,1,38,231
1340 DATA 57,60,192,48,12,240,60
,192,192,192,60,48,204,204,204,4
8
1350 DATA 240,204,240,204,204,25
2,192,240,192,252,142,61,210,16,
142,14
1360 DATA 44,52,32,198,8,215,6,1
66,128,141,42,215,3,141,38,150
1370 DATA 3,221,1,68,84,154,1,21
8,2,148,7,212,7,136,255,200
1380 DATA 255,237,164,49,168,32,
10,6,38,221,53,32,49,34,16,140
1390 DATA 14,52,37,205,57,198,4,
215,1,95,72,89,89,10,1,38
1400 DATA 249,57,254,4,8,16,32,6
4,128,254,124,130,130,130,130,13
0
1410 DATA 130,124,130,130,162,14
6,146,138,130,130,130,130,68,40,
40,68
1420 DATA 130,130,65,82,48,50,54
,180,150,83,84,65,82,48,52,54
1430 DATA 205,134,83,84,65,82,88
,50,54,219,148,76,73,70,69,60
1440 DATA 999
1450 DATA 6,161,7,24,8,154,9,17,
10,238,11,16,12,1,13,62,13,153,9
99
```



GUPPIE GRAPHICS

By Jim Kent

I've been a RAINBOW reader for years. The first high resolution machine language graphics program I ever worked was Dennis Lewandowski's shoot-'em-up space game. Since then, I've been plugging away at my little CoCo and now have a program of my own to offer - Digital Aquarium.

A "digital aquarium" is no substitute for a real aquarium. It doesn't make gurgling noises (though there are bubbles) and the fish won't recognize you or flare their fins at a mirror. Still, there are no water filters to clean and, if one tires of a fish, one can simply remove it from the object list with no twinges of conscience. It is also a good example of what your CoCo can do when you ask it to. There are over 50 objects being moved at once. (I'd like to see that on a Commodore!)

It would really be best to get RAINBOW ON TAPE to enjoy this article. I've included a listing of the assembly language source. Those who don't already know can see that machine language programs tend to be very long if they intend to do much. This is because each line in assembly code does only something very simple. The basic line:

```
i = j + k
```

translates to:

```
lda j
adda k
sta i
```

Similarly, a BASIC line of any complexity is apt to turn into a CoCo screen-full of assembly language. Unless you are trying to learn assembly language, typing in this program will probably be impossible, for a single typing error will be enough to make everything not work. It is a pretty program, and I hope people not trying to learn assembly language will be able to enjoy it as well.

For those who have done some graphics programming, *Digital Aquarium* derives its remarkable speed and smoothness through the use of preshifted shapes, a DRAW-DRAW architecture, as well as from being written in fairly tightly-coded 6809 assembler. Anyone wishing to incorporate parts of this code into any of their programs, even commercial ones, should feel free to do so. I hope it encourages some games that *don't* involve missiles in space or jungle munch-'em-ups. My only regret with *Digital Aquarium* is that it doesn't use joysticks — yet.

There ?re two listings that accompany this article. Listing 1, *F*, written to be assembled using The Micro Works' Macro-80C, is the machine language code that actually does the work of animating the fish and also contains the code to produce a small aquarium. Listing 2, ***FISH***, is a BASIC program that uses the *Digital Aquarium* as a USR ( ) routine. The aquarium will run until a key is pressed, then it will return to BASIC. The PCLS ( ) call can be followed by other BASIC graphics commands if a more elaborate, non-moving background is desired. (The October 1985 RAINBOW ON TAPE includes a longer version of the machine language program, called *FLARGE*, as well. To use this program, just change Line 20 of Listing 2 to LOADM "FLARGE".

I have tried to document the assembly language as much as possible. I did not have the luxury of an 80-column screen when I developed this program, so comments appear on the line after the code they refer to rather than at the end of the same line. All parameters are passed to subroutines in registers. Subroutines are required to save registers they use unless those registers return values. The one exception to this is the software blitter (rectangle mover) routine named XORBLIT.

The only truly hairy routine in this package is CLPBLT or "clip blit." Personally, I am just happy it works and that I no longer have to understand exactly how. This is a vital routine, however. If it weren't for CLPBLT, the fish that are half on screen would also be half on the code, and believe me, fins in place of RTSs lead quickly to seaweed soup!

Most of the machine code is set up to interpret the animation list, which is declared at the bottom of Listing 1 and Listing 2. Members of the animation list are characters, which are themselves composed of three parts: position, path and image. In each frame three functions are called for each character. First, a function associated with the path of the character is called and updates the position, then a function associated with the image is called and stuffs Register X with a "raster" (a rectangular array of pixels — little square dots), and Registers A and B with the dimensions of the raster. Next a function to draw the raster at the character position is called. The fun part of this setup is that different images can have the same path and vice-versa. Furthermore, new types of paths and new types of images can easily be added.

Currently there are four types of paths: STRAIT, BOUNCE, YBOUNC and SAMPLED. The STRAIT and BOUNCE paths require only two numbers to define in addition to a "type byte" at the beginning. These are naturally the horizontal and vertical offsets to be added to the character position in each frame.

The SAMPLED path is a bit more complex. The format is a type byte followed by two bytes to say how many samples there are, two bytes to say which sample we're on now, and then a list of the samples themselves. The samples are really offsets just like the straight type path. The difference is that you move each frame on to the next offset rather than always using the same one.

To go with the four types of paths there are three types of images. The first of these is STILL. A STILL contains the type byte stating it is a STILL, followed by the dimensions of the raster containing the image, followed by the raster itself. STILL shapes will jerk if moved horizontally, but require only one-fourth the memory of the other types of shapes.

The second type of image is GLIDE. As you may know, in the color Hi-Res screen modes of your CoCo a single pixel takes up two bits. This means there are four pixels in a byte. The CoCo is a byte-oriented machine; it is not especially good at shifting around bits within a byte — it will do it, but not nearly as fast as it will throw around bytes whose bits it doesn't have to twiddle.

The upshot of this is that if you want smooth movement in the horizontal direction, you must have four copies of an image, one for each pixel that lives in a byte. This way, when you want the first pixel of an image to go to a place on the screen that is itself, say, the third pixel of a byte, you do not spend ages shifting the whole image a bit at a time before making the copy, but rather simply decide which one (in this case the third) of four pre-shifted images to put as a byte array onto the screen.

Therefore, when an image is declared to be type GLIDE you supply the first raster just as if it were a STILL, but then you reserve space for three more copies of the image - pre-shifted copies, for the machine to fill in once it gets going.

The type WIGGLE is very much like GLIDE, except instead of the CoCo doing the pre-shifting you do it yourself. The advantage of this is that without using any more memory than it would take for smooth motion, you can have a shape that wiggles a little as it moves horizontally.

The program only works on machines with Extended Color Disk BASIC. To get it to run on a cassette-based system add the line: 25POKE 3000,8 to Listing 2. This tells it your screen is starting at $800 instead of $E00. It should run on 16K machines with no problem.

The Digital Aquarium was written using a Micro Works Macro-80C assembler and editor with local lables. (That's what all the A@'s are, folks.) If you own a less intelligent assembler, I'm afraid you'll have to make up new labels for them.

Listing 1: F

```
0080                XMAX    EQU 128
0060                YMAX    EQU 96
                    *DIMENSIONS OF SCREEN

0E00                        ORG $3000

3000 0E00           SCREEN FDB $E00
                    *POKE IN NEW SCREEN LOCATION
                    *HERE IF WANT TO

3002                        ORG $3010
                    *SKIP A BIT .. ROOM FOR
                    *MORE PARAMETERS FOR LATER
                    *VERSIONS

3010                ZTART
3010 CC0E00                 LDD #$E00
3013 FD3000                 STD SCREEN
3016 CE330B                 LDU #ANLIST
3019 170012                 LBSR OBINIT
                    *PRE-SHIFT SHAPES

301C 170088                 LBSR DRAALL
                    *DRAW EVERYTHING ONE UP TOP SO
                    *LOOP CAN BE UNDRAW-DRAW INSTEAD
                    *OF DRAW-UNDRAW

301F                LOOP
301F 170098                 LBSR FRAME
3022 170223                 LBSR INTRPT
3025 27F8                   BEQ LOOP
                    *GO AROUND FOREVER OR UNTIL
                    *USER HITS KEYBOARD

3027 CE330B                 LDU #ANLIST
302A 17007A                 LBSR DRAALL
                    *FINAL ERASING XOR

302D 39                     RTS
                    *RETURN ALL THE WAY TO BASIC

                    *******************************
                    *
                    *       OBINIT
                    *
                    *INPUT:
                    *       U->CHARACTER LIST
                    *
                    *WILL PROMOTE GLIDE TYPE IMAGES
                    *TO WIGGLE TYPES BY FILLING
                    *IN THE PRESHIFTED IMAGES
                    *******************************
302E 3452           OBINIT PSHS A,U,X

3030 11830000               CMPU #0
                    *JUST TO SET CONDITION CODE
3034 2013                   BRA C@
3036 AEC90004       A@      LDX CIMAGE,U
303A A6890000               LDA ITYPE,X
303E 8102                   CMPA #GLIDE
3040 2603                   BNE B@
3042 170008                 LBSR PREGLI

3045 EEC90006       B@      LDU CNEXT,U
3049 26EB           C@      BNE A@

304B 35D2                   PULS A,U,X,PC

                    *******************************
                    *
                    *       PREGLI
                    *
                    *INPUT:
                    *       X->IMAGE
                    *
                    *MAKES 3 SHIFTED VERSIONS
                    *OF RASTER AND ALL STENCILS
                    *******************************
304D                PREGLI
304D 3476                   PSHS X,Y,U,A,B
304F 3276                   LEAS -10,S

3051 8601                   LDA #WIGGLE
3053 A7890000               STA ITYPE,X
                    *PROMOTE TO PRESHIFTED TYPE

3057 EC890001               LDD IXDIM,X
305B EDE4                   STD ,S
                    *SAVE DIMENSIONS

305D EC890003               LDD ILENG,X
3061 ED62                   STD 2,S
                    *SAVE RASTER SIZE

3063 30890005               LEAX IDATA,X
                    *POINT X TO START OF FIRST RASTER
3067 338B                   LEAU D,X
                    *POINT U TO NEXT RASTER
```

```
ØØ41 3Ø69 86Ø3             LDA #3
ØØ42 3Ø6B A764             STA 4,S
                   *COUNT THROUGH THREE RASTERS

ØØ43 3Ø6D ECE4     A@      LDD ,S
ØØ44 3Ø6F 1CFE             ANDCC #$FE       CLEAR CARRY
ØØ45 3Ø71 17ØØ13           LBSR ASRXU
ØØ46 3Ø74 1F31             TFR U,X
ØØ47 3Ø76 1CFE             ANDCC #$FE
ØØ48 3Ø78 17ØØØC           LBSR ASRXU
ØØ49 3Ø7B EC62             LDD 2,S
ØØ5Ø 3Ø7D 33CB             LEAU D,U
ØØ51 3Ø7F 6A64             DEC 4,S
ØØ52 3Ø81 26EA             BNE A@
                   *SHIFT THE IMAGES

ØØ53 3Ø83 326A     PREANZ LEAS 1Ø,S
ØØ54 3Ø85 35F6             PULS A,B,X,Y,U,PC

                   ******************************
                   *
                   *       ASRXU
                   *
                   *INPUT:
                   *       X->SOURCE RASTER
                   *       U->DESTINATION RASTER
                   *       A = X DIMENSION OF
                   *               RASTER
                   *       B = Y DIMENSION
                   *
                   *SHIFTS RASTER RIGHT ONE.
                   *CARRY BIT MUST BE SET ABOVE
                   ******************************
ØØ55 3Ø87 3456     ASRXU   PSHS X,U,A,B
ØØ56 3Ø89 327B             LEAS -5,S

ØØ57 3Ø8B EDE4             STD ,S
ØØ58 3Ø8D 1FA9             TFR CC,B
ØØ59 3Ø8F E762             STB 2,S

ØØ6Ø 3Ø91 E662     ASRXU1 LDB 2,S
ØØ61 3Ø93 1F9A             TFR B,CC
                   *PUT CARRY BIT IN RIGHT PLACE
ØØ62 3Ø95 E68Ø     A@      LDB ,X+
ØØ63 3Ø97 56               RORB
ØØ64 3Ø98 E7CØ             STB ,U+
ØØ65 3Ø9A 4A               DECA
ØØ66 3Ø9B 26F8             BNE A@
                   *SHIFT ONE LINE
ØØ67 3Ø9D A6E4             LDA ,S
ØØ68 3Ø9F 6A61             DEC 1,S
ØØ69 3ØA1 26EE             BNE ASRXU1

                   *SHIFT RASTER

ØØ7Ø 3ØA3 3265             LEAS 5,S
ØØ71 3ØA5 35D6             PULS X,U,A,B,PC

                   ******************************
                   *
                   *       DRAALL
                   *
                   *INPUT:
                   *       U->CHARACTER LIST
                   *
                   ******************************
ØØ72 3ØA7          DRAALL
ØØ73 3ØA7 344Ø             PSHS U
ØØ74 3ØA9 1183ØØØØ         CMPU #Ø
ØØ75 3ØAD 2ØØ7             BRA C@
ØØ76 3ØAF 17ØØ38   A@      LBSR DRAOBJ
ØØ77 3ØB2 EEC9ØØØ6         LDU CNEXT,U
ØØ78 3ØB6 26F7     C@      BNE A@

ØØ79 3ØB8 35CØ             PULS U,PC

                   ******************************
                   *
                   *       FRAME
                   *
                   *INPUT:
                   *       U-> START OF CHARACTER
                   *               LIST
                   ******************************
ØØ8Ø 3ØBA 344Ø     FRAME   PSHS U
ØØ81 3ØBC 1183ØØØØ         CMPU #Ø
ØØ82 3ØCØ 2ØØD             BRA C@
ØØ83 3ØC2 17ØØ25   A@      LBSR DRAOBJ
                   *ERASE XOR IMAGE
ØØ84 3ØC5 17ØØØB           LBSR MOVOBJ
ØØ85 3ØC8 17ØØ1F           LBSR DRAOBJ
                   *ACTUALLY DRAW THE THING
```

```
ØØ86 3ØCB EEC9ØØØ6            LDU CNEXT,U
ØØ87 3ØCF 26F1        C@      BNE   A@
ØØ88 3ØD1 35CØ                PULS U,PC

                      *****************************
                      *
                      *       MOVOBJ
                      *
                      *INPUT:
                      *       U->CHARACTER
                      *
                      *MOVES CHARACTER
                      *FORWARD ONE 'CLICK' IN TIME
                      *****************************
ØØ89 3ØD3 3436        MOVOBJ PSHS D,X,Y
ØØ9Ø 3ØD5 1ØAEC9ØØØ2          LDY CPATH,U
ØØ91 3ØDA A6A9ØØØØ            LDA PTYPE,Y
ØØ92 3ØDE 48                  LSLA
ØØ93 3ØDF 8E31Ø4              LDX #NOFFT
ØØ94 3ØE2 AD96                JSR [A,X]
                      *LOOK UP FUNCTION THAT
                      *MOVES THIS TYPE OF OBJECT
ØØ95 3ØE4 EDC9ØØØØ            STD CXPOS,U
                      *AND STORE FUNCTION RESULT
                      *IN OBJECT POSITION

ØØ96 3ØE8 35B6                PULS X,Y,D,PC

                      *****************************
                      *
                      *       DRAOBJ
                      *
                      *INPUT:
                      *       U->CHARACTER
                      *****************************
ØØ97 3ØEA 3436        DRAOBJ PSHS A,B,Y,X
ØØ98 3ØEC 1ØAEC9ØØØ4          LDY CIMAGE,U
ØØ99 3ØF1 A6A9ØØØØ            LDA ITYPE,Y
Ø1ØØ 3ØF5 48                  LSLA
Ø1Ø1 3ØF6 8E31ØC              LDX #DOBJT
Ø1Ø2 3ØF9 AD96                JSR [A,X]
                      *CALL FUNCTION TO GET NEXT
                      *RASTER AND STENCIL
Ø1Ø3 3ØFB ECC9ØØØØ            LDD CXPOS,U
                      *GET CURRENT POSITION
Ø1Ø4 3ØFF 17ØØ8C              LBSR CLPBLT

Ø1Ø5 31Ø2 35B6                PULS A,B,X,Y,PC


Ø1Ø6 31Ø4 3145316B31  NOFFT  FDB NEWBOU,NEWSMP,NEWSTR
Ø1Ø7 31ØA 3134               FDB NEWYBC

Ø1Ø8 31ØC 311Ø3115    DOBJT  FDB DSTILL,DWIGGL

                      *****************************
                      *
                      *       DSTILL
                      *
                      *INPUT Y->IMAGE
                      *       U->CHARACTER
                      *
                      *OUTPUT X->RASTER
                      *
                      *DRAW A STILL OBJECT
                      *****************************
Ø1Ø9 311Ø             DSTILL
Ø11Ø 311Ø 3ØA9ØØØ5            LEAX IDATA,Y
Ø111 3114 39                  RTS

                      ******************************
                      *
                      *       DWIGGLE
                      *
                      *INPUT Y->IMAGE
                      *       U->CHARACTER
                      *
                      *OUTPUT X->RASTER
                      *
                      *DRAW AN OBJECT CAPABLE OF
                      *SMOOTH MOVEMENT, IE WITH
                      *4 PRESHIFTED SHAPES
                      ******************************
Ø112 3115             DWIGGL
Ø113 3115 34Ø6                PSHS A,B
Ø114 3117 3ØA9ØØØ5            LEAX IDATA,Y
                      *POINT X TO START OF FIRST
                      *RASTER

Ø115 311B E6C9ØØØØ            LDB CXPOS,U
Ø116 311F C4Ø3                ANDB #3
                      *FIND WHICH PIXEL IN BYTE
```

```
0117 3121 5C                    INCB
0118 3122 3404                  PSHS B
                        *AND PUSH IT TO USE AS COUNTER

0119 3124 ECA90003              LDD ILENG,Y

0120 3128 2002                  BRA C@
0121 312A 308B          A@      LEAX D,X
0122 312C 6AE4          C@      DEC ,S
0123 312E 26FA                  BNE A@

0124 3130 3504                  PULS B
                        *CLEAR COUNTER

0125 3132 3586                  PULS A,B,PC

                        *****************************
                        *
                        *       NEWYBC
                        *
                        *NEW Y BOUNCE.
                        *
                        *INPUT:
                        *       U->CHARACTER

                        *       Y->PATH
                        *
                        *OUTPUT:
                        *       A-XPOS
                        *       B-YPOS
                        *
0126 3134               NEWYBC
0127 3134 ECC90000              LDD CXPOS,U
0128 3138 AB21                  ADDA 1,Y

0129 313A EB22          A@      ADDB 2,Y
0130 313C C160                  CMPB #YMAX
0131 313E 2504                  BLO Z@
0132 3140 6022                  NEG 2,Y
0133 3142 20F6                  BRA A@
0134 3144               Z@

0135 3144 39                    RTS

                        ******************************
                        *
                        *       NEWBOU
                        *
                        *INPUT:
                        *       U->CHARACTER
                        *       Y->PATH
                        *
                        *OUTPUT:
                        *       A - X POSITION
                        *       B - Y POSITION
                        *
                        *UPDATES BOUNCE POSITION,
                        *AND CHANGES DIRECTION AT WALLS
                        ******************************
0136 3145               NEWBOU

0137 3145 A6C90000      NBX     LDA CXPOS,U     XPOSITION
0138 3149 AB21                  ADDA 1,Y        XSPEED
0139 314B 817B                  CMPA #XMAX-5
0140 314D 2504                  BLO NBXOK
0141 314F 6021                  NEG 1,Y         BOUNCE
0142 3151 20F2                  BRA NBX
0143 3153               NBXOK
0144 3153 E6C90001      A@      LDB CYPOS,U     YPOS
0145 3157 EB22                  ADDB 2,Y        YSPEED
0146 3159 C15B                  CMPB #YMAX-5
0147 315B 2504                  BLO Z@
0148 315D 6022                  NEG 2,Y
0149 315F 20F2                  BRA A@
0150 3161               Z@

0151 3161 39                    RTS


                        ******************************
                        *
                        *       NEWSTR
                        *
                        *INPUT:
                        *       U->CHARACTER
                        *       Y->PATH
                        *OUTPUT:
                        *       A - X POSITION
                        *       B - Y POSITION
                        *
                        *GET NEXT POSITION FOR A
```

```
                        *STRAIGHT PATH CHARACTER
                        ******************************
0152 3162               NEWSTR
0153 3162 ECC90000              LDD  CXPOS,U
0154 3166 AB21                  ADDA 1,Y
0155 3168 EB22                  ADDB 2,Y
0156 316A 39                    RTS

                        ******************************
                        *
                        *       NEWSMP
                        *
                        *INPUT:
                        *       U->CHARACTER
                        *       Y->PATH
                        *
                        *OUTPUT:
                        *       A - XPOS
                        *       B - YPOS
                        *
                        *UPDATES POSITION OF CHARACTER
                        *BASED ON A OFFSET LIST TYPE
                        *PATH
                        ******************************
0157 316B               NEWSMP
0158 316B 3410                  PSHS X

0159 316D 3021                  LEAX 1,Y
0160 316F EC84                  LDD ,X
0161 3171 C30001                ADDD #1
0162 3174 10A302                CMPD 2,X
0163 3177 2503                  BLO A@
0164 3179 CC0000                LDD #0
0165 317C ED84          A@      STD ,X
                        *UPDATE POINTER TO NEXT SAMPLE
                        *AND MOVING POINTER BACK TO
                        *START IF PAST END

0166 317E 3004                  LEAX 4,X
                        *MOVE OVER QUEUE POINTERS
0167 3180 58                    LSLB
0168 3181 49                    ROLA
                        *TWO BYTES/ENTRY
0169 3182 308B                  LEAX D,X
                        *MOVE X TO CURRENT SAMPLE

0170 3184 ECC90000              LDD CXPOS,U
0171 3188 AB84                  ADDA ,X
0172 318A EB01                  ADDB 1,X
                        *ADD THIS AS AN OFFSET FROM
                        *CURRENT LOCATION

0173 318C 3590                  PULS X,PC

                        ******************************
                        *
                        *       CLPBLT
                        *
                        *INPUT:
                        *   D XY LOCATION OF UPPER LEFT
                        *   X POINTS TO RASTER
                        *   Y ->IMAGE
                        *
                        *CLIPS RASTER TO SCREEN AND
                        *THEN CALLS XORBLT TO
                        *ACTUALLY DRAW IT
                        ******************************
0174 318E 3476          CLPBLT PSHS A,B,U,X,Y
0175 3190 3271                  LEAS -15,S

0176 3192 3365                  LEAU 5,S        U TO STACK AREA
0177 3194 EDE4                  STD ,S
                        *SAVE UPPER LEFT POS.OF RASTER

0178 3196               CLPVER
0179 3196 E622                  LDB 2,Y
0180 3198 E7C4                  STB ,U          # OF LINES
0181 319A AF46                  STX 6,U         RASTER START
                        *START OFF OPTIMISTIC, INITIAL-
                        *IZE THINGS FOR NO CLIPPING

0182 319C A661          CLPUP  LDA 1,S
0183 319E A762                  STA 2,S
0184 31A0 2A1A                  BPL CLPDN
                        *IF BELOW TOP OF SCREEN OK
0185 31A2 ABC4                  ADDA ,U
0186 31A4 102B007F              LBMI BLITZ
0187 31A8 1027007B              LBEQ BLITZ
0188 31AC A7C4                  STA ,U          LESS LINES
                        *DON'T BOTHER IF COMPLETELY
                        *OFF THE TOP
0189 31AE 40                    NEGA
```

```
0190 31AF AB22             ADDA 2,Y
0191 31B1 E621             LDB 1,Y       RASTER WIDTH
0192 31B3 3D               MUL
0193 31B4 E346             ADDD 6,U
0194 31B6 ED46             STD 6,U
                    *MOVE RASTER POINTER FORWARD ONE
                    *RASTER WIDTH FOR EACH PIXEL
                    *ABOVE SCREEN
                    *IE "CLIP UPWARDS"
0195 31B8 6F62             CLR 2,S       YCOOR SCREEN 0
0196 31BA 2016             BRA CLPHOR

0197 31BC A661      CLPDN  LDA 1,S       UPPER BOUND OF RA
0198 31BE           STER
0199 31BE AB22             ADDA 2,Y      HEIGTH OF RASTER
0200 31C0 8060             SUBA #YMAX
0201 31C2 250E             BLO CLPHOR
                    *IF BOTTOM OF OBJECT IN SCREEN
                    *NO PROBLEM
0202 31C4 8660             LDA #YMAX
0203 31C6 A061             SUBA 1,S
0204 31C8 102B005B         LBMI BLITZ
0205 31CC 10270057         LBEQ BLITZ
                    *IF TOP OF OBJECT BELOW SCREEN
                    *NOTHINGTO DRAW
0206 31D0 A7C4             STA ,U
                    *ELSE STORE ACTUAL # OF LINES
                    *OF OBJECT ON SCREEN
0207 31D2           CLPHOR
0208 31D2 E621             LDB 1,Y       RASTER WIDTH
0209 31D4 E741             STB 1,U       TO BLAT WIDTH
                    *START OPTIMISTIC W/ NO CLIPPING

0210 31D6 E6E4      CLPLF  LDB ,S        ABSOLUTE X POSN
0211 31D8 57               ASRB
0212 31D9 57               ASRB          FROM PIXEL TO BYTE
0213 31DA E7E4             STB ,S        X BYTE POSITION
0214 31DC E763             STB 3,S       XPOSN AFTER CLIP
0215 31DE 2A18             BPL CLPRT
                    *IF POSITIVE THEN NOT TOO FAR
                    *LEFT
0216 31E0 C1C0             CMPB #SC0     HALF WRAP-AROUN
0217 31E2 2514             BLO CLPRT
                    *IF THINK WAY POSITIVE INSTEAD
                    *OF JUST NEGATIVE HAND IT TO
                    *RIGHT CLIPPING
0218 31E4 EB21             ADDB 1,Y      RASTER WIDTH
0219 31E6 2B3F             BMI BLITZ
0220 31E8 273D             BEQ BLITZ
                    *IF EVEN RIGHT BYTE NEGATIVE
                    *DON'T EVEN BLIT
0221 31EA E741             STB 1,U       BLAT WIDTH
0222 31EC 50               NEGB
0223 31ED EB21             ADDB 1,Y      B HAS MODULO
0224 31EF AE46             LDX 6,U       RASTER START
0225 31F1 3A               ABX
0226 31F2 AF46             STX 6,U       ADD TO RASTER PT
                    *ELSE ADJUST WIDTH AND MODULUS
                    *(SKIP WIDTH) FOR RASTER, AND
                    *MOVE RASTER POINTER FORWARD
                    *TO FIRST BYTE ACTUALLY DRAWN
0227 31F4 6F63             CLR 3,S
                    *START SCREEN AT LEFT
0228 31F6 200C             BRA ALLCLP

0229 31F8           CLPRT
0230 31F8 8620             LDA #XMAX/4
0231 31FA A0E4             SUBA ,S
0232 31FC 2B29             BMI BLITZ
                    *ALL THE WAY OFF SCREEN
0233 31FE A121             CMPA 1,Y
0234 3200 2402             BHS ALLCLP
                    *ALL IN SCREEN
0235 3202 A741             STA 1,U       BYTES/LINE

0236 3204           ALLCLP
                    *END OF CLIPPING..WHEW!
0237 3204 A662             LDA 2,S       Y DEST.
0238 3206 C620             LDB #32
0239 3208 3D               MUL
0240 3209 F33000           ADDD SCREEN
0241 320C 1F01             TFR D,X
0242 320E E663             LDB 3,S       X DEST.
0243 3210 3A               ABX
                    *X POINTS TO DESTINATION ON
                    *SCREEN

0244 3211 AF44             STX 4,U       SCREEN START

0245 3213 EC46             LDD 6,U       RASTER START
0246 3215 E323             ADDD 3,Y      RASTER LENGTH
0247 3217 ED48             STD 8,U       STENCIL START
```

```
0248 3219 8620             LDA #32
0249 321B A041             SUBA 1,U      LINE LENGTH
0250 321D A742             STA 2,U       SCREEN MODULO

0251 321F A621             LDA 1,Y       RASTER WIDTH
0252 3221 A041             SUBA 1,U      LINE LENGTH
0253 3223 A743             STA 3,U       RASTER MODULO

0254 3225 8D04             BSR XORBLT

0255 3227           BLITZ

0256 3227 326F             LEAS 15,S
0257 3229 35F6             PULS A,B,U,X,Y,PC

                    ********************************
                    *
                    *      XORBLIT
                    *
                    *INPUT: U->
                    *     ,U LINES IN RASTER
                    *    1,U #OF LINES IN RASTER
                    *    2,U SCREEN MODULUS
                    *    3,U RASTER MODULUS
                    *    4,U SCREEN START
                    *    6,U RASTER START
                    *
                    *XORS SOURCE WITH DEST,
                    *TRASHES ALL REGISTER AS WELL
                    *AS INPUT STRUCTURE.
                    *THIS IS OK - CLPBLT EXPECTS IT
                    ********************************
0258 322B           XORBLT
0259 322B AE44             LDX 4,U
0260 322D 10AE46           LDY 6,U
0261 3230 E641      A@     LDB 1,U
0262 3232 A684      B@     LDA ,X
0263 3234 A8A0             EORA ,Y+
0264 3236 A780             STA ,X+
0265 3238 5A               DECB
0266 3239 26F7             BNE B@
0267 323B A642             LDA 2,U
0268 323D 3086             LEAX A,X
0269 323F A643             LDA 3,U
0270 3241 31A6             LEAY A,Y
0271 3243 6AC4             DEC ,U
0272 3245 26E9             BNE A@
0273 3247 39               RTS

                    ********************************
                    *
                    *      INTERRUPT
                    *
                    *CHECKS FOR KEYBOARD OR PENDOWN
                    *RETURNS EQ FOR NO KEYBOARD OR
                    *NEW PENDOWN
                    ********************************
0274 3248           INTRPT
0275 3248 3412             PSHS X,A
0276 324A 4F               CLRA
0277 324B B7FF02           STA $FF02
0278 324E B6FF00           LDA $FF00
0279 3251 8A80             ORA #$80
0280 3253 43               COMA
0281 3254 3592             PULS X,A,PC

0282 3256           ZENDCD
                    *END CODE
0283 0000           BOUNCE EQU 0
0284 0001           SAMPLE EQU 1
0285 0002           STRAIT EQU 2
0286 0003           YBOUNC EQU 3
                    *PATH TYPES

0287 0000           STILL  EQU 0
0288 0001           WIGGLE EQU 1
0289 0002           GLIDE  EQU 2
                    *IMAGE TYPES

                    **
                    **
                    **PATH STRUCTURE
                    **
                    **
0290 0000           PTYPE  EQU 0
                    *THE REST DEPENDS ON WHAT
                    *TYPE OF PATH

0291 3256 00        BPATH0 FCB BOUNCE
0292 3257 FE01             FCB -2,1

0293 3259 00        BPATH1 FCB BOUNCE
```

```
Ø294 325A Ø1FF             FCB 1,-1         INITIAL DIR

Ø295 325C ØØ        BPATH2 FCB BOUNCE
Ø296 325D Ø1FF             FCB 1,-1

Ø297 325F ØØ        BPATH3 FCB BOUNCE
Ø298 326Ø Ø1FF             FCB 1,-1

Ø299 3262 ØØ        BPATH4 FCB BOUNCE
Ø3ØØ 3263 Ø5FE             FCB 5,-2

Ø3Ø1 3265 ØØ        BPATH5 FCB BOUNCE
Ø3Ø2 3266 Ø1FF             FCB 1,-1

Ø3Ø3 3268 ØØ        BPATH6 FCB BOUNCE
Ø3Ø4 3269 Ø3Ø1             FCB 3,1

Ø3Ø5 326B ØØ        BPATH7 FCB BOUNCE
Ø3Ø6 326C Ø1ØØ             FCB 1,Ø

Ø3Ø7 326E ØØ        BPATH8 FCB BOUNCE
Ø3Ø8 326F FFØ1             FCB -1,1

Ø3Ø9 3271 ØØ        BPATH9 FCB BOUNCE
Ø31Ø 3272 FFØ1             FCB -1,1

Ø311 3274 Ø3        YBOUN1 FCB YBOUNC
Ø312 3275 FEØ1             FCB -2,1

Ø313 3277 Ø3        YBOUN2 FCB YBOUNC
Ø314 3278 FFØ1             FCB -1,1

Ø315 327A Ø3        YBOUN3 FCB YBOUNC
Ø316 327B FFØ1             FCB -1,1

                    **
                    **IMAGE STRUCTURE
                    **
Ø317 ØØØØ           ITYPE  EQU Ø
                    *STILL, GLIDE, OR WIGGLE
Ø318 ØØØ1           IXDIM  EQU 1
                    *IN BYTES - 4 TIMES PIXELS
Ø319 ØØØ2           IYDIM  EQU 2
                    ** OF LINES - IN PIXELS
Ø32Ø ØØØ3           ILENG  EQU 3
                    *ALWAYS - IXDIM*IYDIM
Ø321 ØØØ5           IDATA  EQU 5
                    *START OF RASTER(S)


Ø322 327D Ø2        BALL   FCB GLIDE
Ø323 327E Ø3Ø5             FCB 3,5          DIMENSIONS
Ø324 328Ø ØØØF             FDB 15           LENGTH IN BYTES
Ø325 3282 ØØFFØØ           FCB Ø,$FF,Ø
Ø326 3285 Ø3FFCØ           FCB 3,$FF,$CØ
Ø327 3288 Ø3FFCØ           FCB 3,$FF,$CØ
Ø328 328B Ø3FFCØ           FCB 3,$FF,$CØ
Ø329 328E ØØFFØØ           FCB $Ø,$FF,Ø

Ø33Ø 3291                  RMB 15*3

Ø331 32BE Ø2        GUPPY  FCB GLIDE
Ø332 32BF Ø3Ø6             FCB 3,6
Ø333 32C1 ØØ12             FDB 18

Ø334 32C3 ØØAØØØ           FCB $ØØ,$AØ,$ØØ
Ø335 32C6 Ø2ØØØØ           FCB $Ø2,$ØØ,$ØØ
Ø336 32C9 2A8AØØ           FCB $2A,$8A,$ØØ
Ø337 32CC B56ØØØ           FCB $B5,$6Ø,$ØØ
Ø338 32CF 2A88ØØ           FCB $2A,$88,$ØØ
Ø339 32D2 ØØØ28Ø           FCB $ØØ,$Ø2,$8Ø

Ø34Ø 32D5                  RMB 3*18
                    *SPACE FOR SHIFTED IMAGES

                    **
                    **  CHARACTER STRUCTURE
                    **
Ø341 ØØØØ           CXPOS  EQU Ø
Ø342 ØØØ1           CYPOS  EQU 1
Ø343 ØØØ2           CPATH  EQU 2
Ø344 ØØØ4           CIMAGE EQU 4
Ø345 ØØØ6           CNEXT  EQU 6
                    *OFFSETS WITHIN CHARACTER
                    *STRUCTURE
Ø346 ØØØ8           CHARLN EQU 8
                    *LENGTH OF CHARACTER STRUCTURE

Ø347 33ØB           ANLIST
Ø348 33ØB 1621             FCB 22,33
Ø349 33ØD 3274             FDB YBOUN1
Ø35Ø 33ØF 32BE             FDB GUPPY
Ø351 3311 3313             FDB ANLIST+CHARLN
```

```
Ø352 3313 1A19             FCB 26,25
Ø353 3315 3277             FDB YBOUN2
Ø354 3317 32BE             FDB GUPPY
Ø355 3319 331B             FDB ANLIST+2*CHARLN

Ø356 331B 1716             FCB 23,22
Ø357 331D 327A             FDB YBOUN3

Ø358 331F 32BE             FDB GUPPY
Ø359 3321 3323             FDB ANLIST+3*CHARLN

Ø36Ø 3323 2A3Ø             FCB XMAX/3,YMAX/2
Ø361 3325 3256             FDB BPATHØ
Ø362 3327 327D             FDB BALL
Ø363 3329 332B             FDB ANLIST+4*CHARLN

Ø364 332B 4Ø2Ø             FCB XMAX/2,YMAX/3
Ø365 332D 3259             FDB BPATH1
Ø366 332F 327D             FDB BALL
Ø367 3331 3333             FDB ANLIST+5*CHARLN

Ø368 3333 4Ø3Ø             FCB XMAX/2,YMAX/2
Ø369 3335 325C             FDB BPATH2
Ø37Ø 3337 327D             FDB BALL
Ø371 3339 333B             FDB ANLIST+6*CHARLN

Ø372 333B 2A2Ø             FCB XMAX/3,YMAX/3
Ø373 333D 325F             FDB BPATH3
Ø374 333F 327D             FDB BALL
Ø375 3341 3343             FDB ANLIST+7*CHARLN

Ø376 3343 4Ø18             FCB XMAX/2,YMAX/4
Ø377 3345 3262             FDB BPATH4
Ø378 3347 327D             FDB BALL
Ø379 3349 334B             FDB ANLIST+8*CHARLN

Ø38Ø 334B 2Ø3Ø             FCB XMAX/4,YMAX/2
Ø381 334D 3265             FDB BPATH5
Ø382 334F 327D             FDB BALL
Ø383 3351 3353             FDB ANLIST+9*CHARLN

Ø384 3353 2A18             FCB XMAX/3,YMAX/4
Ø385 3355 3268             FDB BPATH6
Ø386 3357 327D             FDB BALL
Ø387 3359 335B             FDB ANLIST+1Ø*CHARLN

Ø388 335B 2Ø2Ø             FCB XMAX/4,YMAX/3
Ø389 335D 326B             FDB BPATH7
Ø39Ø 335F 327D             FDB BALL
Ø391 3361 3363             FDB ANLIST+11*CHARLN

Ø392 3363 2Ø18             FCB XMAX/4,YMAX/4
Ø393 3365 326E             FDB BPATH8
Ø394 3367 327D             FDB BALL
Ø395 3369 336B             FDB ANLIST+12*CHARLN

Ø396 336B 4Ø13             FCB XMAX/2,YMAX/5
Ø397 336D 3271             FDB BPATH9
Ø398 336F 327D             FDB BALL
Ø399 3371 ØØØØ             FDB Ø            END OF LIST



Ø4ØØ 3373           ZENDDA
                    *END DATA

Ø4Ø1 3373                  END
```

Listing 2: FISH

```
1Ø CLEAR 5Ø,&H3ØØØ
2Ø LOADM"F
3Ø PMODE1,1
4Ø PCLS(1)
5Ø DEFUSRØ = &H3Ø1Ø
6Ø SCREEN 1,1
7Ø A=USRØ(Ø)
8Ø SCREEN Ø,1
9Ø INPUT "AGAIN";A$
1ØØ IF A$ = "Y" THEN 6Ø
```

16K
ECB

OPERATION FREEDOM

By Steve Britton IV

"This is the only message we have received from them," your superior says as he hands you the interstellargram. "The lives of almost 80 of the Earth's most brilliant scientists are at stake! That is why I have decided to send you, our best agent, to free as many of those people as you can, before it is too late.."

With these words ringing in your ears you now stand at the beginning of what seems to be a deserted lunar street. This is only one of the many that run throughout Luna Alpha, the first lunar colony of its kind, population: 80.

Operation Freedom will run on any 16K CoCo with Extended Color BASIC. The object of the game is to free as many captives as possible and safely escape without being killed by terrorist laser fire.

The playing screen is divided into three views. The largest is a map showing your position in Luna Alpha. At the bottom of the screen toward the middle is a side view of you as you run along the lunar streets. The smallest view, in the lower left corner of the screen, is the view of your laser weapon's targeting system.

You may move around the streets (yellow lines) of the map using the right joystick. Your position is indicated by the blue dot. The small red areas are the locations of captives being held prisoner by the terrorists. To free these captives, move up to an area that you have not been to yet and watch for the prison bars to appear on the side view and press the firebutton. The number of captives you have freed from this area is indicated under the word "FREED" located at the top right corner of the screen. The number of captives in an area will vary from one to six.

As you move around the deserted streets of Luna Alpha, you will encounter terrorists quite frequently. When a terrorist has been encountered, a red square will appear in the view of your targeting system. This square represents the relative position of the terrorist.

Notice the two rows of green dots running up and down and left to right along the two sides of this view. These are aim indicators. To aim at the terrorist, use the right joystick. Moving the joystick left and right controls the horizontal indicators, while moving the joystick up and down controls the vertical indicators. Align the blinking indicators up so when you fire, the point where the indicators intersect is on top of the red square.

On paper this sounds quite complex, but when you actually play a few times it becomes quite simple. Just hitting the red square will not kill the terrorist because the terrorists have developed a special armor. Keep firing, though, until you hit a weak spot. However, don't concentrate too deeply on killing the terrorist because he is firing at you, too! He will either shoot at your head or at your feet. When a terrorist fires, you will see the shot coming toward you (on the side view) at a speed determined by the skill level chosen at the beginning of the game (one is easy and five is hard). To dodge this shot, push the right joystick all the way up and press the firebutton to jump. Push the right joystick all the way down and press the firebutton to duck. Be sure to hold the firebutton down until the shot has safely passed.

All of this running and shooting can drain one's strength and ammunition supply rather quickly. Your strength and ammo are indicated by red lines at the lower right of the screen. When the red line under strength reads zero, you will die of fatigue. When your ammo reaches zero, you will not be able to shoot at terrorists; you will simply have to dodge any shots fired until the terrorist goes away. They usually only take a few shots and retreat anyway, so it is relatively easy to survive without ammunition.

The green areas on the map are the secret hideaways inhabited by the scientists who managed to escape the terrorists. Stop at these places to rest and reload your laser weapon with energy by moving up beside them until a green figure appears in front of you in the side view. When this occurs, hit the firebutton and your strength and ammo are now ready for combat once again.

Free as many captives as you can, being careful not to use too much strength in the process and stop to restore your essentials until you reach your destination; the starbase where an escape ship awaits to "beam you up" (located at the top left of the map and represented by a green star-like shape).

When you successfully complete Phase one hy reaching the escape ship, you then start over on Phase two. After each successful completion of a phase, 1,000 bonus points are awarded. The higher the phase, the harder the game. For each phase you gain, the difficulty factor will increase by one until it reaches five (the highest skill level). For beginners, I would recommend starting on Phase one or two to get used to the controls. Intermediates will find challenge on Level three, while experts will find levels four and five quite difficult to master.

Good luck!

The listing: FREEDOM

```
0 CLS0:C$=CHR$(128):GOTO40000
1 PH=1:CLS0:PRINT@45,"PHASE 1";:
FORX=1TO500:SCREEN0,1:NEXTX
2 POKE140,150
99 PMODE1,1:PCLS3
100 DRAW"BM16,3C2R2D2R2L6BM9,8R2
F2R8F2H2L4D2R2L2D4R2D2R2F2D2BM14
,18D4L2BM16,8R2"
101 DRAW"BM38,3R2D2R2L6BM31,8R2F
2R8D2L4D4R2D2R4D2BM36,18D4L2BM38
,8R2"
102 DRAW"BM56,1R2D2R2L6BM54,8E2R
6F2H2L2D2L2D4L2D2R6D2"
103 DRAW"BM82,9R2D2R2L6BM80,14R6
D2L4D2L2H2F2D4L2R2U4R6D4R4"
104 DRAW"BM100,6R2F2G2L2U2D6U4R2
F2D2BM108,14R1BM112,6D8BM116,14R
1BM120,6D8U4R2E2H2L2BM100,20D2R1
4L2U2L4BM124,22L6R2U2R2"
105 DRAW"BM135,10C4R18D2L18R4D8L
4R18D2L18U2R8U8R4D8R4U8"
106 DRAW"BM162,7C1R4D2L4R2D4R2F2
H2L4H2F2R2D6R2D2R2D2BM160,22U2R2
U2"
110 DIMR1(1,2),R2(2,2),J(1,2),D(
1,2),K(2,2),P(1,2),H(1,2),IK(1,2
),IS(1,2),ZT(7,7)
120 GET(9,3)-(22,23),R1:GET(30,2
)-(45,23),R2:GET(54,1)-(65,16),J
:GET(76,9)-(90,22),D:GET(100,6)-
(125,22),K:GET(135,10)-(155,22),
P:GET(160,7)-(169,22),H
121 PCLS3:COLOR4,3:LINE(5,3)-(19
7,117),PSET,B:LINE(5,135)-(53,17
7),PSET,B:LINE(68,135)-(165,176)
,PSET,B
122 DRAW"BM184,106C1R2L4D2R4F2H2
U2E2G2L4H2F2D2G2E2U2R4U1C2U29L40
D26L38U26L42D26L36U50R40D10R38U3
0R48D18R20U40L76D24L58U18L24U8C1
R2L4U2R4E2G2L4H2F2D2G2E2R4F2H2BM
183,133C2R6L6D4R6D4L6BM193,133R4
L2D8BM201,133R4D6F2H2U2L4D4U8BM2
11,133R6L6D4R2L2D4R6"
123 DRAW"BM168,79C2D8C4R2L4D2R4B
M145,76C1L2U2R2BM145,88C2L8C4U2L
2D4R2BM106,99C1L2U2D4L2U4BM84,79
C2D8C4R2D2L4BM65,80C1L2U2L2D4R2B
M47,100C2U8C4R2U2L4D2BM28,96C1L2
U2L2D4R2BM221,133C2D8U8F4R2D4U8B
M231,133R6D2U2L6D8R6U2R2L4BM243,
133R4L2D8BM251,133D8U4R4U4D8"
124 DRAW"BM31,64C2R8C4U2R2D4L2BM
52,55C2D8C4R2D2L4U2BM73,56C1R2U2
R2D4L2BM108,47C2R10D4C4R2D2L4U2B
M127,31C2U8C4R2U2L4D2BM132,35C1D
2R2D2L4U2BM174,35C2L6U6C4R2U2L4D
2BM151,12C2D10C4R2D2L4U2BM178,12
C1R2U2R2D4L2"
125 DRAW"BM98,19C2L6C4U2L2D4R2BM
71,32C2U8L8C4U2L2D4R2BM51,36C2D6
C4R2D2L4U2BM45,20C1R2U2R2D4L2BM2
3,16C2D20C4R2D2L4U2BM205,2C2R6L6
D8U4R2BM215,2R4D6F2H2U2L4D4U8BM2
25,2R6L6D4R2L2D4R6BM235,2R6L6D4R
2L2D4R6BM245,2R4F2D4G2L4U8"
126 DRAW"BM183,156C2G2D6U4R4U2D6
BM189,156D8U8R2D2R6U2L2D2G2E2R4D
6BM203,156D8U8R2D2R6U2L2D2G2E2R4
D6BM219,156G2D4F2E2U4H2BM103,16C
1R2U2R2D4L2":FORX=11TO47STEP4
127 PSET(X,137,1):NEXTX
128 FORY=139TO175STEP4
129 PSET(9,Y,1):NEXTY
130 COLOR2,3
131 IFPH>1ANDLQ<7THENLQ=LQ+1:GOT
O135
132 CLS:INPUT"LEVEL <1-5>";LQ:IF
LQ>5ORLQ<1THEN132ELSELQ=LQ+2
135 PRESET(X,Y):PLAY"V31O3T255":
SCREEN1,0
140 X=186:Y=100:C=222:OO=205:PP=
25:V=222:W=0
150 S=20:A=20
151 COLOR4,3:LINE(182,146)-(220,
148),PSET,B:LINE(174,168)-(220,1
70),PSET,B:COLOR2,3
152 PRESET(X,Y)
155 J=JOYSTK(0):J1=JOYSTK(1):IFY
=11THENY=10
156 IFX=18AND Y=10THEN20000ELSEI
FX=17AND Y=10THEN20000
157 PUT(139,150)-(150,170),R1:EX
EC43359:IFRND(11)=1THEN8000
158 IFPPOINT(X+2,Y)=1ORPPOINT(X-
2,Y)=1ORPPOINT(X,Y+2)=1ORPPOINT(
X,Y-2)=1THENRH=1:PUT(111,155)-(1
20,170),H:ELSEIFRH=1THENLINE(104
,155)-(120,170),PRESET,BF:RH=0
159 IFPPOINT(X+2,Y)=4ORPPOINT(X-
2,Y)=4ORPPOINT(X,Y+2)=4ORPPOINT(
X,Y-2)=4THENPC=1:PUT(111,155)-(1
31,167),P ELSEIFPC=1THENLINE(111
,155)-(131,167),PRESET,BF:PC=0
160 IFJ=63ORJ=0ORJ1=0ORJ1=63THEN
500
165 IFTIMER>=100THEN600ELSEPUT(1
39,150)-(152,170),R2:EXEC43359:F
ORQ=1TO35:NEXTQ:LINE(139,150)-(1
56,170),PRESET,BF
170 PE=PEEK(65280):IFPE=126ORPE=
254THEN7000ELSE155
500 IFRH=1ANDPEEK(65280)=126ORPE
EK(65280)=254THEN7000ELSEIFJ=63T
HENIFPPOINT(X+2,Y)=2THENPSET(X,Y
):X=X+2:PRESET(X,Y):GOTO165
505 IFJ=0THENIFPPOINT(X-2,Y)=2TH
ENPSET(X,Y):X=X-2:PRESET(X,Y):GO
TO165
510 IFJ1=63THENIFPPOINT(X,Y+2)=2
THENPSET(X,Y):Y=Y+2:PRESET(X,Y):
GOTO165
515 IFJ1=0THENIFPPOINT(X,Y-2)=2T
HENPSET(X,Y):Y=Y-2:PRESET(X,Y):G
OTO165
520 GOTO165
600 TIMER=0:S=S-1:C=C-2:PSET(C,1
46,2):PSET(C,148,2):IFC=182THEN5
000ELSEGOTO165
700 Q=RND(5):FD=FD+Q:FORZ=1TOQ:S
OUND(Z*30),1:LINE(OO,PP)-(OO+3,P
P+2),PSET,B:OO=OO+8:IFOO>=245THE
NPP=PP+6:OO=205
701 IFPPOINT(X+2,Y)=4THENPSET(X+
2,Y,3):PSET(X+3,Y,3)
702 IFPPOINT(X-2,Y)=4THENPSET(X-
2,Y,3):PSET(X-3,Y,3)
703 IFPPOINT(X,Y+2)=4THENPSET(X,
Y+2,3):PSET(X,Y+3,3)
704 IFPPOINT(X,Y-2)=4THENPSET(X,
Y-2,3):PSET(X,Y-3,3)
705 NEXTZ:SC=SC+(7*Q):GOTO155
800 V=V-2:PSET(V,168,2):PSET(V,1
70,2):RETURN
1000 XT=RND(30)+11:YT=RND(33)+13
8
1005 LINE(XT,YT)-(XT+3,YT+3),PSE
T,BF
1010 AZ=1
1015 GOTO170
5000 LINE(139,150)-(150,170),PRE
SET,BF:PUT(131,154)-(156,170),K:
FORX=1TO15:PLAY"T25CCA":NEXTX
5001 CLS4:PRINT@102,"may"C$"you"
C$"rest"C$"in"C$"peace";
5002 FORE=128TO480STEP32:PRINT@E
,STRING$(31," ");:NEXT
5003 PRINT@224,"YOU FREED "FD" P
EOPLE ...";
5004 PRINT@320,"AND KILLED "XI"
IMPERIALS ...";
5005 PRINT@384,"FOR A TOTAL SCOR
E OF "SC;
5006 IFINKEY$=""THEN5006
5007 GOTO121
6000 INPUTX,Y:PSET(X,Y,1):SCREEN
1,0:FORX=1TO200:NEXTX:GOTO6000
6999 RH=1
7000 IFRH=0ANDPC=0THEN155ELSEIFP
C=1THEN700ELSEIFRH=1THEN FOR C=1
82 TO220:PSET(C,146,4):PSET(C,14
8,4):PLAYSTR$(INT((224-C)/4)):NE
XTC:S=20:C=222
7005 FORV=174TO220:PSET(V,168,4)
:PSET(V,170,4):PLAYSTR$(INT((224
-V)/4)):NEXTV:V=222
7010 IFPPOINT(X+2,Y)=1THENPSET(X
+2,Y,3):PSET(X+3,Y,3)
7015 IFPPOINT(X-2,Y)=1THENPSET(X
-2,Y,3):PSET(X-3,Y,3)
7020 IFPPOINT(X,Y+2)=1THENPSET(X
,Y+2,3):PSET(X,Y+3,3)
7025 IFPPOINT(X,Y-2)=1THENPSET(X
,Y-2,3):PSET(X,Y-3,3)
7030 GOTO155
8000 EX=RND(19)+15:EY=RND(16)+14
3
8005 LINE(70,137)-(129,170),PRES
ET,BF
8010 COLOR4,3:LINE(EX+2,EY+1)-(E
X,EY+5),PSET,BF:COLOR2,3:J=JOYST
K(0):J1=JOYSTK(1)
8011 IFJ<6THENJ=6
8012 IFJ1<6THENJ1=6
8015 J=INT(J/6):WX=(J*4)+6:J1=IN
T(J1/6):WY=(J1*4)+134
8020 PSET(WX,137,4):PSET(9,WY,4)
8025 Q=PEEK(65280):IFQ=126ORQ=25
4THENPK=PPOINT(WX,WY):IFV<176THE
N8050ELSEGOSUB800:LINE(WX,137)-(
WX,WY),PSET:LINE(134,157)-(73,15
7),PSET:LINE(9,WY)-(WX,WY),PSET:
PLAY"ABABABAB":IFPK=4THEN8030ELS
ELINE(WX,137)-(WX,WY),PRESET:LIN
E(9,WY)-(WX,WY),PRESET
8026 LINE(73,157)-(134,157),PRES
ET:GOTO8050
8027 GOTO8050
8030 XI=XI+1:COLOR4,3:LINE(EX-4,
EY)-(EX+10,EY+10),PSET,BF:COLOR2
,3:FORQ=30TO100STEP5:PLAY"T"+STR
$(Q)+"AB":NEXTQ:LINE(EX-4,EY)-(E
X+10,EY+10),PRESET,BF:SC=SC+300:
LINE(WX,137)-(WX,WY),PRESET:LINE
(9,WY)-(WX,WY),PRESET:PSET(WX,13
7,1):PSET(9,WY,1)
8031 LINE(73,157)-(134,157),PRES
ET:GOTO155
8050 PSET(WX,137,1):PSET(9,WY,1)
:IFRND(16)<=(LQ-2) THEN8060ELSE8
010
8060 IO=IO+1:IFIO=4THENIO=0:LINE
(EX-4,EY)-(EX+10,EY+10),PRESET,B
F:GOTO155ELSEZZ=1:Q=70:PSET(EX+2
,EY+3,4):PLAY"ABABCDCD":PSET(EX+
2,EY+3,1):IFRND(2)=1THEN9000
8065 Q=Q+ZZ:ZZ=ZZ+LQ
8066 EXEC43359
8067 J=JOYSTK(0):J1=JOYSTK(1)
8070 IFPPOINT(146,150)=2ANDQ>=14
6THEN5000ELSEPSET(Q,150,4):DV=PE
EK(65280):IF(DV=126ORDV=254)ANDJ
1=63THENPUT(139,157)-(150,170),D
:LINE(139,150)-(150,155),PRESET,
BF ELSEPUT(139,150)-(150,170),R1
```

```
8075 IFQ<=149THEN8065
8080 LINE(71,150)-(150,150),PRES
ET:PLAY"CDCDABAB":LINE(139,150)-
(150,170),PRESET,BF:PUT(139,150)
-(150,170),R1:LINE(156,151)-(162
,151),PRESET:LINE(167,151)-(180,
151),PRESET:GOTO8050
9000 Q=Q+ZZ:ZZ=ZZ+LQ
9001 EXEC43359
9005 J=JOYSTK(0):J1=JOYSTK(1)
9010 IFPPOINT(146,166)=2ANDQ>=14
6THEN5000ELSEPSET(Q,166,4):DV=PE
EK(65280):IF(DV=126ORDV=254)ANDJ
1=0THENPUT(139,150)-(150,165),J:
LINE(139,167)-(150,170),PRESET,B
F ELSEPUT(139,150)-(150,170),R1
9020 IFQ<=149THEN9000
9030 LINE(71,166)-(150,166),PRES
ET:PLAY"CDCDABAB":LINE(139,150)-
(150,170),PRESET,BF:PUT(139,150)
-(150,170),R1:LINE(156,167)-(162
,167),PRESET:LINE(167,167)-(180,
167),PRESET:GOTO8050
10000 CLS0
10001 PRINT@136,"";
10002 POKE140,245
10005 A$="operation":GOSUB10030
10010 A$="freedom"
10015 POKE1169,58:EXEC43359
10016 PRINT@146,"";
10020 GOSUB10030
10021 FORX=1TO100:NEXTX:POKE1198
,45:EXEC43359:PRINT@175,"";:A$="
by":GOSUB10030
10022 POKE1201,45:EXEC43359:PRIN
T@202,"";:A$="steve":GOSUB10030
10023 PRINT@208,"";:A$="britton"
:GOSUB10030
10025 FORX=1TO800:NEXTX:GOTO1
10030 FORX=1TOLEN(A$)
10031 POKE140,150
10035 B$=MID$(A$,X,1):PRINTB$;:E
XEC43359:FORZ=1TO RND(50):NEXTZ:
NEXTX
10040 RETURN
10045 NEXTKQ
10050 DRAW"BM88,4D7U3R4U4D7BM94,
8D3F1R3BM95,8R2D2L2BM100,6D6BM10
3,8D7U5R2E1U1H1L2BM116,4D4BM116,
11M116,11BM120,4D4BM120,11M120,1
1BM123,4D4BM123,11M123,11"
10060 FORKQ=1TO200:NEXTKQ:LINE(8
8,4)-(123,15),PRESET,BF
10070 GOTO10020
20000 C$=CHR$(128):PMODE1,1:PCLS
3:SCREEN1,0:COLOR2,3:DRAW"BM0,12
0C4R4L2M8,99R66M80,120R2L4BM75,9
9E8U12H8L66G8D12F8":PAINT(36,88)
,4,4:PSET(20,180,2):DRAW"BM16,18
3C2F2R4E2M20,187G2D2BM23,190U2":
GET(16,180)-(24,190),IK:LINE(16,
180)-(24,190),PRESET,BF
20010 FORJZ=1TO10:PRESET(RND(80)
,RND(25)+71):NEXTJZ
20020 FORX=1TOINT(FD/5)
20025 FORZ=250TO40STEP-20
20030 GET(Z,111)-(Z+10,120),IS:P
UT(Z,111)-(Z+10,120),IK:EXEC4335
9:FORA=1TO50:NEXTA:PUT(Z,111)-(Z
+10,120),IS
20040 NEXTZ:PUT(40,111)-(50,120)
,IK:LINE(40,99)-(50,120),PSET,BF
:FORT=50TO100STEP10:PLAY"T"+STR$
(T)+"ABABABA":NEXTT:LINE(50,120)
-(40,99),PRESET,BF:NEXTX
20050 DRAW"BM40,100C2D2R6U2L4D6R
2U4C4BM39,100D6R2U2BM44,108L2D2R
2BM48,100D6L2U2":GET(0,71)-(83,1
20),ZT
20060 PLAY"O1T1;4;6;5;1;T1;3"
20070 FORY=71TO0STEP-1:PCLS3:W=W
+2:PUT(0,Y)-(83,Y+49),ZT:PLAY"T"
+STR$(W)+"O2ABCD":NEXTY
20080 CLS0:PRINT"you"C$"have"C$"
escaped";:POKE1024+17,33:POKE102
4+18,33
20090 FORE=128TO480STEP32:PRINT@
E,STRING$(31," ");:NEXT
20095 SC=SC+1000
30000 PRINT@224,"YOU FREED "FD"
PEOPLE ...";
30010 PRINT@320,"AND KILLED "XI"
 IMPERIALS ...";
30020 PRINT@384,"FOR A TOTAL SCO
RE OF "SC;
30030 IFINKEY$=""THEN30030
30031 PH=PH+1:CLS0:PRINT@45,"PHA
SE ";PH;:FORX=1TO500:SCREEN0,1:N
EXTX:GOTO121
40000 CLS:A$="TO:THE FEDERATION
OF EARTH":PRINT@0,"";:GOSUB10030
:A$="FROM:PEOPLE´S REVOLUTION FO
R":PRINT@64,"";:GOSUB10030:A$="A
NARCHY":PRINT@96,"";:GOSUB10030:
A$="DATELINE:LUNA ALPHA":PRINT@1
35,"";:GOSUB10030
40009 FORX=1TO200:NEXTX
40010 A$="WE HAVE TAKEN LUNA ALP
HA":PRINT@192,"";:GOSUB10030:PRI
NT@218,"-STOP-";:SOUND200,1
40014 FORX=1TO100:NEXTX
40015 A$="WE HAVE PRISONERS":PRI
NT@256,"";:GOSUB10030:PRINT@282,
"-STOP-":SOUND200,1
40019 FORX=1TO100:NEXTX
40020 A$="THEY START DYING IN 8
HRS.":PRINT@320,"";:GOSUB10030:P
RINT@346,"-END-":SOUND200,1
40030 FORX=1TO500:NEXTX:GOTO1000
0
```

THE QUEST FOR THE FALCONS' LAIR

by Aron Fransen

In the year of our Lord 1991, several of Earth's deep space probes encounter several intelligent forms of life. Not all are peaceful.

One of theses races (through the probes' information) has found its way to Earth and is now proceeding to overtake it.

Several of their "motherships" encircled the Earth and surrounded it with a blue shield of energy, with the motherships themselves acting as the charging bases, and in the process leaving them open without defences. However, each mothership has one falcon ship, a smaller and more versatile ship capable of firing back and blocking your laser.

You will be controlling three successive attack bases, capable of firing at the falcon ship and its base. Caution: Do not fire when the blue bar is directly below your own base, or you will be destroyed! To fire, simply press the firebutton on the right joystick, which controls your base.

There are nine levels of play, indicated by the blue bars at the bottom of the screen, one being the easiest and nine the most difficult.

You will receive three bases, indicated by the red bars at the bottom of the screen (emanating from the right). Your score is indicated by a red line at the bottom (emanating from the left).

What you must do is shoot through the defence of the falcon ship's base (the Falcon's Lair) so you may then destroy the Lair's power cell. Once it is destroyed the entire ship falls to the ground, explodes and you are transported to another Lair's area.

The more of the shield you destroy before destroying the power cell, the higher your score will be at the end of the round. This is not as easy as it may seem. As each level progresses, the shield starts to rebuild itself at speeds according to the level of play.

The listing is rather long, but hopefully you'll find the game to be worth it.

Instructions

1) Load game

2) RUN it (you may have to run it twice because of a bug in BASIC).

3) Press Reset (and RUN) until screen is red.

4) After the title page comes up and the introduction sounds are played, press fire.

5) Adjust the right joystick to select difficulty, then press the fire button.

6) After a few seconds the game will begin. The object is to destroy the blue power cell at the centre of the Falcon's Lair (at the top of the screen), while at the same time either destroying or avoiding the small falcon ship which hovers just below the Falcon's Lair. Do not fire when the blue bar (the one that moves about) is directly below you, or you will be destroyed.

7) After you have lost all three bases, the game ends, showing you your final score and asking if you would like to try again. Your selection is made by adjusting the right joystick until the white box is around the appropriate answer; then press fire.

The listing: FLCNLAIR

```
1 PCLEAR7:PMODE4,1:SCREEN1,1:PMO
DE3:PCLS3
2 I$=INKEY$:IFI$=""THEN2
3 CLS:DIM X,Y,EX,EY,PX,T,R,LEV,M
EN,A(10),E(656),E1(50),E2(50),Y(
16),P(31),D$(61),D(50),N$(9):ER$
="R8DL8DR8DL8DR8DL8BU5"
4 POKE187,0:FORT=0TO16:READ Y(T)
:NEXT:FORT=0TO26:READ D$(T):NEXT
:FORT=0TO9:READ N$(T):NEXT
5 PMODE4,4:PCLS:PMODE4,1:PCLS1:S
CREEN1,1:I$="THE":DRAW"C0S8BM100
,20":GOSUB79:I$="FALCONS":DRAW"S
16BM14,60":GOSUB81:I$="LAIR":DRA
W"BM60,90":GOSUB81
6 I$="BY@AARON@MARTIN@FRANSEN":D
RAW"S4BM30,140":GOSUB79:FORT=0TO
7:GET(25,145-T)-(210,145-T),E,G:
PUT(25+T,145-T)-(210+T,145-T),E,
PSET:NEXT
7 PMODE3:COLOR3:LINE(0,0)-(255,1
91),PSET,B:COLOR2:LINE(4,4)-(251
,187),PSET,B:PMODE4:PLAY"L20T8BA
GFEDCBAGFEDCBAGFEDCAGFEDCAGFEDCA
GFEDCGFEDCGFEDCGFEDCFEDCFEDCFEDC
EDCEDCEDCDCDCDCCCCC
8 P=PEEK(65280):IFP=126ORP=254TH
EN9ELSE8
9 CLS:PRINT@226,"WITH THE JOYSTI
CK, CHOOSE A       LEVEL OF DIFFI
CULTY, THEN           PRESS THE FI
RE BUTTON":PLAY"P1
10 LEV=INT(JOYSTK(0)/7.4)+1:PRIN
T@202,"LEVEL =";LEV
11 P=PEEK(65280):IFP=126ORP=254T
```

```
HEN12ELSE10
12 CLS1:PRINT@227,"YOU WILL BEGI
N IN A MOMENT";:SCREEN0,1
13 MEN=3
14 PMODE4,2:PCLS:PMODE3:PCLS1
15 PMODE3:COLOR2:LINE(0,0)-(31,6
),PSET,BF:GET(0,0)-(31,6),P:PCLS
1
16 GET(0,0)-(80,20),E:PMODE3:COL
OR3:FORT=10TO20STEP2:LINE(T,4)-(
T+20,9),PSET:LINE(T+30,9)-(T+50,
4),PSET:LINE(T,34)-(T+20,29),PSE
T:LINE(T+30,29)-(T+50,34),PSET:N
EXTT:GET(0,0)-(80,20),E1:GET(0,2
0)-(80,40),E2:PCLS
17 PMODE3,4:FORT=40TO74:CIRCLE(1
28,144),T,3,.4:NEXTT:FORT=0TO20:
CIRCLE(128,144),T,2,.7:NEXTT:COL
OR2:LINE(0,140)-(70,134),PSET,BF
:LINE(255,140)-(185,134),PSET,BF
:LINE(0,148)-(70,154),PSET,BF:LI
NE(255,148)-(185,154),PSET,BF
18 PMODE3,1:LINE(0,12)-(15,22),P
RESET,BF:GET(0,12)-(15,22),E:DRA
W"BM8,12C3M0,21M15,21M7,12":PAIN
T(7,16),3,3:PMODE4:GET(0,12)-(15
,22),A:PUT(0,12)-(15,22),E
19 PCOPY7TO1:PLAY"L100T10002":PM
ODE4,2:SCREEN1,1:PMODE4,1:FORT=0
TO47:GET(0,T)-(255,T+32),E:PUT(0
,T+1)-(255,T+33),E:PLAY"FC":NEXT
T:PCOPY2TO1:PMODE4,2
20 RESTORE:FORT=0TO16:READ Y(T):
NEXTT
21 PMODE3,2:COLOR2:FORT=1TO(LEV*
28)STEP28:LINE(T,188)-(T+20,190)
,PSET,B:NEXTT:PMODE4,2
22 PMODE3,2:COLOR3:ON MEN GOSUB7
7,76,75
23 GET(0,33)-(255,46),E:SH=1:Y=1
60:X=120:R=RND(-TIMER)
24 J=JOYSTK(0):IFJ<15THENX=X-8EL
SEIFJ>48THENX=X+8
25 IFX<0ORX>240THENX=0
26 IFRND(10-LEV)=1THENPUT(PX,172
)-(PX+31,178),E:PX=RND(20)*8:PUT
(PX,172)-(PX+31,178),P
27 PUT(Q,160)-(Q+15,170),E:PUT(X
,160)-(X+15,170),A:P=PEEK(65280)
:IFP=126ORP=254THENGOSUB39
28 IFTIMER>1500-(LEV*100)THENTIM
ER=0:GOSUB57
29 IFRND(12-LEV)=1THENGOSUB37
30 GOSUB34
31 SH=-SH:IFSH=-1THENPUT(EX,EY)-
(EX+80,EY+20),E1 ELSEPUT(EX,EY)-
(EX+80,EY+20),E2
32 PMODE3,2:COLOR3:LINE(0,184)-(
SCR/100,185),PSET,B:PMODE4,2
```

```
33 Q=X:GOTO24
34 EX=EX+((RND(3)-2)*8):IFEX<0OR
EX>172THENEX=QX
35 EY=EY+((RND(3)-2)*2):IFEY<30T
HENEY=30ELSEIFEY>120THENEY=120
36 QX=EX:RETURN
37 LINE(EX+35,EY+10)-(EX+35,170)
,PSET:LINE(EX+45,EY+10)-(EX+45,1
70),PSET:PLAY"L20T4004BGEDC":IF(
EX+46>X AND EX+46<X+15)OR(EX+34>
X AND EX+34<X+15)THEN 59
38 LINE(EX+35,EY+10)-(EX+35,170)
,PRESET:LINE(EX+45,EY+10)-(EX+45
,170),PRESET:RETURN
39 IFX>=PX AND X<PX+20THEN59
40 IFX>EX AND X<EX+70THEN45
41 T=X/8:IFT<7ORT>23THENTY=0:GOS
UB74:RETURN
42 T=T-7:TY=Y(T):GOSUB74:IFY(T)=
0THEN44ELSEY(T)=Y(T)-4:LINE((T*8
)+60,Y(T))-((T*8)+67,Y(T)+8),PRE
SET,BF:SCR=SCR+50
43 IF(T>6AND T<10)AND Y(T)<10THE
N47
44 RETURN
45 SCR=SCR+20:TY=EY+5:GOSUB74:PL
AY"O1ABDGEFABDFE":PMODE4,2:FORT=
0TO18:PUT(EX+RND(80),EY+RND(20))
-(EX+RND(80),EY+RND(20)),E,NOT:N
EXTT46 PUT(EX,EY)-(EX+80,EY+20),
E:EX=RND(20)*8:EY=RND(100)+32:RE
TURN
47 PMODE4,2:R=1:PLAY"T2L2":GET(1
08,0)-(149,11),D,G:FORT=0TO77:PU
T(108+R,0)-(149+R,11),D,PSET:PLA
Y"ET+":IFR=1THENR=0ELSER=1
48 NEXTT:PCOPY6TO1:PMODE4,1:GET(
0,15)-(255,80),E:FORT=15TO67STEP
2:PUT(0,T)-(255,T+65),E:PLAY"L10
0T50C":NEXTT:PMODE4,2:FORT=20TO1
50STEP4:PUT(0,T)-(255,T+65),E
49 PLAY"L100T20C":NEXT:PMODE4,2:
FORT=0TO4:X=RND(128)+64:Y=191-RN
D(32):FORR=3TORND(20)+10:CIRCLE(
X,Y),R,1,.9:CIRCLE(X,Y),R-3,0,.9
:NEXTR:PLAY"L50T5001CEDCC":NEXTT
50 FORT=0TO30:PMODE3,2:SCREEN1,1
:PLAY"O3GEDC":PMODE4,2:SCREEN1,1
:PLAY"GEDC":NEXT
51 FORT=0TO16:INC=INC+((Y(T)-30)
*-.5):NEXTT:CLS:PRINT@233,"BONUS
 :":FORT=SCR TO SCR+INC:PRINT@24
0,T:PLAY"T8E":NEXTT:SCR=SCR+I
NC
52 SCREEN1,0:PCLS1:DRAW"C0BM80,4
0":I$="SCORE":GOSUB79:DRAW"BM128
,40":I$=STR$(SCR):GOSUB80:DRAW"B
M40,60":I$="ADVANCE@TO@NEXT@LEVE
L":GOSUB79
```

```
53 DRAW"BM36,80":I$="PRESS@FIRE@
TO@CONTINUE":GOSUB79:PLAY"L200T1
00"
54 C=RND(12):FORT=21TO1STEP-1:PL
AY"V"+STR$(T)+";"+STR$(C):P=PEEK
(65280):IFP=126ORP=254THEN55ELSE
NEXTT:GOTO54
55 PCLS:PLAY"V28":LEV=LEV+1:IFLE
V>9THENLEV=9
56 GOTO19
57 T=RND(16):IFY(T)>28THENRETURN
58 PMODE4,1:GET((T*8)+60,Y(T))-(
(T*8)+67,Y(T)+4),E:PMODE4,2:PUT(
(T*8)+60,Y(T))-((T*8)+67,Y(T)+4)
,E:Y(T)=Y(T)+4:GET(0,20)-(60,40)
,E:RETURN
59 PMODE4,2:PLAY"L10T8BGEDCAFDCG
EDCDCC":FORT=0TO3:Q=X+RND(40)-10
:W=Y+RND(20)-5:FORR=3TORND(10)+5
:CIRCLE(Q,W),R,1:CIRCLE(Q,W),R-3
,0:NEXTR:PLAY"L20T2001CEDC":NEXT
T
60 MEN=MEN-1:IFMEN<1THEN62
61 PMODE4,2:PCLS:GOTO18
62 PCOPY6TO1:PMODE4,1:GET(0,15)-
(255,80),E:FORT=15TO47STEP2:PUT(
0,T)-(255,T+65),E:PLAY"L100T50C"
:NEXTT:PMODE3,2:COLOR2:R=.8:X=0:
FORT=60TO191:LINE(128-X,T)-(128+
X,T),PSET:X=X+R:NEXTT
63 PLAY"L20T20CDEFGABO+CDEFGO-CD
EFABO+CDEFGO-CDEFGABO+CDEFGABO-C
DEFGABO+CDEFGO-DEFGABO+CDEFGO-DE
FGABO+CDEFGO-DEFGABO+CDEFGO-DEFG
ABO+CDEFGO-EFGABO+CDEFGO-EFGABO+
CDEFGO-EFGABO+CDEFGO-EFGABO+CDEF
GO-
64 PLAY"FGABO+CDEFGO-FGABO+CDEFG
O-FGABO+CDEFGO-FGABO+CDEFGO-GABO
+CDEFGO-GABO+CDEFGO-GABO+CDEFGO-
GABO+CDEFGO-ABO+CDEFGO-ABO+CDEFG
O-ABO+CDEFGO-ABO+CDEFGO-BO+CDEFG
O-BO+CDEFGO-BO+CDEFGO-BO+CDEFG
65 PLAY"CDEFGCDEFGCDEFGCDEFGDEFG
DEFGDEFGDEFGEFGEFGEFGEFGFGFGFGFG
GGGG
66 PMODE4,2:I$="THE@ENEMY@HAS@SU
CCEEDED":DRAW"BM32,4":GOSUB82:I$
="YOU@HAVE@FAILED":DRAW"BM64,50"
:GOSUB82
67 I$="YOU@WILL@RECEIVE@":DRAW"B
M48,120":GOSUB82:I$=RIGHT$(STR$(
SCR),LEN(STR$(SCR))-1):GOSUB80
68 I$="DOLLARS@FOR@YOUR@EFFORTS"
:DRAW"BM32,130":GOSUB82
69 I$="DO@YOU@WISH@TO@TRY@AGAIN"
:DRAW"BM32,150":GOSUB82:I$="@@YE
S@@@@@@@NO@@":DRAW"BM64,170":GOS
UB82
```

```
70 T=SGN(JOYSTK(0)-31):IFT=-1THE
NR=0ELSER=1
71 COLORR:LINE(132,168)-(196,178
),PSET,B:COLORR+1:LINE(124,168)-
(60,178),PSET,B
72 P=PEEK(65280):IFP=126ORP=254T
HEN73ELSE70
73 IFT=-1THENRUNELSECLS:PRINT"TH
IS IS THE END...":END
74 PMODE3,2:COLOR2:LINE(X+7,Y)-(
X+9,TY),PSET,B:PLAY"O2BGDC":LINE
(X+7,Y)-(X+9,TY),PRESET,B:RETURN
75 LINE(167,182)-(193,184),PSET,
B
76 LINE(197,182)-(223,184),PSET,
B
77 LINE(227,182)-(253,184),PSET,
B:RETURN
78 DATA16,24,24,28,28,28,32,32,3
2,32,32,28,28,28,24,24,16
79 FORT=1TOLEN(I$):Q$=MID$(I$,T,
1):R=ASC(Q$)-64:DRAWD$(R):NEXTT:
RETURN
80 FORT=1TOLEN(I$):Q=VAL(MID$(I$
,T,1)):DRAW N$(Q)+"BL7"+N$(Q)+"B
L":NEXTT:RETURN
81 FORT=1TOLEN(I$):Q$=MID$(I$,T,
1):R=ASC(Q$)-64:DRAWD$(R)+"S4BL2
9BUS16"+D$(R)+"S4BL29BUS16"+D$(R
)+"S4BL3BD2S16":NEXTT:RETURN
82 FORT=1TOLEN(I$):Q$=MID$(I$,T,
1):R=ASC(Q$)-64:DRAW"C0"+ER$+"C1
"+D$(R)+"BL7"+D$(R)+"BL":NEXTT:R
ETURN
83 DATA BR8,BRGD2NDR4NDU2HNL2BR5
,RNR3D4NLR3EHNLEBR3BU,BRR2NFL2GD
2FR2EBU3BR4
84 DATA RNR2D4NLR2EU2HBR5,NR4D2N
R3D2R4BR4BU4,NR4D2NR3D2BR8BU4,BR
1NR3GD2FR3ENL2BR3BU3,D2ND2R4ND2U
2BR4
85 DATA BR2NLNRD4NLRBR5BU4,BR4D3
GL2NHBR7BU4,D2ND2RNE2F2BR5BU4,D4
R4BR4BU4,ND4F2E2ND4BR4
86 DATA ND4F4U4BR4,NR4D4R4U4BR4,
ND4R3FGNL1BR5BU2,BRNR2GD2FR2ENFN
HU2BR4BU,"ND4R3FDGLM+2,+1BU4BR4"
87 DATA BR4L3GFR3FGL3BR7BU4,R2ND
4R2BR4,D3FR2EU3BR4,D2F2E2U2BR4,D
4E2F2U4BR4
88 DATA F4H2G2E4BR4,DF2NDE2UBR4,
R4G4R4BU4BR4
89 DATA BRGD2FR2EU2HNLBR5,BR2NGD
4NLRBR5BU4,BDER2FDL3GDR4BR4BU4
90 DATA BDER2FGFGL2NHBR7BU4,D2R4
ND2NU2R1BR3BU2,NR4D2R3FGL3BR8BU4
91 DATA BR4L3GD2FR3EHNL3BR4BU2,R
4G2D2BR6BU4,BRNR2GFGFR2EHEBR4BU1
,BR3NFL2GFR3NUDGL2BR7BU4
```

COLOUR CARTOONS

By Wayne Thume

Along with many others, I have always wanted to experiment with animation. Being and avid "Bugs Bunny" fan, I've often said to myself, "Hey! I'd like to make cartoons, too." Now with the help of my Color Computer and this program, I can create animation for games, graphics demonstrations or just for fun.

The first problem confronted in designing this program was space. I wanted the pictures for the animation to be held in the computer's memory and not on tape or disk. Since storing too many pictures would take up and incredible amount of space, I decided to reduce each picture to a 16-by-32 grid.

Now before you quit reading this article in disgust, let me assure you there is more than enough room for significant high resolution animation. Using this grid 96 pictures can be created, which will allow for about 20 seconds of animation. I'll be the first to admit that 20 seconds doesn't sound like very long, but after drawing 96 pictures you will be convinced that it is far too long.

The next problem was which language to write the program in. I chose BASIC so anyone could easily interpret and modify the program to suit his or her own tastes. Using GET and PUT, I was able to store and retrieve pictures at the high rate of speed that is needed to show the animation. The pictures are stored on pages five through eight of the graphics memory. They are displayed on the Hi-Res screen PMODE 4, which uses the first four pages of graphic memory. Using this small amount of memory allows for enough room to create a program to utilize the graphics for some different programming applications.

After the program is typed in and run, there will be a menu which allows you to either 1) create a picture, 2) see the animation, 3) save animation, 4) load animation or 5) quit the program. Choosing the first option will take you to the Hi-Res screen PMODE 4,1 where there is a flashing rectangle. The size of the rectangle represents the size of the area you may animate. The rectangle may be moved to any area of the screen with a joystick. Once you have chosen the area to animate, press the joystick firebutton and the area within the flashing rectangle will be enlarged and placed on the Lo-Res screen. You have several commands to create the picture at this point (these are all summarized in a chart at the end of this article). Once your picture is finished, it will be stored and you will again return to the menu.

The second option in the menu will display the animation that has been created. If you choose this option, the computer will switch to the Hi-Res mode and the first picture in the animation sequence will be shown. To start the animation, press the space bar. You may stop the animation at any point by pressing any key other than the space bar. To resume animation press the space bar again. To quit the display sequence before it has ended, press 'Q' and the program will return to the menu.

Option three will save the picture sequence on cassette tape or disk drive. If you choose this option, you will see what appears to be garbage forming at the top of the screen. This "garbage" is actually the screen locations for all 96 pictures that have been poked into the video memory. The program stores all of the information in two files, one for the screen locations of the pictures and one for the pictures themselves. Once you have given a filename and specified tape or disk, the computer will save the data and return to the menu.

With the fourth option you will be asked for the name of a file that has been previously saved. Once this is entered, merely specify whether you are using tape or disk drive and the computer will load the pictures and return to the menu.

The last option allows you to exit the program. The screen will clear and return to the BASIC mode.

There are many areas of the program that are not explained adequately, so I have included a sample animation program. Since an actual cartoon would be too long to create, I had to stick to a simple graphics demonstration. The animation is that of a circle travelling in sine wave moving downward. The circle is simultaneously passing back and forth across a vertical line in the center of the screen. It may not be the best example, but it does show how much greater speed can be achieved in your BASIC programs.

When the program is run, it will first display a line and the circle, then the program will GET the 16-by-32 area which includes the circle and part of the line. Next the computer clears the screen and redraws the picture with the circle moved to the new position. Slowly, each new position is drawn and recorded until the circle reaches the bottom. The reason it takes so long to draw this sequence is because the computer must draw and erase each picture before it can draw the next one.

After this sequence has been recorded the animation will be displayed. First, some colourful patterns are drawn on either side of the screen to demonstrate that the animation can be displayed on a screen without having to erase any picture which may already be there. Next, you will see the circle move down the screen - note the smooth and quick movement as the circle glides down the screen.

And now to explain the gimmick. Refer to the graphics picture above. There is no program included to draw this picture. The fact is, I drew this picture dot by dot with a graphics editor and it would be impossible to create a short program to reproduce it. When I create my animation, I display it on the little movie screen in the picture; this makes my feeble cartoons look much more impressive.

If you would like a copy of this screen, I would be glad to send one if you send me a tape or disk and enough money to pay for postage to send it back. I would also like to see any animation you create using my program. My address is Route 2, Box 119, Trappe, MD21673.

Summary of Commands for Option 1:
Create Picture

Key Function

- Arrow - will move dot around screen to desired position
- S - sets point on the screen
- R - resets point on the screen
- D - shows Hi-Res screen PMODE 4,1. Press any key and it will switch to PMODE 4,5 where the pictures are stored. Press any key again and the computer will return to the Lo-Res screen.
- G - gets any of the 96 pictures you choose and puts it on the screen
- C - clears screen
- X - switch mode. The first time this is pushed you will be given the option to choose the (S)et or (R)eset mode. Now when you move the cursor dot with the arrow keys, it will either erase points or set points wherever it moves. To exit this mode press the 'X' key again.
- Q - quit and return to the menu

210	237
470	73
660	153
960	194
1240	166
1520	173
1790	141
END	51

Listing 1: ANIMATE

```
10 PCLEAR 8
20 DIM S(32,16)
30 DIM T(97,2)
40 PMODE 4,1
50 AB$="NONE"
60 CLS
```

```
70 SCREEN 0,0
80 PRINT@40,"graphic";
90 POKE 1071,32
100 PRINT@48,"animator";
110 PRINT@73,"BY WAYNE THUME"
120 PRINT@133,STRING$(21,"x")
130 PRINT@160,"     x<1> CREATE
PICTURE x"
140 PRINT      "     x
        x"
150 PRINT"     x<2> SEE ANIMATIO
N  x"
160 PRINT      "     x
        x"
170 PRINT"     x<3> SAVE ANIMATI
ON x"
180 PRINT"     x
   x"
190 PRINT"     x<4> LOAD ANIMATI
ON x"
200 PRINT"     x
   x"
210 PRINT"     x<5> QUIT
   x"
220 PRINT@453,STRING$(21,"x")
230 A$=INKEY$
240 IF A$="" THEN 230
250 ON VAL(A$) GOTO 270,1270,143
0,1810,2120
260 GOTO 230
270 REM CHOOSE PICTURE
280 CLS
290 PRINT"LAST PICTURE: ";AB$
300 PRINT@224,"WHICH PICTURE ";
310 INPUT AB$
320 IF VAL(AB$)<1 OR VAL(AB$)>96
 THEN 300
330 REM CREATE PICTURE
340 SCREEN 1,1
350 J=JOYSTK(0):K=JOYSTK(1)
360 J1=J*3.57:K1=K*2.82
370 GET(J1,K1)-(J1+31,K1+15),S,G
380 PUT(J1,K1)-(J1+31,K1+15),S,P
RESET
390 C=PEEK(65280):T=J1:U=K1
400 IF C=126 THEN R=X:GOTO 440
410 IF C=254 THEN R=X:GOTO 440
420 PUT(J1,K1)-(J1+31,K1+15),S,P
SET
430 GOTO 330
440 REM ENLARGE PICTURE
450 V=VAL(AB$):T(V,1)=T:T(V,2)=U
460 SCREEN 0,0
470 PUT(J1,K1)-(J1+31,K1+15),S,P
SET
480 CLS
490 FOR X=15 TO 46
500 FOR Y=7 TO 22
510 C=PPOINT(J1+(X-15),K1+(Y-7))
520 IF C=1 THEN SET(X,Y,1) ELSE
RESET(X,Y)
530 NEXT Y,X
540 REM START
550 CD=15:CE=7:D=0
560 C=POINT(CD,CE)
570 PRINT@0,STRING$(32,143);
580 A$=INKEY$
590 IF C=1 THEN RESET(CD,CE):SET
(CD,CE,1):ELSE SET(CD,CE,1):RESE
T(CD,CE)
600 IF A$="" THEN 580
610 J2=J1+(CD-15):K2=K1+(CE-7)
620 IF P=1 THEN SET(CD,CE,1):PSE
T(J2,K2)
630 IF P=2 THEN RESET(CD,CE):PRE
SET(J2,K2)
640 IF A$="S" THEN SET(CD,CE,1):
PSET(J2,K2)
650 IF A$="R" THEN RESET(CD,CE):
PRESET(J2,K2)
660 IF A$="X" THEN GOSUB 810
670 IF A$="C" THEN GOSUB 900
680 IF A$="G" THEN GOTO 1000
690 IF A$="O" THEN GOSUB 1100
700 IF A$="Q" THEN GOTO 1200
710 IF A$=CHR$(94) THEN CE=CE-1
720 IF A$=CHR$(10) THEN CE=CE+1
730 IF A$=CHR$(8) THEN CD=CD-1
740 IF A$=CHR$(9) THEN CD=CD+1
750 IF CD<15 THEN CD=46
760 IF CD>46 THEN CD=15
770 IF CE<7 THEN CE=22
780 IF CE>22 THEN CE=7
790 GOTO 560
800 GOTO 60
810 REM SWITCH MODES
820 IF FL=1 THEN FL=0:P=0:RETURN
830 FL=1
840 PRINT@0,"SWITCH TO <S> OR <R
>ESET        ";
850 B$=INKEY$
860 IF B$="" THEN 850
870 IF B$="S" THEN P=1:RETURN
880 IF B$="R" THEN P=2:RETURN
890 GOTO 850
900 REM CLEAR SCREEN
910 PRINT@0,"CLEAR SCREEN (Y/N)"
920 B$=INKEY$
930 IF B$=""THEN 920
940 IF B$="Y" THEN COLOR 0,1:LIN
E(T,U)-(T+31,U+15),PSET,BF:COLOR
 1,0:GOTO 970
950 IF B$="N" THEN 540
960 GOTO 920
970 FOR XI=103 TO 359 STEP 32
980 PRINT@XI,STRING$(17,128);
990 NEXT XI:RETURN
1000 REM GET PICTURE
1010 PRINT@0,"GET WHICH PICTURE
";
```

```
1020 INPUT AC$:V=VAL(AC$)
1030 IF V<1 OR V>96 THEN 1010
1040 GOSUB 1580
1050 PMODE 4,5
1060 GET(X,Y)-(X+31,Y+15),S,G
1070 PMODE 4,1
1080 PUT(T,U)-(T+31,U+15),S,PSET
1090 GOTO 440
1100 REM SEE HI-RES SCREEN
1110 PMODE 4,1
1120 SCREEN 1,1
1130 IF INKEY$="" THEN 1130
1140 PMODE 4,5
1150 SCREEN 1,1
1160 IF INKEY$="" THEN 1160
1170 PMODE 4,1
1180 SCREEN 0,0
1190 RETURN
1200 REM SAVE AND QUIT
1210 GOSUB 1580
1220 GET(T,U)-(T+31,U+15),S,G
1230 PMODE 4,5
1240 PUT(X,Y)-(X+31,Y+15),S;PSET
1250 PMODE 4,1
1260 GOTO 60
1270 REM SHOW ANIMATION
1280 SCREEN 1,1
1290 RJ=1
1300 FOR X=0 TO 255 STEP 32
1310 FOR Y=0 TO 191 STEP 16
1320 PMODE 4,5
1330 GET(X,Y)-(X+31,Y+15),S,G
1340 PMODE 4,1
1350 PUT(T(RJ,1),T(RJ,2))-(T(RJ,
1)+31,T(RJ,2)+15),S,PSET
1360 RJ=RJ+1
1370 IF PEEK(339)=251 THEN 60
1380 IF PEEK(345)=247 THEN 1390
ELSE 1370
1390 NEXT Y,X
1400 GOTO 60
1410 END
1420 GOTO 60
1430 REM END PROGRAM
1440 CLS
1450 Y=1
1460 FOR X=1024 TO 1215 STEP 2
1470 POKE X,T(Y,1)
1480 POKE X+1,T(Y,2)
1490 Y=Y+1
1500 NEXT X
1510 PRINT@480,"    SAVE ON (T)AP
E OR (D)ISK";
1520 A$=INKEY$:IF A$="" THEN 152
0
1530 PRINT@480,"
                 ";
1540 IF A$="D" THEN 1650
1550 IF A$="T" THEN 1720
1560 SOUND 1,1:GOTO 1510
1570 GOTO 60
1580 REM FIND AREA FOR PICTURE
1590 VA=0
1600 FOR X=0 TO 255 STEP 32
1610 FOR Y=0 TO 191 STEP 16
1620 VA=VA+1
1630 IF VA=V THEN RETURN
1640 NEXT Y,X
1650 REM SAVE INFO ON DISK
1660 PRINT@416,"NAME";:INPUT N$
1670 N$=LEFT$(N$,7)
1680 SAVEM N$,9728,15871,0
1690 N$=N$+"2"
1700 SAVEM N$,1024,1215,0
1710 GOTO 60
1720 REM LOAD FROM TAPE
1730 PRINT@416,"NAME";:INPUT N$
1740 N$=LEFT$(N$,7)
1750 PRINT@416,"POSITION TAPE AN
D PRESS ANY KEY.";
1760 IF INKEY$="" THEN 1760
1770 CSAVEM N$,7680,13823,0
1780 N$=N$+"2"
1790 CSAVEM N$,1024,1215,0
1800 GOTO 60
1810 REM LOAD PICTURE
1820 CLS
1830 INPUT"NAME ";N$
1840 N$=LEFT$(N$,7)
1850 CLS:PRINT@224,"LOAD FROM (T
)APE OR (D)ISK?"
1860 A$=INKEY$:IF A$="" THEN 186
0
1870 IF A$="D" THEN 1900
1880 IF A$="T" THEN 2000
1890 SOUND 1,1:GOTO 1850
1900 LOADM N$
1910 N$=N$+"2"
1920 LOADM N$
1930 Y=1
1940 FOR X=1024 TO 1215 STEP 2
1950 T(Y,1)=PEEK(X)
1960 T(Y,2)=PEEK(X+1)
1970 Y=Y+1
1980 NEXT X
1990 GOTO 60
2000 CLS:PRINT@224,"POSITION TAP
E AND PRESS ANY KEY."
2010 IF INKEY$="" THEN 2010
2020 CLOADM N$
2030 N$=N$+"2"
2040 CLOADM N$
2050 Y=1
2060 FOR X=1024 TO 1215 STEP 2
2070 T(Y,1)=PEEK(X)
2080 T(Y,2)=PEEK(X+1)
2090 Y=Y+1
2100 NEXT X
2110 GOTO 60
2120 CLS:END
```

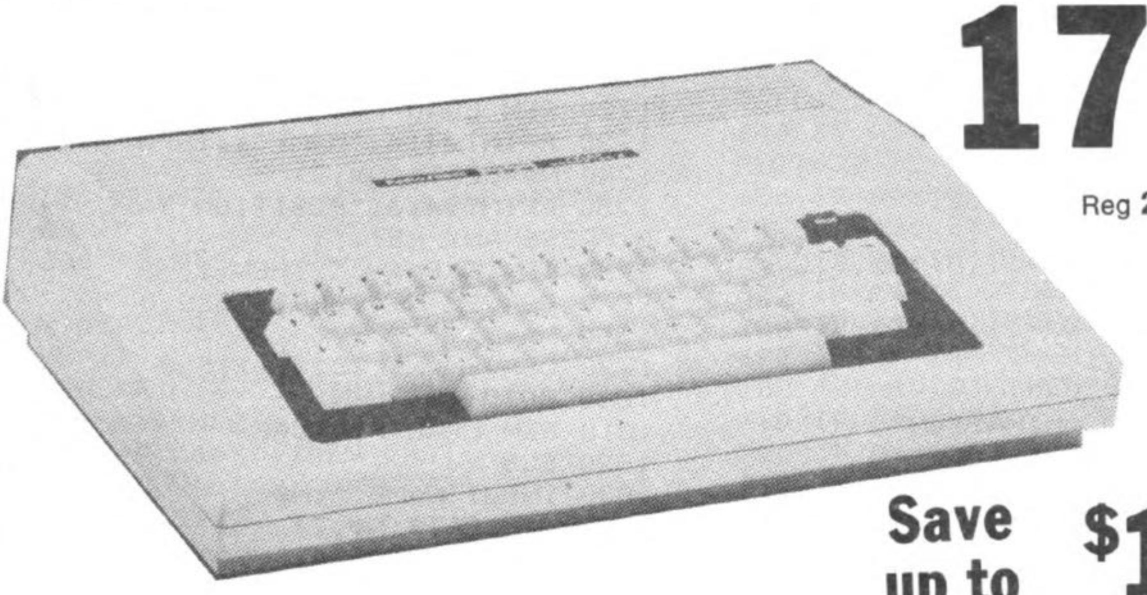
Tandy
ELECTRONICS

VERSATILE COLOR COMPUTERS FOR THE HOME

LOW AS

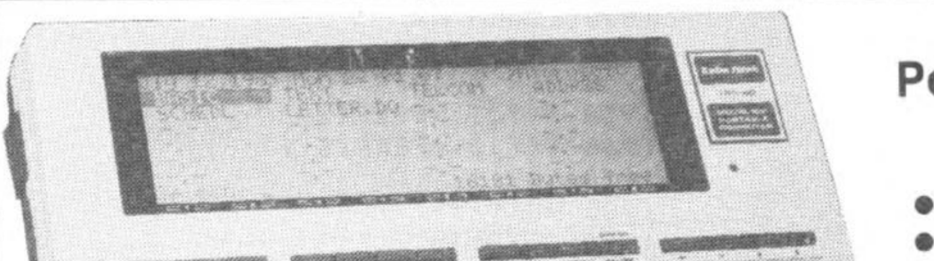
179^95

Reg 239.95



Save up to $150

16K Standard BASIC Color Computer. An affordable introduction to colourful and exciting computing! Program Paks™ will entertain or educate your family for hours. 26-3134. **Was 239.95 NOW 179.95 . . . . . . Save $60!**
16K Extended BASIC Color Computer. All the necessary features for multiple uses — education, entertainment, personal and household tasks! Easy expansion! 26-3136. Was 299.95 **NOW 229.95 . . . . Save $70!**
64K Extended BASIC Color Computer. Our most powerful! Advanced home users find this computer affordable and well suited to their business and personal needs. 26-3127 Was 449.95 **NOW 299.95 . . . . Save $150!**

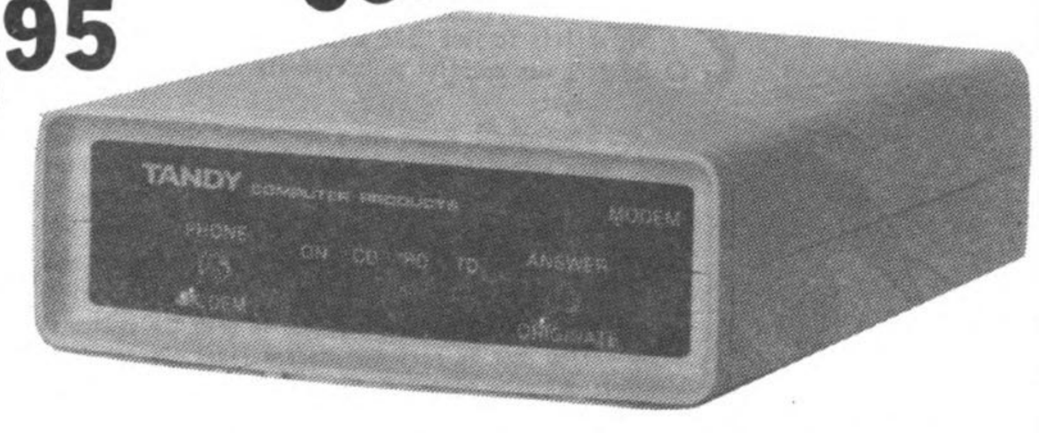


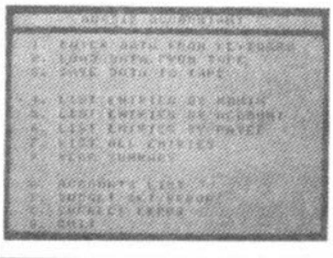
Model 100 Portable Computer

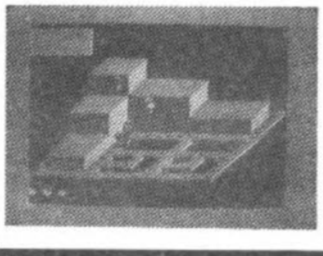
- *Has Five Built-in Programs*
- *8-Line by 40-Character Display*
- *Retains Data When Power is Off*

$899

Model 100 Portable. Amazing Micro Executive Workstation™ operates on batteries — the perfect companion for journalists, business people, students! Built-in programs for word-processing, address/phone directory, and appointment scheduling. 26-3802
8K RAM Expansion. Requires installation. 26-3816 . . . **119.95**


TANDY COMPUTER PRODUCTS
MODEM
PHONE
ON
CD
RD
TD
ANSWER
ORIGINATE





GAME

16K
ECB

Sir Eggbert Jumper Leaps To The Rescue

By David Dawson

One terrible day a hole appeared in the ground in a small kingdom. Terrible creatures began to spring from it and ravage the countryside. Many knights descended into the pit but none returned.

Finally, Eggbert, a strange knight who never used weapons and had only his odd ability to jump like a rabbit, volunteered to rid the kinodom of the menace. The king was doubtful but desperate, so he agreed to send the young,peculiar knight.

Sir Eggbert Jumper is a one-player game similar to many "climb-up" games like Donkey Kong but with more of a "swords and sorcery" theme. There are eight dungeon screens included in Eggy's tasks (his friends call him Eggy), five of which are different.

The last screen holds many surprises which you will have to play to see.

GAME PLAY

The main objective for Eggy is to grab the yellow key and move, not jump, below the yellow door to exit the dungeon level. Be careful - jumping into the door could erase it and trap you on that level.

Touching any red object will knock Eggy out. Touching yellow objects, except the key, will cause the screen to change colour. While the colours remain abnormal it is not possible to jump off the level Eggy is on, but be careful not to step off the platform. The creatures cannot harm Eggy while they are not red, so use this time to get away from them.

Touching blue objects, and all objects while the colours are not normal, scores points.

Listing 2: DEMO

```
 10 PCLEAR 8
 20 PMODE 4,1
 30 SCREEN 1,1
 40 Z=1:B=8
 50 DIM S(32,16)
 60 FOR X=0 TO 255 STEP 32
 70 FOR Y=0 TO 191 STEP 16
 80 GOSUB 140
 90 GET(112,B-7)-(143,B+8),S,G
 100 PMODE 4,5
 110 PUT(X,Y)-(X+31,Y+15),S,PSET
 120 NEXT Y,X
 130 GOTO 270
 140 PMODE 4,1:PCLS
 150 B=B+1
 160 A=SIN(Z)*10
 170 A=A+128
 180 IF C=1 THEN C=0:GOTO 250
 190 CIRCLE(A,B),5
 200 C=1
 210 LINE(127,0)-(127,192),PSET
 220 LINE(128,0)-(128,192),PRESET
 230 LINE(129,0)-(129,192),PSET
 240 Z=Z+.1:GOTO 260
 250 Z=Z+.1:GOTO 140
 260 RETURN
270 REM SEE ANIMATION
280 PMODE 4,1:PCLS
290 PMODE 3,1
300 FOR UO=1 TO 4:PCLS UO:NEXT:P
CLS2
310 PMODE 4,1:SCREEN 1,1
320 LINE(112,0)-(143,192),PRESET
,BF
330 LINE(9,9)-(102,182),PSET,BF
340 LINE(153,9)-(246,182),PSET,B
F
350 LINE(127,0)-(127,192),PSET
360 LINE(129,0)-(129,192),PSET
370 CIRCLE(56,96),40,0
380 CIRCLE(199,96),40,0
390 W=7
400 FOR X=0 TO 255 STEP 32
410 FOR Y=0 TO 191 STEP 16
420 PMODE 4,5
430 GET(X,Y)-(X+31,Y+15),S,G
440 PMODE 4,1
450 PUT(112,W)-(143,W+15),S,PSET
460 W=W+2
470 NEXT Y,X
480 CLS:PRINT@228,"AGAIN (Y/N)";
490 A$=INKEY$:IF A$="" THEN 490
500 IF A$="Y" THEN SCREEN 1,1:W=
1:GOTO 390
510 END
```

A creature will appear if too much time is taken on a level. This creature leaves a permanent trail that may make it impossible to continue and effectively end the game. Be careful to finish quickly.

JOYSTISK OPERATION

Move right or left by moving the joystick in the direction desired.

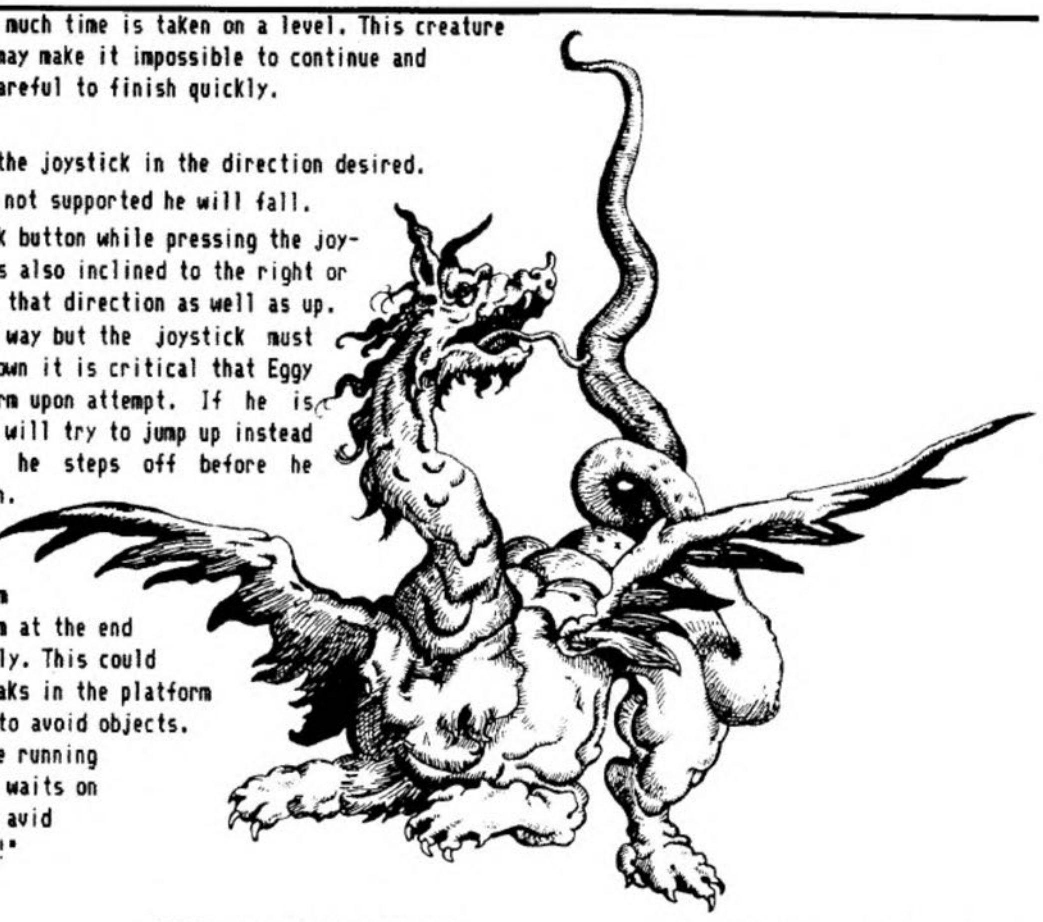
If at any time Eggy's feet are not supported he will fall.

To jump up, press the joystick button while pressing the joy--stick forward. If the stick is also inclined to the right or left, Sir Eggbert will jump in that direction as well as up.

Jumping down is done the same way but the joystick must be pulled back. When jumping down it is critical that Eggy is near the edge of the platform upon attempt. If he is too far away from the edge he will try to jump up instead of down; on the other hand, if he steps off before he jumps he will fall to his death.

If a platform is above Eggy when he makes an upward jump he will bounce off the platform and, if a platform is below him at the end of the jump, he will land safely. This could be used to jump over small breaks in the platform on which Eggy is travelling or to avoid objects.

Use PCLEAR2 and ENTER before running the program. As a hint to what waits on the last screen: The slogan of avid Eggy players is "Jump A Dragon!"



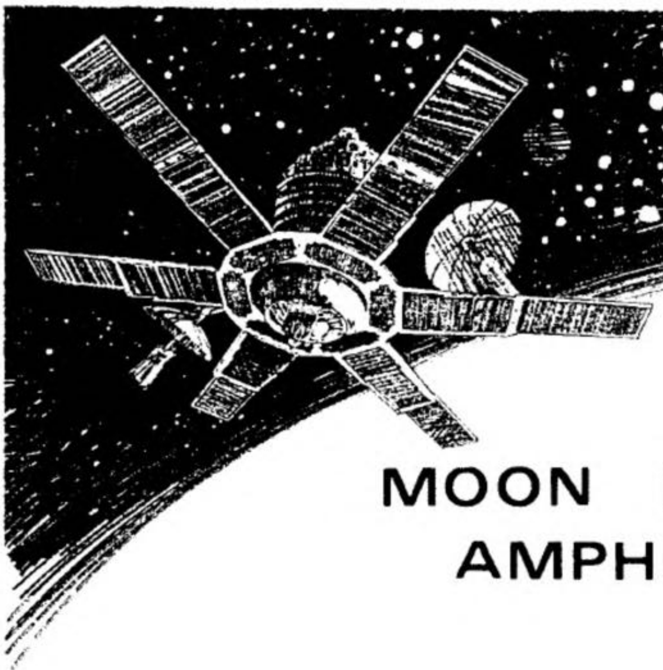
590	12
750	202
1130	152
1310	172
2000	67
4005	215
4420	187
4840	91
4960	223
END	56

The listing: EGGBERT

```
5 CLEAR100:PCLEAR2:PMODE1,1:PCLS
10 DIM H1(11,15):DIM H2(11,15):D
IM OJ(9,5):DIM C1(13,17):DIM C2(
13,15)
20 DRAW"C3BM20,120D2NR6D4R2NF4L2
D2L2D2L2BM106,120D2NL6D4L2NG4R2D
2R2D2R2"
30 FOR C=1TO8:READX,Y:PSET(X,Y,4
):NEXT:DATA 20,116,20,118,22,116
,22,118,104,116,104,118,106,116,
106,118
40 DRAW"C2BM36,90D4L2U2NU2R8D2"
50 COLOR4,1:LINE(86,58)-(98,72),
PSET,BF:PRESET(98,70):PRESET(98,
72):PRESET(98,58):PRESET(96,58):
PRESET(88,58):PRESET(88,60)
60 LINE(86,56)-(86,66),PRESET:LI
NE(92,68)-(98,70),PRESET,BF
70 PRESET(92,66):PRESET(90,66):P
SET(96,62,2):PSET(96,68,3)
80 GET(16,116)-(26,130),H1:GET(1
00,116)-(110,130),H2:GET(36,90)-
(44,94),OJ:GET(88,58)-(100,72),C
2
100 CLS:PRINT@128,"(C) 1982 BY D
AVID LIONELL DAWSON":PE=3
400 CL=0:KX=20:KY=168
405 IF RO=0 THEN 480
408 IF RO=8 THEN DRAW"BM40,80R4B
R3NU3D3BR4U6F6U6BR8F2ND4E2BR4D6R
6U6NL6BR4D6R6U6BR4D3ND3R4NF2U3NL
4BR12NR6D3NR6D3BR12U3NR6U3R6D6BR
4NR6U6R6BR4NR6D3NR6D4R6":FOR XX=
1 TO 3:PLAY "T10;L4;01;12;12;8;8
;4":NEXTXX
409 PLAY"T10;L4;03;1;3;5;3;5;6;6
;8;6;5;6;8;10;12;04;L1;1"
410 DRAW"BM"+STR$(DX)+","+STR$(D
Y)
420 SOUND255,8:FOR X=1 TO52:DRAW
"S"+STR$(X):GOSUB8200:NEXT:SC=SC
+RO*100:DRAW"S4"
480 RO=RO+1
482 IF RO=9 THEN SCREEN0,0:FOR X
=1 TO 250 STEP6:CLSRND(8):SOUNDX
,1:NEXT:CLS:PRINT@224, "DUNGEON
COMPLETE,BONUS=";SC*100:SC=SC+SC
*100:RESTORE
485 IF RO=9 THEN FOR C=1TO4:READ
X,X,X,X:NEXT
500 IF RO=9 THEN RO=1
510 COLOR 3,1:PCLS
515 LINE(0,0)-(254,4),PSET,BF
520 ON RO GOSUB 4000,4200,4000,4
400,4200,4400,4600,4800
540 READ F:FOR C=1 TO F
550 READPX,PY:DRAW"BM"+STR$(PX)+
","+STR$(PY):GOSUB 8000
570 NEXT
590 READ F:FOR C=1 TO F
610 READ PX,PY:DRAW"BM"+STR$(PX)
+","+STR$(PY):GOSUB8100
630 NEXT
640 READDX,DY
650 DRAW"BM"+STR$(DX)+","+STR$(D
Y)
660 GOSUB 8200
670 READNO:IF NO=0 THEN 700
680 FOR C=1 TO NO
690 READ PX(C),PY(C)
695 NEXT
700 READ PX,PY:PUT(PX,PY)-(PX+8,
PY+4),OJ
710 DATA 2,140,92,140,184,2,230,
85,130,131,10,0,2,80,30,50,76,14
,82
720 DATA 3,10,46,170,46,60,92,2,
50,89,160,132,128,52,2,128,30,12
8,76,208,36
```

```
730 DATA 2,140,92,140,184,2,230,
85,130,131,10,0,2,80,30,50,76,14
,82
740 DATA 3,190,46,130,92,8,138,2
,10,37,170,129,45,0,6,100,122,20
0,76,10,76,50,30,100,30,160,30,2
38,82
750 DATA 3,10,46,170,46,60,92,2,
50,89,160,132,128,52,2,128,30,12
8,76,208,36
760 DATA 3,190,46,130,92,8,138,2
,10,37,170,129,45,0,6,100,122,20
0,76,10,76,50,30,100,30,160,30,2
38,82
770 DATA 4,40,138,74,92,130,92,1
0,92,1,110,85,90,98,1,180,122,20
,129
780 DATA 1,180,138,1,100,129,10,
54,0,240,175
785 CO=200-RO*10
1000 IF RO<3 THEN NO=0
1010 CNT=0:K=0:I=1
1020 HX=50:HY=168
1025 SCREEN 1,0
1030 LX=HX:LY=HY:EX=0
1031 CNT=CNT+1
1039 IF PPOINT(HX+5,HY+18)=7 THE
N 1100
1040 IF PPOINT(HX+5,HY+18)=3 THE
N 1100
1050 IF PPOINT(HX+5,HY+35)=3 OR
PPOINT(HX+5,HY+35)=7 THEN HY=HY+
17:GOTO1200 ELSE 3000
1100 X=JOYSTK(0):Y=JOYSTK(1):IF
X>40 THEN M=1 ELSE IF X<20 THEN
M=2 ELSE M=0
1110 IF Y<20 THEN U=1 ELSE IF Y>
40 THEN U=2 ELSE U=0
1120 IF PEEK(65280)=254 OR PEEK(
65280)=126 THEN 1130 ELSE 1190
1130 PLAY"T255;12;1":IF U=2 THEN
 IF PPOINT(HX+15,HY+18)=1 OR PPO
INT(HX-15,HY+18)=1 THEN HY=HY+46
:GOTO1190
1140 IF PPOINT(HX+5,HY-27)=1 THE
N HY=HY-46:EX=15:GOTO1190 ELSEHY
=HY-17:EX=15:GOTO1190
1190 IF M=0 THEN 1200 ELSE IF M=
1 THEN HX=HX+10+EX ELSE IF M=2 T
HEN HX=HX-10-EX
1200 IF HX>245 THEN HX=245 ELSE
IF HX<15 THEN HX=15
1210 LINE(LX-10,LY)-(LX+14,LY+14
),PRESET,BF:IF M=1 THEN PUT(HX,H
Y)-(HX+12,HY+14),H1 ELSE PUT(HX,
HY)-(HX+12,HY+14),H2
1220 IF K=1 THEN PUT(HX,HY)-(HX+
10,HY+6),OJ
1230 IF NO=0 THEN 1300
1240 X=RND(NO):LINE(PX(X)-4,PY(X
))-(PX(X)+16,PY(X)+14),PRESET,BF
1250 IF HX<PX(X) THEN PX(X)=PX(X
)-6 ELSE PX(X)=PX(X)+6
1260 PUT(PX(X),PY(X))-(PX(X)+12,
PY(X)+14),C1
1300 IF CNT<CO THEN 1400
1310 KX=KX+I*4:IF KX<10 THEN KY=
KY-46:I=1 ELSE IF KX>245 THEN KY
=KY-46:I=-1
1315 IF KY< 30 THEN KY=168
1320 PUT(KX,KY)-(KX+12,KY+14),C2
:CIRCLE(KX-2,KY+5),5,4,3
1400 IF K=0 THENIF RND(40)=1 THE
N PUT(PX,PY)-(PX+8,PY+4),OJ
1410 IF RO<>8 THEN 1450
1420 IF DF=0 THENDF=1: PX(1)=136
:PY(1)=48 ELSE LINE(PX(1),PY(1))
-(PX(1)+12,PY(1)+14),PRESET,BF
1430 IF PX(1)<HX THEN PX(1)=PX(1
)+4 ELSE PX(1)=PX(1)-4
1435 PY(1)=PY(1)+RND(30)
1440 IF PPOINT(PX(1)-6,PY(1))<>1
 OR PY(1)>182 THEN DF=0:PLAY"T25
5;2;6;10" ELSE PUT(PX(1),PY(1))-
(PX(1)+12,PY(1)+14),C1
1450 IF PPOINT(HX-12,HY+8)=1 AND
 PPOINT(HX+12,HY+8)=1 THEN 2000
1455 IF PPOINT(HX-12,HY+8)=5 THE
N 2000
1460 IF PPOINT(HX-12,HY+8)=4 OR
PPOINT(HX+12,HY+8)=4 THEN 3200
1470 IF PPOINT(HX-12,HY+8)=2 OR
PPOINT(HX+12,HY+8)=2 THEN IF HX>
PX-13 AND HX<PX+18 AND HY<PY AND
 HY>PY-40 THEN K=1:PLAY"T1;04;L1
6;12":GOTO1800 ELSE CL=CNT+50:SC
REEN 1,1:GOTO1800
1472 SC=SC+RO*10
1480 SOUND 250,1
1800 FOR X=12 TO1 STEP -2:PLAY"T
255;L16;04;"+STR$(X):CIRCLE(HX,H
Y+6),X+10,2,.2:CIRCLE(HX,HY+6),X
+10,1,.2:NEXT
2000 IF CNT>CL THEN SCREEN 1,0
2005 IF K=1 AND PPOINT(HX+5,HY-1
2)=2 THEN 400
2010 GOTO 1030
3000 FOR X=HY TO 171:HY=X:LINE(H
X-5,HY)-(HX+10,HY-10),PRESET,BF:
PUT(HX,HY)-(HX+10,HY+14),H1:NEXT
3200 FORX=1TO10:SOUNDX,1:SCREEN1
,1:SCREEN1,0:NEXTX
3210 LINE(HX,HY)-(HX+10,HY+12),P
RESET,BF
3220 DRAW"BM"+STR$(HX)+","+STR$(
HY+16)+"H4R8NU6R8U2L2"
3225 SOUND100,10:SOUND1,20
3230 IF PE=1 THENSCREEN 0,0:CLS:
PRINT@160,"TRY AGAIN ADVENTURER!
":PRINT"YOU REACHED LEVEL";RO:PR
INT"SCORE WAS";SC:END ELSE PE=PE
-1:CLS:SCREEN 0,0: PRINT@160,"AD
VENTURERS REMAINING";PE:FORX=1TO
2000:NEXT:GOTO1010
4000 LINE(0,184)-(254,191),PSET,
BF
4005 LINE(0,0)-(254,2),PSET,BF
4010 LINE(0,138)-(208,144),PSET,
BF
4020 LINE(0,92)-(70,98),PSET,BF
4030 LINE(104,92)-(254,98),PSET,
BF
4040 LINE(0,46)-(100,52),PSET,BF
4050 LINE(144,46)-(214,52),PSET,
BF
4060 LINE(208,46)-(214,92),PSET,
BF
4070 COLOR4,1:LINE(202,26)-(214,
40),PSET,BF:PRESET(214,26):PRESE
T(202,26):CIRCLE(208,32),5,2:PSE
T(208,32):LINE(204,38)-(212,38),
PRESET
4080 GET(202,26)-(214,40),C1:LIN
E(202,26)-(214,40),PRESET,BF
4100 RETURN
4200 LINE(0,184)-(254,191),PSET,
BF:LINE(0,138)-(96,144),PSET,BF
4210 LINE(128,138)-(216,144),PSE
T,BF:LINE(44,92)-(254,96),PSET,B
F
4220 LINE(0,46)-(96,52),PSET,BF:
LINE(128,46)-(228,52),PSET,BF
4230 COLOR4,1:CIRCLE(18,172),14,
4:PAINT(18,172),4,4:FOR X=1 TO 4
:PSET(RND(28),146+RND(12),4):NEX
T:LINE(10,158)-(28,165),PSET,BF
4240 RETURN
4400 LINE(0,184)-(254,191),PSET,
BF:LINE(0,138)-(44,144),PSET,BF
4410 LINE(154,138)-(230,144),PSE
T,BF:LINE(64,92)-(152,98),PSET,B
F
4420 LINE(202,92)-(254,98),PSET,
BF:LINE(0,46)-(64,52),PSET,BF:LI
NE(178,46)-(218,52),PSET,BF
4430 RETURN
4600 LINE(0,184)-(254,191),PSET,
BF
4610 LINE(0,138)-(204,144),PSET,
BF:LINE(0,92)-(28,98),PSET,BF:LI
NE(74,92)-(148,98),PSET,BF:COLOR
4,1:LINE(148,0)-(254,98),PSET,BF
:LINE(0,0)-(148,52),PSET,BF
4620 RETURN
4800 LINE(0,62)-(26,34),PSET:LIN
E-(44,56),PSET:LINE-(54,30),PSET
:LINE-(216,30),PSET:LINE-(222,52
),PSET:LINE-(240,28),PSET:LINE-(
244,60),PSET:LINE-(254,42),PSET
4810 PAINT(100,20),3,3
4820 COLOR 4,1
4830 LINE(76,0)-(70,14),PSET:LIN
E-(118,46),PSET:LINE-(94,16),PSE
T:LINE-(112,2),PSET:LINE-(130,22
),PSET:LINE-(138,30),PSET:LINE-(
138,40),PSET:LINE-(142,44),PSET:
LINE-(146,40),PSET:LINE-(148,30)
,PSET:LINE-(154,22),PSET
4840 LINE-(150,16),PSET:LINE-(14
2,12),PSET:LINE-(136,16),PSET:LI
NE-(130,22),PSET:LINE(156,22)-(1
78,4),PSET:LINE-(212,18),PSET:LI
NE-(202,46),PSET:LINE-(232,12),P
SET:LINE-(232,0),PSET
4850 PAINT(144,0),2,4:PAINT(144,
20),4,4
4870 LINE(120,10)-(122,28),PSET:
LINE-(136,34),PSET:LINE-(148,34)
,PSET:LINE-(162,28),PSET:LINE-(1
64,12),PSET
4880 PAINT(158,22),4,4:PAINT(128
,22),4,4
4885 CIRCLE(138,26),2,2,.5:CIRCL
E(148,26),2,2,.5
4890 PSET(138,24,3):PSET(138,26,
3):PSET(148,24,3):PSET(148,26,3)
:PSET(144,40,3):PSET(140,40,3)
4900 COLOR 2,1:LINE(142,12)-(134
,9),PSET:LINE(146,8)-(138,5),PSE
T:DRAW"C4BM128,32D8ND8R4ND8L8ND8
BM156,32D8ND8R4ND8L8D8"
4910 COLOR 3,1:LINE(0,92)-(162,9
8),PSET,BF
4920 LINE(0,184)-(254,191),PSET,
BF:LINE(62,138)-(128,144),PSET,B
F:LINE(178,138)-(200,144),PSET,B
F
4930 LINE(50,90)-(140,100),PRESE
T,BF
4950 CIRCLE(142,58),4,4:CIRCLE(1
42,58),1,2:COLOR4,1:LINE(146,56)
-(138,48),PSET
4960 GET(136,48)-(148,62),C1
4970 LINE(136,48)-(148,62),PRESE
T,BF
4980 RETURN
8000 DRAW"C3U10E4R4F4NL8D10":PAI
NT(PX+4,PY-3),3,3
8010 RETURN
8100 DRAW"C2NU8R8E4H4ND8L8G4F4"
8110 RETURN
8200 DRAW"C2NR20NL4D2NL4D2NL4D6N
L4D2NL4D2NL4D6R20U20":PAINT(DX+2
,DY+2),2,2
8205 DRAW"BM-14,+4C3R6D2L2ND4L4"
8210 RETURN
```

DESTINATION -



MOON BASE AMPHIBIA

by Nick Bradbury

The idea for Amphibia is loosely based on the game arcade game Moon Patrol. A Hi-Res 32K game, Amphibia has several features which you may find interesting:

1) Hi-Res artifact Graphics

2) Machine language subroutines for "superfast" speed

3) Four totally different screens

4) Scrolling screens for continuous play

Because of the many machine language subroutines in Amphibia, I will explain how these routines work and also how the graphics were created.

How To Play

You travel the surface of an alien planet trying to reach your final destination: Moon Base Amphibia! Your vehicle is capable of traveling under water as well as overland. To reach your base, you must complete three levels of difficult obstacles: 1) boulders and rockets, 2) bombes and craters, and 3) geysers and surface rockets.

On each level, the right joystick controls your movements (you can only move left and right) and the firebutton controls jumps.

On Level one, you must jump over boulders while avoiding jumping up into the overhead rockets. Your timing must be perfect to jump over the boulders. After the timer runs out (at the bottom of the screen), you will go to the next screen.

In the second screen, you are under the alien ocean. Large craters appear and you must jump over them. Occasionally, undersea patrollers appear above you. You must avoid the bombs they drop and watch out for the craters the bombs leave when they hit the ocean floor.

After this screen, you will go on to level three. There are two obstacles you must avoid in this screen. The surface rockets appear to your left. These rockets pause for a moment, then unexpectedly shoot across the screen. You must jump over these rockets using split-second timing. Geysers will occasionally appear at the right of the screen. These are impossible to completely jump, but you must jump high enough so the water does not reach your cockpit.

After completion of these three screens, you will appear in front of the moon base. Above and below you are two revolving treadmills. The top treadmill is slowly coming down upon you, so you must act fast. To your right are two berserkers which you must avoid before you can reach your base. To reach your base, you must move your vehicle to the right. Upon completion of this screen, you will receive a bonus and will return to the first screen.

How to Create Graphics

The shapes in *Amphibia* are created in an extremely simple way. All shapes were originally drawn on graph paper. Because of Extended BASIC's DRAW command, this drawing can be easily translated into a draw string. For example, if a line on the graph paper moves right four spaces, I translate this to equal "R4." After the complete string has been created, I draw it the size I want using the powerful scale option of the DRAW command. The PAINT command was used to fill in the shapes.

The colors used in this game are the famous "artifact" colors that give different colors in PMODE 4 using SCREEN 1,1. The best way I've found to use these colors is by using memory location 178. By changing the value in this location, the colors in the graphics commands are changed. By poking a 1 into this location, blue will show up. By painting in this mode with the format of PAINT (x,y),,1, the color blue will show up instead of white.

The ML Subroutines

The most important subroutine is the routine that copies an object onto the graphics screen. The other routines do the same action or scroll the background. I will only explain the copy routine, for I feel it is the most useful.

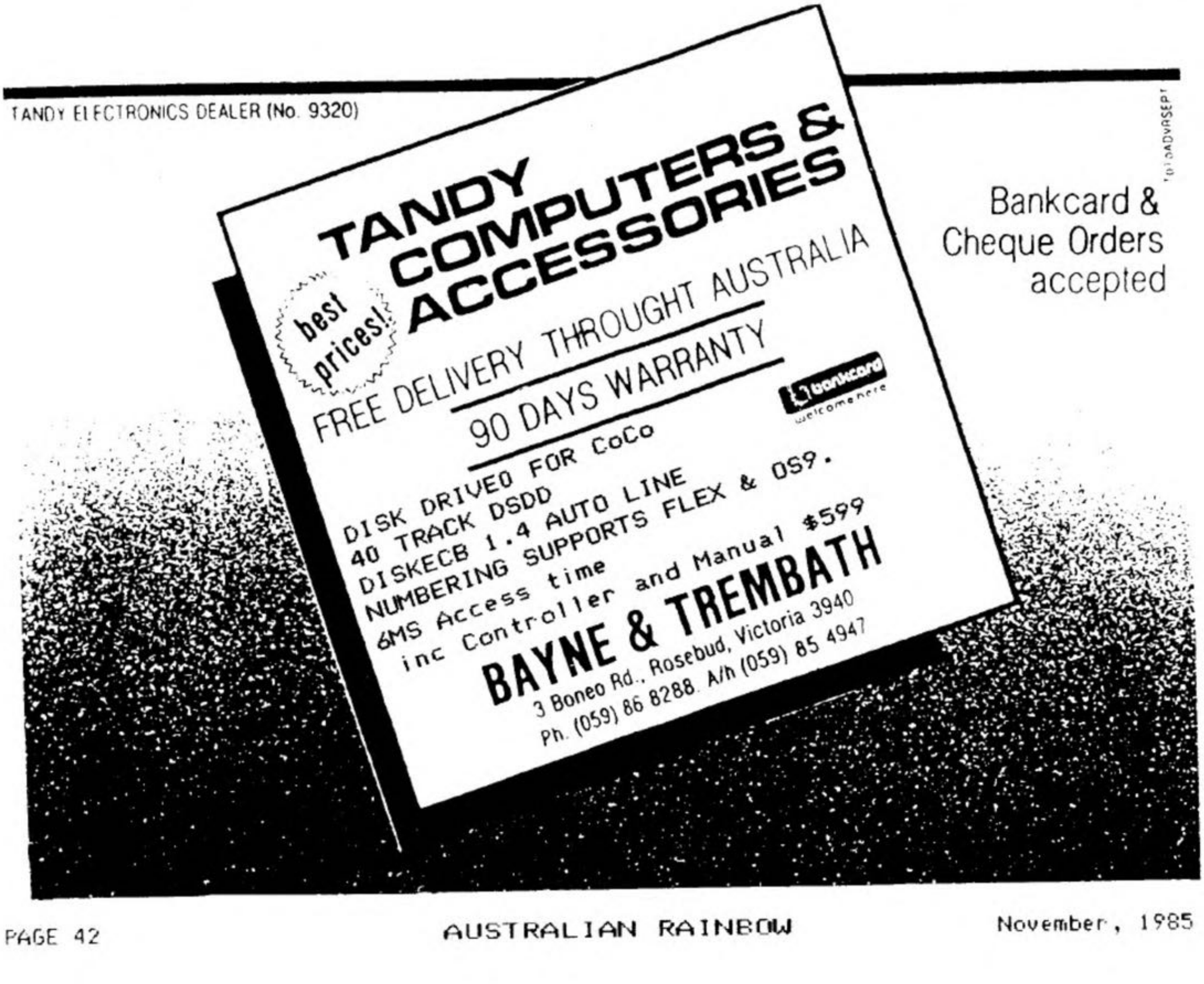
This routine (see accompanying listing) uses less than 50 bytes. First, draw the shapes onto pages five through eight. The ML routine copies these shapes onto a specified section of the viewing screen. The ROM routine at $B3ED (hexadecimal) is used to send the location from the USR command to the ML routine. This ROM routine transfers the contents in between the USR brackets and places it into the 'D' register. The ML routine takes this location and places the shape in the desired memory location. You may use this routine in any of your programs.

Getting it Working

There are two problems you may find in *Amphibia*. First, the artifact colors may not turn out correctly. To solve this, you will see a cleared screen at the beginning of the program. If this screen is a bluish color, press Reset and rerun the program until the colors are correct. After a short pause, the screen will change and the program will continue.

The other problem you may have is with the famous "speed-up" POKE that is used. If you have problems with this POKE, change the value of "PK" in Line 0 to 65494. This will set the speed for the entire program.

112	233	5255	218
410	33	6900	24
1016	104	7124	85
1218	60	7430	95
2000	179	7905	158
3052	69	8130	104
3162	234	8400	33
3316	218	8984	62
5050	151	9190	118
5175	128	END	128



Listing 1: AMPHIBIA

```
0 CLS0:CLEAR85,31000:PCLEAR8:DIM
A1(10):PK=65495:POKEPK,0:PMODE3,
1:PCLS3:PMODE4,1:SCREEN1,1:S1=0:
NS=4:PX=10:PY=12:GOSUB5000:GOSUB
2000:CLS:PRINT@32,Z$;:PRINT@448,
Z$;:PRINT@72,"a m p h i b i a":P
RINT@131,"(C)1983 BY NICK BRADBU
RY":PRINT@160,Z$;
20 SCREEN0,0:PMODE4,1:PCLS0:PMOD
E4,5:PCLS0:POKEPK,0:PRINT@224:PR
INT@256:PRINT@284:GOSUB1000:GOSU
B3000:DEFUSR0=31000:DEFUSR1=3105
0:DEFUSR2=31100:DEFUSR3=31150:DE
FUSR4=31200:DEFUSR5=31250:DEFUSR
8=31400:GOSUB6300:GOSUB6500:PLAY
"O3V30L255T255":POKE65494,0:
85 IFINKEY$=CHR$(13)THEN90ELSEPL
AY"FAAE":GOTO85
90 R1=0:R2=0:TI=0:POKEPK,0:BG=52
54:BX=5:E1=0:E2=0:PC=0:JP=0:U=US
R0(BG)
98 LV=1:PMODE4,1:COLOR1,0:SCREEN
1,1
100 J0=JOYSTK(0):IFJP=1THEN108EL
SEIFJ0<20THEN200ELSEIFJ0>43THEN2
25
108 PC=1-PC:U=USR0(BG):U=USR4(0)
:IFPC=1THENEXEC31550
112 IFR1=1THENGOSUB405
120 IFJP=1THEN310ELSEIF(PEEK(&HF
F00)AND1)=0THEN30J
124 IFPC=1THENBG=BG-32ELSEBG=BG+
32
130 U=USR4(0):U=USR3(0):IFR1=1TH
ENGOSUB405ELSEIFRND(15)=1THENGOS
UB400
144 IFR2=1THENGOSUB425ELSEIFRND(
26)=1THENGOSUB420
180 TI=TI+1:IFTI>8THENTI=0:GOSUB
6400
190 PLAY"O5V31P255":GOTO100
200 IFBX<1THEN108ELSEBX=BX-1:BG=
BG-1
205 U=USR1(BG+7):GOTO108
225 IFBX>16THEN108ELSEBX=BX+1:BG
=BG+1
230 U=USR1(BG-1):GOTO108
300 JP=1:J2=0:J1=0:PLAYP2$
310 IFJ2=1THEN320ELSEJ1=J1+1:IFJ
1>4THENJ2=1ELSEBG=BG-128
315 GOTO330
320 J1=J1-1:IFJ1<1THEN350ELSEBG=
BG+128
330 GOTO130
350 J2=0:JP=0:GOTO130
400 R1=1:E1=5755:U=USR5(E1):X1=2
9
405 P1=PEEK(E1+98):IFP1<>255THEN
910ELSEE1=E1-1:X1=X1-1:IFX1<1THE
N414
410 U=USR5(E1):RETURN
414 R1=0:I=480:U=USR1(E1-I):U=US
R1(E1-I+1):U=USR1(E1-I+2):U=USR1
(E1-I+3):U=USR1(E1-I+4):RETURN
420 R2=1:E2=4768:U=USR8(E2):X2=1
425 P1=PEEK(E2+258):IFP1<>42THEN
910ELSEE2=E2+1:X2=X2+1:IFX2>28TH
EN440
430 U=USR8(E2):RETURN
440 R2=0:FORI=0TO4:U=USR1(E2+I):
NEXTI:RETURN
910 PLAYPP$:NS=NS-1:CLS:SCREEN0:
PRINT@32,Z$;:PRINT@448,Z$;:PRINT
@99,"YOU HAVE"NS"BUGGIES LEFT.":
PRINT@163,"YOUR SCORE IS"S1"POIN
TS.":PMODE4,1:COLOR1,0:LINE(0,98
)-(255,150),PRESET,BF:FORX=0TO25
5:Y=143+RND(3):PSET(X,Y):PSET(X,
Y+1):NEXTX:PMODE4,1:COLOR1,0:
955 R1=0:JP=0:J2=0:PC=0:R2=0:GOS
UB6500:POKE65494,0:PRINT@387,"PR
ESS <ENTER> TO CONTINUE"
975 IFINKEY$=CHR$(13)THEN980ELSE
975
980 CLS:POKEPK,0:GOTO98
1000 PMODE3,1:COLOR4,1:LINE(0,0)
-(255,30),PSET,B:LINE(4,4)-(248,
26),PSET,B:PAINT(2,2),3,4:PMODE4
,1:COLOR1,0:FORX=0TO255:Y=143+RN
D(3):PSET(X,Y):PSET(X,Y+1):NEXTX
:POKE178,2:PAINT(128,190),,1:COL
OR1,0:Y=90:X1=256:FORX=0TO128:PS
ET(X,Y):PSET(X1,Y):PSET(X,Y+1)
1016 NEXTX:PMODE4,1:COLOR1,0:LIN
E(0,170)-(255,192),PRESET,BF:LIN
E(0,170)-(255,192),PSET,B:D1$="R
2F1E1R1D1F1D1L1G1H1D1L2H1G1L1H1L
1E1U1R1E1R1":PMODE4,1:DRAW"C1S16
BM44,58;XD1$;BM128,61;XD1$;BM212
,58;XD1$;":COLOR1,0:PAINT(60,60)
,,1:PAINT(130,64),,1:PAINT(214
1100 PMODE4,1:COLOR1,0:PCLS0:D1$
="E1U1F1R1F1":LINE(0,150)-(255,1
88),PSET,BF:FORX=6TO250STEP12:CI
RCLE(X,56),8,1:NEXTX:LINE(0,52)-
(255,70),PRESET,BF:LINE(251,51)-
(256,51),PSET:POKE178,1:PAINT(12
8,60),,1:COLOR1,0:PAINT(100,160)
,,1:PMODE4,1:COLOR1,0:PMODE4,1
1185 PMODE3,1:COLOR4,1:LINE(0,0)
-(255,30),PSET,B:LINE(4,4)-(248,
26),PSET,B:PAINT(2,2),3,4:PMODE4
,1:COLOR1,0:FORI=1TO30:PRESET(RN
D(255),161+RND(7)):NEXTI:RETURN
1200 PMODE4,1:COLOR1,0:PCLS0:PMO
DE3,1:COLOR3,1:LINE(0,156)-(255,
180),PSET,BF:PMODE4,1:COLOR1,0:F
ORI=1TO100:PRESET(RND(256),159+R
ND(8)):NEXTI:X1=256:COLOR1,0:Y=8
0:FORX=0TO128:PSET(X,Y):PSET(X,Y
+1):PSET(X1,Y):PSET(X1,Y+1):X1=X
1-1:Y=Y-2+RND(3):IFY<72THENY=7
1218 NEXTX:PMODE3,1:COLOR4,1:LIN
E(0,0)-(255,30),PSET,B:LINE(4,4)
-(248,26),PSET,B:PAINT(2,2),3,4:
PMODE4,1:COLOR1,0:D1$="R2F1E1R1D
1F1D1L1G1H1D1L2H1G1L1H1C1E1U1R1E
1R1":PMODE4,1:DRAW"C1S16BM36,40;
XD1$;BM110,44;XD1$;BM210,40;XD1$
;":PAINT(42,49),,1:PAINT(116,5
1280 PMODE4,1:COLOR1,0:LINE(0,17
0)-(255,192),PRESET,BF:LINE(0,17
0)-(255,192),PSET,B:POKE178,1:PA
INT(128,66),,1:RETURN
```

```
1300 PMODE4,1:COLOR1,0:PCLS0:LIN
E(0,170)-(255,192),PRESET,BF:LIN
E(0,170)-(255,192),PSET,B:PMODE3
,1:COLOR4,1:LINE(0,0)-(255,30),P
SET,B:LINE(4,4)-(248,26),PSET,B:
PAINT(2,2),3,4:PMODE4,1:COLOR1,0
:D1$="U4E2R1U4E1U2H1L1H1U1H1U1E1
U1E1R1E1U1E1D1F1E1U1F1D1F1R1F1
1340 LINE(0,31)-(255,48),PSET,BF
:LINE(0,162)-(255,169),PSET,BF:P
MODE4,1:COLOR1,0:CIRCLE(180,89),
10,1:POKE178,1:PAINT(180,115),,1
:COLOR1,0:RETURN
2000 P2$="02V30L255T255DCGAFB":P
3$="03V30L255T255FA":P4$="01V30L
255T255A02G01A":P5$="05V30L255T2
55E03A01G":PP$="01V30L255T255FFA
FEAV25EGFADCEV20BGFEDAV15GFDACEV
10CBEGDDAV5FDEADV2GFEDA":Z$=STRI
NG$(32,"*")
2505 TM$="BM+3,0U6L3R6BR2BD6;BM+
5,-6L4R2D6L2R4BR3;BM+1,0U6R1F2D1
U1E2R1D6BR3;BM+5,0L4U6R4BD3BL2L1
BD3BR6;":RETURN
3000 PMODE4,5:D1$="E1R9NR2G1D2R4
U2NH1D3G1U1L2D1L3U1L2D1L1U1L2D1L
1H1U3":D2$="R2D1NL2D1L1NU2L1U2":
DD$="NL2R1D1R1H1F1D1L5U1E1":PMOD
E4,5:DRAW"C1S16BM34,92;XDD$;":PO
KE178,1:PAINT(32,97),,1:COLOR1,0
:CIRCLE(28,101),2,1:CIRCLE(39,10
1),2,1:GET(20,90)-(46,105),A1,
3010 DRAW"C1S16BM8,8;XD1$;":DRAW
"BM14,8;XD2$;BM22,8;XD2$;BM30,8;
XD2$;":POKE178,1:PAINT(40,21),,1
:COLOR1,0:CIRCLE(20,24),4,1:CIRC
LE(32,24),4,1:CIRCLE(52,24),4,1:
DRAW"C1S16BM8,42;XD1$;":DRAW"BM1
4,42;XD2$;BM22,42;XD2$;BM30,42;X
D2$;":POKE178,2:PAINT(40,55),,
3028 LINE(0,30)-(92,68),PSET,B:P
OKE178,1:PAINT(1,31),,1:PMODE3,5
:COLOR2,1:LINE(0,30)-(92,68),PSE
T,B:PMODE4,5:COLOR1,0:D3$="R1F1D
1F1L6U1E1R1E1":PMODE4,5:COLOR1,0
:DRAW"BM98,13;XD3$;":PAINT(98,20
),,1:LINE(92,21)-(94,23),PRESET:
LINE(92,21)-(95,18),PRESET:LIN
3052 LINE(140,14)-(150,22),PSET,
B:POKE178,2:PAINT(144,16),,1:COL
OR1,0:PMODE4,5:COLOR1,0:D5$="R8F
1L9U1":DRAW"C1S12BM215,16;XD5$;"
:PAINT(220,16),,1:PSET(243,19):P
MODE3,5:COLOR3,1:X=214:FORY=13TO
15:LINE(X,Y)-(X+12,Y),PSET:X=X+2
:NEXTY:X=222:FORY=19TO21:LINE(
3100 PMODE4,5:COLOR1,0:D1$="E1U1
E1R2F1D1R1E1R3E1R2E1U1H1R4G1D3F1
D3H1D3F1L4E1U1H1L2H1L3H1L1D1G1L2
H1U1H1":DRAW"C1S8BM102,48;XD1$;"
:CIRCLE(108,47),3,1:PAINT(126,47
),,1:PAINT(108,51),,1:PAINT(108,
43),,1:LINE(118,46)-(134,48),PRE
SET,B:COLOR3,1:FORX=120TO130ST
3142 PMODE4,5:DRAW"C1S8BM182,52;
XD2$;":POKE178,2:PAINT(188,52),,
1:COLOR1,0:DRAW"BM168,52;XD2$;":
D1$="D4R12U4G2L1G1L1G1H1L1H2L1H1
":DRAW"C1S16BM208,50;XD1$;":PAIN
T(220,60),,1:PAINT(246,60),,1:PM
ODE4,5:COLOR1,0:LINE(94,33)-(256
,64),PSET,B:POKE178,1:PAINT(95
3162 PMODE4,5:COLOR1,0:LINE(254,
49)-(255,64),PSET,B:PMODE4,5:COL
OR1,0:D1$="R1E1U1F1R1F1R2E1R1E1F
1R1F1":D2$="E1U1H1L1G1U2E1R2F1E1
R2F1D2H1L1G1D1F1":PMODE4,5:DRAW"
C1S16BM12,101;XD1$;":LINE(12,102
)-(64,102),PSET:POKE178,2:PAINT(
50,100),,1:PAINT(26,100),,1:PM
3220 DRAW"C4S16BM29,98;XD2$;":PM
ODE4,5:POKE178,1:PAINT(36,88),,1
:PMODE3,5:PMODE4,5:DRAW"C1S16BM8
4,101;XD1$;":LINE(84,102)-(136,1
02),PSET:POKE178,2:PAINT(96,100)
,,1:PAINT(124,100),,1:PMODE3,5:C
OLOR3,1:LINE(84,102)-(136,102),P
SET:DRAW"C4S24BM98,98;XD2$;":P
3246 DRAW"S8BM34,92;XD2$;":COLOR
4,1:LINE(110,100)-(110,80),PSET:
LINE(106,100)-(106,80),PSET:LINE
(114,98)-(114,80),PSET:PMODE4,5:
COLOR1,0:D3$="E1R5F1R1F1R1F1L8E1
L3D1H1U1E1":DRAW"C1S12BM24,121;X
D3$;":PAINT(42,124),,1:D4$="U1E1
R2F2L5":D5$="G2R3E2L3":PMODE3,
3266 DRAW"BM28,124;XD5$;":PAINT(
28,128),3,3:LINE(32,130)-(10,130
),PRESET:COLOR2,1:FORX=40TO54STE
P4:PRESET(X,126):NEXTX:LINE(46,1
21)-(48,123),PSET,BF:PRESET(49,1
21):RETURN
3300 PMODE4,5:COLOR1,0:PCOPY5TO1
:PCLS0:PCOPY1TO5:LINE(0,29)-(255
,48),PRESET,BF:D1$="NU4ND4NL4NR4
":D2$="NE4NF4NG4NH4":LINE(0,42)-
(255,42),PSET:LINE(0,54)-(255,54
),PSET:LINE(0,40)-(255,40),PSET:
LINE(0,56)-(255,56),PSET:POKE178
,2:PAINT(100,55),,1:PAINT(100,
3316 LINE(0,62)-(255,62),PSET:LI
NE(0,74)-(255,74),PSET:LINE(0,60
)-(255,60),PSET:LINE(0,76)-(255,
76),PSET:POKE178,2:PAINT(128,61)
,,1:PAINT(128,75),,1:COLOR1,0:FO
RX=8TO256STEP16:CIRCLE(X,48),7,1
:CIRCLE(X,68),7,1:NEXTX:DRAW"C1S
4":FORX=8TO256STEP16:DRAW"BM"+
5000 ML$="BDB3ED1F01108E1E01C607
A6A0A7805A26F930881931A819108C21
872DEB39":TT=31000:GOSUB5500:ML$
="BDB3ED1F01108E1E08C601A6A0A780
5A26F930881F31A81F108C21882DEB39
":TT=31050:GOSUB5500:ML$="BDB3ED
1F018600C607A7805A26FB39":TT=311
00:GOSUB5500
5030 ML$="8E1001108E1000A680A7A0
8C129F26F78E129F108E12BFA684A7A4
3088E031A8E08C0FFF26F139":TT=311
50:GOSUB5500:ML$="8E1801108E1800
A680A7A08C18BF26F78E18BF108E18DF
A684A7A43088E031A8E08C17FF26F139
":TT=31200:GOSUB5500
5050 ML$="BDB3ED1F01108E1FAAC605
A6A0A7805A26F930881B31A81B108C21
2E2DEB39":TT=31250:GOSUB5500:ML$
="BDB3ED1F01108E1FB9C606A6A0A780
5A26F930881A31A81A108C20BE2DEB39
":TT=31400:GOSUB5500
5090 ML$="8E0CE1108E0CE0A680A7A0
8C0FDF26F78E0FDF108E0FFFA684A7A4
3088E031A8E08C0CFF26F139":TT=315
50:GOSUB5500:ML$="BDB3ED1F01108E
21C1C607A6A0A7805A26F930881931A8
19108C25C72DEB39":TT=31600:GOSUB
5500
5110 ML$="BDB3ED1F01108E2288C601
A6A0A7805A26F930881F31A81F108C26
682DEB39":TT=31650:GOSUB5500:ML$
="8E18C1108E18C0A680A7A08C1B1F26
F78E1B1F108E1B3FA684A7A43088E031
A8E08C18DF26F186FF8E1A20A7843088
E08C18A026F639":TT=31700:GOSUB55
00
5130 ML$="8E0C01108E0C00A680A7A0
8C0CDF26F78E0CDF108E0CFFA684A7A4
3088E031A8E08C0BDF26F139":TT=317
60:GOSUB5500:ML$="BDB3ED1F01108E
222CC607A6A0A7805A26F930881931A8
19108C25922DEB39":TT=31800:GOSUB
5500
5150 ML$="BDB3ED1F01108E2337C601
A6A0A7805A26F930881F31A81F108C25
172DEB39":TT=31850:GOSUB5500:ML$
="BDB3ED1F01108E2335C601A6A0A780
5A26F930881F31A81F108C25152DEB39
":TT=31900:GOSUB5500:ML$="BDB3ED
1F01108E23BAC606A6A0A7805A26F930
881A31A81A108C253E2DEB39"
5175 TT=31950:GOSUB5500:ML$="BDB
3ED1F01108E27A0C608A6A0AA84A7805
A26F730881831A818108C2BA72DE939"
:TT=32000:GOSUB5500:ML$="393939"
:TT=32050:GOSUB5500:ML$="8E0A811
08E0A80A680A7A08C123F26F78E123F1
08E125FA684A7A43088E031A8E08C0A9
F26F139":TT=32100:GOSUB5500
5210 ML$="8E0E81108E0E80A680A7A0
8C123F26F78E123F108E125FA684A7A4
3088E031A8E08C0E9F26F139":TT=321
```

```
50:GOSUB5500:ML$="8E19E1108E19E0
A680A7A08C1B1F26F78E1B1F108E1B3F
A684A7A43088E031A8E08C19DF26F139
":TT=32200:GOSUB5500
5230 ML$="BDB3ED1F01108E26A1C609
A6A0A7805A26F930881731A817108C2A
C92DEB39":TT=32250:GOSUB5500:ML$
="BDB3ED1F01108E26AAC609A6A0A780
5A26F930881731A817108C2AD22DEB39
":TT=32300:GOSUB5500:ML$="BDB3ED
1F01108E26B2C609A6A0A7805A26F930
881731A817108C2ADB2DEB39"
5255 TT=32350:GOSUB5500:ML$="BDB
3ED1F01108E2BC1C607A6A0A7805A26F
930881931A819108C2EA72DEB39":TT=
32400:GOSUB5500:ML$="BDB3ED1F011
08E2300C60FA6A0A7805A26F93088113
1A811108C252F2DEB39":TT=32450:GO
SUB5500
5280 ML$="BDB3ED1F01108E2580C60F
A6A0A7805A26F930881131A811108C27
6F2DEB39":TT=32500:GOSUB5500:ML$
="BDB3ED1F01108E1E71C602A6A0A780
5A26F930881E31A81E108C21F22DEB39
":TT=32550:GOSUB5500:RETURN
5500 FORI=1TOLEN(ML$)STEP2:AA$=M
ID$(ML$,I,2):AA=VAL("&H"+AA$):PO
KETT+VV,AA:VV=VV+1:NEXTI:VV=0:RE
TURN
6300 TM=0:PMODE3,1:COLOR3,1:TY=1
78:FORTX=40TO240STEP4:LINE(TX,TY
)-(TX,TY+6),PSET:TM=TM+1:NEXTTX:
PMODE4,1:DRAW"C1S4BM4,184;XTM$;"
:RETURN
6400 TX=TX-4:PMODE3,1:COLOR4,0:L
INE(TX,TY)-(TX,TY+6),PSET:S1=S1+
5:TM=TM-1:IFTM<1THEN6900
6425 PMODE4,1:COLOR1,0:PLAYP3$:R
ETURN
6500 PMODE4,1:COLOR1,0:IFNS<=0TH
EN20000
6504 LINE(8,7)-(240,24),PRESET,B
F:X=8:Y=8:FORI=1TONS:PUT(X,Y)-(X
+26,Y+15),A1,OR:X=X+26:NEXTI:RET
URN
6900 IFLV>1THEN7600ELSEPOKE65494
,0:POKE65314,170:PLAY"01V30L255T
255FE02A":POKE65314,255:FORI=1TO
200:NEXTI
6905 CLS:SCREEN0:PRINT@32,Z$;:PR
INT@448,Z$;:NS=NS+1:PRINT@96,"YO
U HAVE COMPLETED LEVEL ONE.":PRI
NT" BONUS 500!!!":S1=S1+500:PRIN
T:PRINT"YOUR SCORE NOW IS"S1".":
PRINT"YOU HAVE"NS"ROVERS LEFT.":
POKEPK,0:LV=2:PMODE4,1:COLOR1,0:
PCLS0:GOSUB6500:GOSUB1100:GOSU
7000 CLS:DEFUSR0=31600:DEFUSR1=3
1650:DEFUSR2=31700:DEFUSR3=31760
:DEFUSR4=31800:DEFUSR5=31850:DEF
USR6=31900:DEFUSR7=31950:POKE654
94,0:CLS:SCREEN0,0:PRINT@227,"PR
ESS <ENTER> TO CONTINUE"
7074 IFINKEY$<>CHR$(13)THEN7074
7090 BG=5287:BX=5:JP=0:PC=0:E1=0
:E2=0:E3=0:POKEPK,0:PMODE4,1:COL
OR1,0:SCREEN1,1
7100 J0=JOYSTK(0):IFJP=1THEN7108
ELSEIFJ0<20THEN7200ELSEIFJ0>43TH
EN7225
7108 PC=1-PC:IFE1=1THENGOSUB7410
7110 U=USR0(BG):U=USR2(0):U=USR3
(0):IFJP=1THEN7310ELSEIF(PEEK(&H
FF00)AND1)=0THEN7300
7124 IFE1=1THENGOSUB7410ELSEIFRN
D(24)=1THENGOSUB7400
7128 IFE2=1THENGOSUB7440
7130 U=USR2(0):ML=ML+1:IFML>8AND
E1=0THENIFRND(INT(TM/3))=1THENGO
SUB7500:ML=0
7140 IFPEEK(BG+1061)=85THEN7905
7180 TI=TI+1:IFTI>10THENTI=0:GOS
UB6400
7190 PLAY"05V31P255":GOTO7100
7200 IFBX<1THEN7108ELSEBX=BX-1:B
G=BG-1
7205 U=USR1(BG+7):GOTO7108
7225 IFBX>16THEN7108ELSEBX=BX+1:
BG=BG+1
7230 U=USR1(BG-1):GOTO7108
7300 JP=1:J2=0:J1=0:PLAYP2$
7310 IFJ2=1THEN7320ELSEJ1=J1+1:I
FJ1>3THENJ2=1ELSEBG=BG-128
7315 GOTO7330
7320 J1=J1-1:IFJ1<1THEN7350ELSEB
G=BG+128
7330 GOTO7124
7350 J2=0:JP=0:GOTO7130
7400 E1=1:EP=3739:U=USR4(EP)
7410 EP=EP-1:IFEP<3711THEN7420EL
SEU=USR4(EP)
7412 IFE2=1THEN7415ELSEIFRND(18)
=1THEN7430
7415 RETURN
7420 E1=0:FORI=0TO6:U=USR1(EP+I)
:NEXTI:RETURN
7430 E2=1:E0=EP+514:PLAYP5$:EY=0
:U=USR5(E0)
7440 EY=EY+6:E0=E0+192:IFEY>42TH
EN7460
7445 IFPC=1THENU=USR5(E0) ELSEU=
USR6(E0)
7448 P1=PEEK(E0+512):IFP1<>85THE
N7905
7450 RETURN
7460 EI=E0+448:U=USR1(E0-448):E2
=0:GOSUB7505:PLAYP4$:RETURN
7500 EI=6299
7505 U=USR7(EI):RETURN
7600 IFLV>2THEN8600ELSEPOKE65494
,0:POKE65314,170:PLAY"01V30L255T
255FE04E":POKE65314,255:FORI=1TO
200:NEXTI
7605 NS=NS+1:S1=S1+1000:CLS:SCRE
EN0:PRINT@32,Z$;:PRINT@448,Z$;:P
RINT@96,"YOU HAVE COMPLETED LEVE
L TWO":PRINT" BONUS 1000!!!":PRI
NT:PRINT"YOU HAVE"NS"ROVERS LEFT
.":POKEPK,0:LV=3:PMODE4,1:COLOR1
,0:PCLS0:GOSUB6500:GOSUB1200:GOS
UB6300:GOTO8000
7905 POKE65494,0:PLAYPP$:CLS:SCR
EEN0,0:NS=NS-1:PRINT@32,Z$;:PRIN
T@448,Z$;:PRINT@99,"YOU HAVE"NS"
BUGGIES LEFT.":PRINT@163,"YOUR S
CORE IS"S1"POINTS.":PMODE3,1:COL
OR2,1:LINE(0,69)-(255,149),PSET,
BF:COLOR4,1:LINE(0,150)-(255,170
),PSET,BF:PMODE4,1:COLOR1,0:FO
7970 POKE65494,0:PRINT@387,"PRES
S <ENTER> TO CONTINUE"
7975 IFINKEY$=CHR$(13)THEN7980EL
SE7975
7980 CLS:POKEPK,0:GOTO7090:STOP
8000 CLS:DEFUSR0=31000:DEFUSR1=3
1050:DEFUSR2=32100:DEFUSR3=32150
:DEFUSR4=32200:DEFUSR5=32250:DEF
USR6=32300:DEFUSR7=32350:DEFUSR8
=32400:POKE65494,0:CLS:SCREEN0,0
:PRINT@227,"PRESS <ENTER> TO CON
TINUE"
8074 IFINKEY$<>CHR$(13)THEN8074
8090 JP=0:BG=5574:BX=5:CT=0:E1=0
:E2=0:F3=0:U=USR4(BG)
8098 PMODE4,1:COLOR1,0:SCREEN1,1
:POKEPK,0:GOSUB6500
8100 J0=JOYSTK(0):U=USR3(0):U=US
R4(0):IFPC=1THENU=USR2(0)
8104 IFJP=1THEN8108ELSEIFJ0<20TH
EN8200ELSEIFJ0>43THEN8225
8108 PC=1-PC:IFF3=1THENGOSUB8550
8110 U=USR0(BG):IFE1=1THENGOSUB8
410ELSEIFE2=1THENPE=PEEK(F2+485)
:IFPE<255THEN8910
8120 IFJP=1THEN8310ELSEIF(PEEK(&
HFF00)AND1)=0THEN8300
8130 U=USR4(0):CT=CT+1:IFCT<7THE
N8180
8135 IFE2=1THENGOSUB8510ELSEIFE1
=0ANDRND(20)=1THENGOSUB8500
8140 IFE2=1THEN8180ELSEIFE1=1THE
NGOSUB8410ELSEIFRND(20)=1THENGOS
UB8400
8180 PE=PEEK(BG+357):IFPE>0THEN8
910
8185 TI=TI+1:IFTI>9THENTI=0:GOSU
B6400
8190 PLAY"03V31P255":GOTO8100
8200 IFBX<1THEN8108ELSEBX=BX-1:B
G=BG-1
```

```
8205 U=USR1(BG+7):GOTO8108
8225 IFBX)16THEN8108ELSEBX=BX+1:
BG=BG+1
8230 U=USR1(BG-1):GOTO8108
8300 JP=1:J2=0:J1=0:PLAYP2$
8310 IFJ2=1THEN8320ELSEJ1=J1+1:I
FJ1)7THENJ2=1ELSEBG=BG-128
8315 GOTO8330
8320 J1=J1-1:IFJ1<1THEN8350ELSEB
G=BG+128
8330 GOTO8130
8350 J2=0:JP=0:GOTO8130
8400 E1=1:F1=5464:FX=27
8410 IFPC=1THEN8450ELSEU=USR5(F1
):F1=F1-1
8420 GOTO8480
8<50 U=USR6(F1):F1=F1-1
8480 FX=FX-1:IFFX<1THENE1=0:U=US
R7(F1):POKE6528,170:CT=0
8490 RETURN
8500 E2=1:F2=5728:X2=2:FF=0:F3=0
8510 IFF3=1THEN8550ELSEFF=FF+RND
(3):IFFF)26THENFF=0:F3=1:GOTO855
0
8514 IFPC=1THENF2=F2-32ELSEF2=F2
+32
8530 U=USR8(F2):GOTO8580
8550 F2=F2+1:FF=FF+1:IFFF)27THEN
8590ELSEU=USR8(F2)
8580 RETURN
8590 E2=0:U=USR7(F2-416):CT=0:F3
=0:RETURN
8600 POKE65494,0:POKE65314,170:P
LAY"01V30L255T255FE02FE03FE04FE0
5FE04FE03FEFEFEF":POKE65314,255
8610 CLS:FORI=1TO200:NEXTI:SCREE
N0,0:PRINT@32,Z$;:PRINT@448,Z$;:
PRINT@96,"YOU HAVE COMPLETED LEV
EL THREE.":PRINT" BONUS 1500!!!"
:S1=S1+1500:PRINT:PRINT"YOUR SCO
RE IS NOW"S1"POINTS.":PRINT"YOU
WILL GO TO THE FINAL LEVEL":PRIN
T"WITH"NS"BUGGIES.":GOTO9000
8910 PLAYPP$:FORI=1TO200:NEXTI:S
CREEN0,0:CLS:PRINT@32,Z$;:PRINT@
448,Z$;:PRINT@98,"YOU HAVE"NS"BU
GGIES LEFT.":PRINT@164,"YOUR SCO
RE IS"S1".":PMODE4,1:COLOR1,0:LI
NE(0,88)-(255,155),PRESET,BF:NS=
NS-1:POKE65494,0:CLS:PRINT@227,"
PRESS <ENTER> TO CONTINUE"
8984 IFINKEY$<>CHR$(13)THEN8984
8990 CLS:JP=0:BG=5574:BX=5:CT=0:
E1=0:E2=0:F3=0:GOTO8098
9000 POKEPK,0:GOSUB3300:GOSUB130
0:GOSUB6300:GOSUB6500:DEFUSR0=31
000:DEFUSR1=31050:DEFUSR2=32450:
DEFUSR3=32500:DEFUSR4=32550:JP=0
:BG=5252:BX=3:RP=6176:E1=0:E2=0:
RQ=RP-3072
9098 PMODE4,1:COLOR1,0:SCREEN1,1
:EX=3183:E1=0:E2=0:EY=5583
9100 J0=JOYSTK(0):IFJ0<20THEN920
0ELSEIFJ0)43THEN9225
9108 PC=1-PC:U=USR0(BG):GOSUB940
0:IFPC=1THENU=USR2(RP):U=USR2(RQ
) ELSEU=USR3(RP):U=USR3(RQ)
9122 IFPC=1THENRQ=RQ+32
9180 PE=PEEK(BG+99):IFPE)0THEN99
00
9185 PE=PEEK(BG+101):IFPE)0THEN9
900
9190 PLAY"P255":GOTO9100
9200 IFBX<1THEN9108ELSEBX=BX-1:B
G=BG-1
9205 U=USR1(BG+7):GOTO9108
9225 IFBX)16THEN9108ELSEBX=BX+1:
BG=BG+1
9230 U=USR1(BG-1):IFBG=5261THEN9
500
9245 GOTO9108
9400 IFE1=1THEN9450
9410 EX=EX+192:U=USR4(EX):E2=E2+
1:IFE2)8THENE1=1
9430 GOTO9470
9450 EX=EX-192:U=USR4(EX):E2=E2-
1:IFE2<1THENE1=0
9470 IFE1=1THEN9485
9475 EY=EY-192:U=USR4(EY):RETURN
9485 IFE2<3THENU=USR4(EY) ELSEEY
=EY+192:U=USR4(EY)
9490 RETURN
9500 U=USR0(BG):POKE65314,170:PL
AY"V31L255T255DEFEDGDE":POKE6531
4,255:POKE65494,0:PLAY"03V30L255
T255DEFFDEEDFADFADFADF":CLS:FORI
=1TO300:NEXTI:PRINT@32,Z$;:PRINT
@448,Z$;:PRINT@96,"CONGRATULATIO
NS!!!":PRINT"YOU HAVE MADE IT TO
":PRINT"MOON BASE AMPHIBIA!!!"
9590 GOTO20
9900 POKE65494,0:PLAYPP$:FORI=1T
O200:NEXTI:SCREEN0,0:CLS:PRINT@3
2,Z$;:PRINT@448,Z$;:PRINT@98,"YO
U HAVE"NS"BUGGIES LEFT.":PRINT@1
64,"YOUR SCORE IS"S1".":GOSUB130
0:JP=0:BG=5252:BX=3:RP=6176:E1=0
:E2=0:RQ=RP-3072:NS=NS-1:POKE654
94,0:CLS:PRINT@227,"PRESS <ENT
9984 IFINKEY$<>CHR$(13)THEN9984
9990 CLS:POKEPK,0:GOTO9098
20000 POKE65494,0:FORI=1TO200:NE
XTI:CLS:SCREEN0,0:PRINT@32,Z$;:P
RINT@448,Z$;:PRINT@128,"YOUR FIN
AL SCORE IS"S1:PRINT@352:INPUT"P
LAY AGAIN (Y/N)";A$:L$=LEFT$(A$,
1):IFL$="Y"THENRUN
```

Listing 2:

```
7918                ØØØØØ         ORG   31ØØØ
                    ØØØØ5 *THIS ROUTINE PUTS THE BUGGY GRAPHIC ON THE GRAPHIC SCREEN
                    ØØØØ7 *AT THE LOCATION SPECIFIED THROUGH THE BASIC PROGRAM.
7918 BD     B3ED    ØØØ1Ø START   JSR   $B3ED        *GET PARAMETER FROM BASIC
791B 1F     Ø1      ØØØ2Ø         TFR   D,X          *STORE PARAMETER IN X
791D 1Ø8E 1EØ1      ØØØ3Ø         LDY   #$1EØ1       *PUT LOCATION OF BUGGY GRAPHIC IN Y
7921 C6     Ø7      ØØØ4Ø         LDB   #7           *LENGTH IN BYTES OF GRAPHIC
7923 A6     AØ      ØØØ5Ø LOOP1   LDA   ,Y+          *LOAD BYTE FROM BUGGY
7925 A7     8Ø      ØØØ6Ø         STA   ,X+          *PLACE BYTE ON SCREEN
7927 5A             ØØØ7Ø         DECB               *COUNT
7928 26     F9      ØØØ8Ø         BNE   LOOP1        *IF NOT ZERO, THEN GO BACK
792A 3Ø     88 19   ØØØ9Ø         LEAX  $19,X        *NEXT LINE
792D 31     A8 19   ØØ1ØØ         LEAY  $19,Y        *NEXT LINE
793Ø 1Ø8C 2187      ØØ11Ø         CMPY  #$2187       *END OF GRAPHIC?
7934 2D     ED      ØØ12Ø         BLT   LOOP1        *NO,RETURN
7936 39             ØØ13Ø         RTS
            ØØØØ    ØØ14Ø         END
ØØØØØ TOTAL ERRORS
```

TAKING BASIC TRAINING

16K ECB

Mastering The DRAW Statement

By Joseph Kolar

We have been working with the DRAW statement and proving to one and all that beginners can master it without special effort. Today, while you are brimming with confidence, we will create an interesting design.

As you work through this tutorial, do not hesitate to stop and run off on a tangent when you get an idea. Pursue it! You can always return to this article and pick up where you left off. Your ideas are too precious to relegate to the back of your mind. Truly, a discovery you make on your own is more important than those which you may glean from this tutorial.

Key in lines 0, 10, 500 from Listing 1. Look at Figure 1. It is a design worked out on graph paper. Each box is one unit long. The star is the point of origin. You must begin drawing the design at some location.

At the starting point, go up and to the right two units, E2. Following along, you can readily see that to advance further, you must go right two units, R2. The next line traces down and to the right. You might go two units, F2. If you do, you have a choice of traveling in three directions — up and to the right, E4; down and to the right, F4; down and to the left, G2.

Although you may create the complete design using any of the three routes, you are likely to choose E4 followed by R8F4E2R2F2G2L2H2G4L8H4G2L2H2, returning to the point of origin (Line 15 of Listing 1).

If you began with E2R2, rather than stopping at F2, you could have continued four more units in the same direction, F4. Put another way, F2F4 or, simply, but no less correct, F6. Using this alternate routing, R8E6R2F2G2L2H6L8G6L2H2, brings us to the starting point (Line 20).

Line 25 creates the same design another way. First the left unit is created (trace it out). BR6 moves us to start creating the large central unit. BR16 moves us over to create the right unit. In order to get back to the starting location, we must move left as many 'B' units as we advanced to the right, BR6+BR16=BR22. The opposite direction of BR22 is BL22.

Key in Line 15. Type RUN, examine, then hit the BREAK key. Ditto for Line 20.

Since we plan to use the design repeatedly, we can avoid keying the same line over and over again, which is boring and subject to error. We put the design into a string and call it with the variable, A$. For practice, you could do this with Line 15 or 20. The instructions to CoCo are omitted in Line 25 and the directions are enclosed in quotation marks (" "). To make the line appear on the screen, a new line, 30, is created. This line has the instructions to CoCo within quotation marks (" "), telling CoCo where you want the design to appear and in what size. Concatenating '+,' A$ tells CoCo what design to display.

Key in Line 30, type RUN, check and then hit the BREAK key. Key in Line 40. This demonstrates how A$ is used to put a design element above the three on the screen and one below them. Now, we have five designs in a column.

Suppose we add a design at the left side. Key in: 50 DRAW"BM50,70"+A$ and type RUN. This location was a lucky guess! It connects nicely. We'll make two more units underneath which give us a total of three design units. Then we'll make a unit farther to the left. This means the next row will have two units side by side; one beginning at horizontal, 0, and the other at 50. Hit the BREAK key and add to the end of Line 50 :DRAW"BM0,90"+A$+"BR25"+A$ and RUN.

If you left out +"BR25", you would have lost the second unit. Delete it and see! Do you know why?

The first A$ at "BM0,90" ended at 49,90. If you merely added another +A$, it would print in the same location. How could you solve it? Move the starting location over by inserting +BR25 between the A$s.

Can you think of another way? If you edited out BL22 from the end of Line 25, and RUN, you will note that it must be pushed to the right to avoid overlap. If you added BR3 to the end of Line 25, it would be just right. Try it and see! Then hit the BREAK and change BR3 back to BL22.

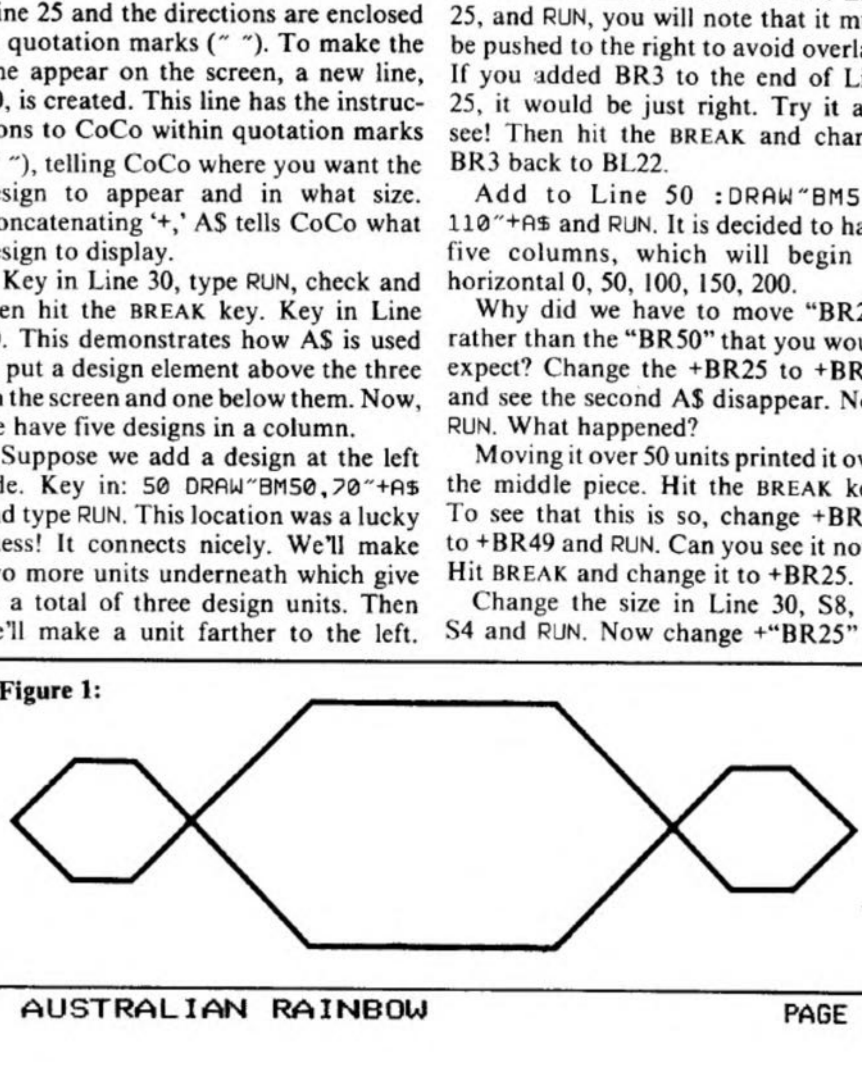
Add to Line 50 :DRAW"BM50,110"+A$ and RUN. It is decided to have five columns, which will begin at horizontal 0, 50, 100, 150, 200.

Why did we have to move "BR25" rather than the "BR50" that you would expect? Change the +BR25 to +BR50 and see the second A$ disappear. Now RUN. What happened?

Moving it over 50 units printed it over the middle piece. Hit the BREAK key. To see that this is so, change +BR50 to +BR49 and RUN. Can you see it now? Hit BREAK and change it to +BR25.

Change the size in Line 30, S8, to S4 and RUN. Now change +"BR25" to

Figure 1:



+"BR50" and RUN.

In other words, size S8 is twice the size of S4, so the design needs to be moved over half the number of units required in S4. We don't really care *why* CoCo does what it does. All we want to know is *what* it does.

Hit the BREAK key and change Line 30 back to S8 and Line 50 back to +"BR25".

Whether or not you understand what the correct shift is matters little. You can always work it out by trial and error to get the solution.

Without peeking at the listing, can you add four units to the right side to make it a symmetrical display? Try it and see (Line 60).

Suppose you wanted to make five design units across the top of the screen at horizontal 10. Line 70 shows one way.

Suppose you wanted to make a similar five units at the bottom to keep the symmetry of the display. Line 80 shows you a second way.

The technique in Line 80 is longer, but it is easy to figure out. Although not as elegant as the solution in Line 50, it is perfectly adequate. Just as long as the result is satisfactory, any method is fine.

Line 70 was a little tricky. O/S (Out of String Space) error message forced us to insert a pair of DRAW statements to overcome this problem.

This could be partially overcome by adding 5 CLEAR 500. When planning to use strings, play it safe and CLEAR 500.

Finally, in Line 70, if you change the first ':' to '+' and delete DRAW, you will find it OK. But, if, in addition, you change the second ':' to '+' and delete DRAW, you will get an L/S (String Too Long) error message.

It is time to introduce the 'A' option of DRAW. The 'A' option allows you to draw a design around a point (location). A0 prints the design as you conceived it; A1 prints it from the same starting point but 90 degrees clockwise, perpendicular to A0 but above it. A2 prints a further 90 degrees clockwise, radiating in the opposite direction of A0; A3 places it 90 degrees further along in a clockwise direction to be opposite the perpendicular, A1

This may sound confusing but it is well worth the effort to add this feature to your store of useful knowledge.

Key in lines 0, 5, 10, 20, 500 from Listing 2. Notice that Line 20 draws the same element used in Listing 1. If you trace it out on graph paper, you will discover that the starting point is at the opposite end.

The object is to give you plenty of practice creating a familiar design and stimulate you to go off into uncharted territory.

Key in the first part of Line 30: 30DRAW"S4BM128,96"+A$ and RUN. You should have the same old design unit. If in doubt, hit the BREAK key and temporarily change S4 to S8. Now RUN and you will be able to see it better. When your design is OK, change back to S4. After S4 insert A0, the first 'A' option. This is the default option. You get it whether you like it or not, unless you advise CoCo otherwise. RUN.

Now, change A0 to A1 and RUN. See? It is perpendicular. Do the same to get A2 and A3 to see all four possible options at work. If you have trouble visualizing the rotation, temporarily add:

```
25LINE(128,0)-(128,191),PSET
```

Run through the 'A' options a few times to observe what is what. When you are satisfied, delete Line 25. Did you notice that A0 radiates out to the left of the starting location; A1 radiates upward; A2 radiates to the right and A3 radiates downward in this example?

Hit the BREAK key. Make sure that Line 30 is A0 and add at the end of it, +"A1"+A$. We told CoCo, in addition to the original design element, we also wanted one radiating upwards. Now, RUN and hit the BREAK key.

Let's tell CoCo to add the next element to radiate to the right. Add to Line 30 +"A2"+A$, RUN and hit the BREAK key. Now, add and check the last option. Did you note when you instructed CoCo to change the orientation of the design element that the information must be enclosed in quotation marks (" ")?

Caution: A0 is not necessarily in the same quarter (radiating left), but could be in any location depending on the case in hand. Anyway, it isn't a bad design. What else can we do?

You could create more designs on the blank parts of the screen. Hint: Choose an arbitrary set of location points, BMx,y and after you create the entire design, move it over to its final resting place by trial and error.

Why do I frequently insist on trial and error? According to "Kolar's Law," no matter how well you calculate a plot, it rarely is correct because you are apt to make changes, modifications or enhancements, throwing your figures awry. So, you may just as well ease your design into its berth. Visualize a bunch of tugs nudging and pushing an ocean liner into its berth. Push a little this way. Nudge a bit that way. Ease it into its dock.

Key in Line 50. We want to superimpose the same enlarged design over our creation. RUN and check. If you are curious, you could add one 'A' option at a time. Note that once you use an 'A' option, every time you change options in each new DRAW line you must indicate an option, even A0, the default option.

To illustrate the concept, delete A0 in Line 50 and RUN. Sorry about that! What happened was that the last instruction CoCo received was A3 at the end of Line 40. It began Line 50 with the downward design; then the upward; the right; and again, downward. **It created A3, the downward element, twice. Of course, you could change A3 to A0 in Line 50 and correct it. Now, if you RUN, you will see that the second downward element is facing left. But, isn't it easier to insert the A0 in the first place?**

At any rate, we have a pleasing design. But, since we superimposed S12 units over S4, what do you say we superimpose S8 on the whole mess and see what we come up with?

Key in Line 60 and RUN. It is really a repetition of the design in Figure 1 rotated in four directions in three sizes.

As a point of information, the instructions to CoCo could be in any order. Line 60 could have been written "BM128,96S8A0C1", just so all the hot scoop was enclosed within quotation marks (" "). C1 is the default color and is included just to illustrate the point.

Line 40 was not used. It was included for you to substitute it for A$ and cycle through the 'A' options to reinforce in your mind what happens when the basic element is radiated right instead of left. Notice that what was A2 using A$ is now A0 using B$.

This tutorial should have given you food for thought. You are invited to get out the old graph paper, plot out your own design element and run it through its four 'A' option paces. Create! Experiment!

Listing 1: LISTING1

```
0 'LISTING1
5 CLEAR 500
10 PMODE4,1:PCLS:SCREEN1,1
15 DRAW"S8BM100,70E2R2F2E4R8F4E2
R2F2G2L2H2G4L8H4G2L2H2"
20 DRAW"S8BM100,90E2R2F6R8E6R2F2
G2L2H6L8G6L2H2"
```

```
25 A$="E2R2F2G2L2H2BR6E4R8F4G4L8
H4BR16E2R2F2G2L2H2BL22"
30 DRAW"S8BM100,110"+A$
40 DRAW"BM100,50"+A$:DRAW"BM100,
130"+A$
50 DRAW"BM50,70"+A$:DRAW"BM0,90"
+A$+"BR25"+A$:DRAW"BM50,110"+A$
60 DRAW"BM150,70"+A$:DRAW"BM150,
90"+A$+"BR25"+A$:DRAW"BM150,110"
+A$
70 DRAW"BM0,10"+A$+"BR25"+A$:DRA
W"BR25"+A$+"BR25"+A$:DRAW"BR25"+
A$
80 DRAW"BM0,170"+A$:DRAW"BM50,17
0"+A$:DRAW"BM100,170"+A$:DRAW"BM
150,170"+A$:DRAW"BM200,170"+A$
500 GOTO500
```

Listing 2: LISTING2

```
0 'LISTING2
5 CLEAR500
10 PMODE4,1:PCLS:SCREEN1,1
20 A$="H2L2G6L8H6L2G2F2R2E6R8F6R
2E2"
30 DRAW"S4A0BM128,96"+A$+"A1"+A$
+"A2"+A$+"A3"+A$
40 B$="E2R2F6R8E6R2F2G2L2H6L8G6L
2H2"
50 DRAW"S12A0BM128,96"+A$+"A1"+A
$+"A2"+A$+"A3"+A$
60 DRAW"S8A0BM128,96"+A$+"A1"+A$
+"A2"+A$+"A3"+A$
500 GOTO500
```

Listing 3: SHUTTLE

```
0 'SHUTTLE
10 '(C) 1984, J. KOLAR
30 PMODE3:PCLS:PMODE4
40 A=90:B=86:R=76:P=1.70
50 DIM S(7),T(7)
60 CIRCLE(8,8),8,1:CIRCLE(7,7),8
,1
61 DRAW"BM8,8NBU3L6U5R10D10L10U5
"
70 GET(0,0)-(16,16),S,G
72 CIRCLE(38,8),6,1:PAINT(40,8),
1,1
73 GET(30,0)-(46,16),T,G
80 PCLS:SCREEN1,1
90 FOR Q=.15 TO .05 STEP -1
100 FORZ=1TO6.55 STEPQ
110 C=Z:C=40-(C)*P*270-R^2
115 K=LOG(C^2)*COS(R)/COS(R^2)
120 X=INT(A-6+R*COS(C)):Y=INT(B-
8+R*SIN(K))
130 PUT(X+36,Y+10)-(X+52,Y+26),S
,OR
135 PUT(X+36,Y+10)-(X+52,Y+26),T
,AND:SOUND100,1
140 X=INT(A-6+R*SIN(K)):Y=INT(B-
8+R*COS(C))
150 PUT(X+36,Y+8)-(X+52,Y+23),S,
 OR
155 PUT(X+36,Y+8)-(X+52,Y+23),T,
AND:SOUND100,1
160 NEXT Z,Q
170 PLAY"V2003L8EEFFAB04L16CC03F
AL8FFAAL4FEL2CP4V25L8FFAAB04CL16
EEDDL8CC02AAL4FEL2CP4V2002L8EEFF
AA03V25L16CC02BAV20L8FFDDL4CV150
1B02DL2C"
180 PCLS:GOTO90
```

TURN OF THE SCREW

Switching Double-Sided Disks

By Tony DiStefano

It was great seeing the whole RAINBOW gang at the Chicago RAINBOWfest in May. That made my first RAINBOWfest anniversary. There were a lot of new products to be seen. Fancy software, new and improved hardware, and a lot of new faces.

These get-togethers are quite warm and friendly. I have gone to many computer shows, some for different kinds of computers and some that host just one brand. But, I have never seen one that came close to the atmosphere at a RAINBOWfest. I tip my hat to the CoCo Community.

Speaking of new products, look forward to seeing my new line of products, starting with the DISTO disk controller.

Clearing up Confusion

The topic of this month's project involves disk drives and disk controllers. There seems to be some confusion about disk drives being double-sided, double-density, single-sided, single-density, 96 or 48 tpi (tracks per inch) and the compatibility between them. Especially when you talk about OS-9.

I hope to clear all that up right here and now and follow it up with a small project to let you see just what side of the fence, uh . . . I mean disk, you are on.

I will start off by describing a diskette and a disk drive. First, a diskette is a form of media. It holds information — what information is up to you. A telephone numbers file, a game or two, your favorite word processor. All of these are files that make your computer function the way it does; this data has to be stored somewhere.

When the Color Computer first came out, the only mass storage available was a cassette recorder. Though the cassette recorder works well for music and speech, it was slow and not well-suited for computer work. A new form of mass storage had to be invented: The diskette was introduced. There are many kinds of diskettes on the market today, but I will limit this discussion to those that are compatibile with our lovable CoCo.

Without going into too much detail, the Radio Shack standard diskette used with the CoCo is a 5½-inch, single-sided, double-density, 35 tracks at 48 tpi, soft-sectored diskette. The Radio Shack Disk BASIC, disk operating system, drive and controller are made to comply with these standards. You can get more details on the DOS in the Disk BASIC manual. The Radio Shack controller is made to handle two or four drives, depending on what cable you have.

The disk drive itself connects to the controller via a 34-pin ribbon connector. Figure 1 shows the pin configuration of the "disk side" of the controller. As you can see from the diagram, four pins are used for selecting or activating up to four drives. Radio Shack drives differ from standard drives by the way they are selected. You see, all four pins on Radio Shack drives are connected together and the selecting is done by missing pins in the cable connector.

For example, to select Drive 2, the cable connector that is configured to be number 2 has the pins that correspond to drive numbers 0, 1 and 3 missing. That way, when another drive is selected, it won't affect that drive because that pin is missing.

There is one more interesting thing about the Radio Shack cable configuration. Drive 3 pin on the controller is not in the normal position for a standard drive. The normal position for a standard Drive 3 is pin #6, where Radio Shack chose to keep this pin empty.

Interestingly enough, though, the place they did put it is where the standard disk drive has its side select, pin #32 (for double-sided drives only). Since this pin is connected to the controller, it gives us access to the second side of a disk drive. All the hardware is there to use the second side, providing you have double-sided drives.

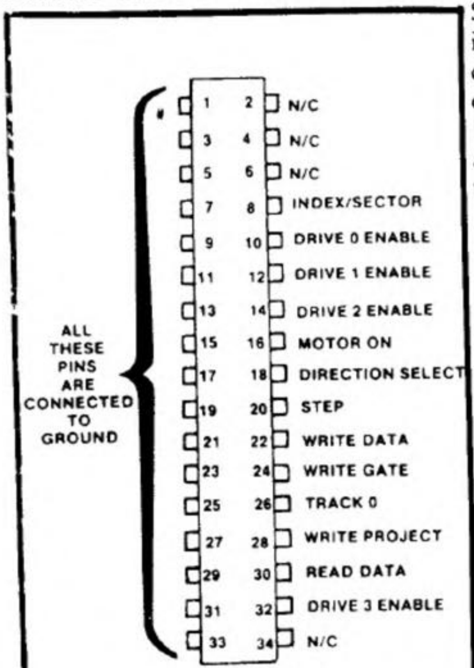


Figure 1
34-pin disk drive connector pinout

# Note: Looking at the DISK CONTROLLER edge connector, pin #1 is the top right-hand pin

Today, the price of double-sided drives is so low that in some cases it is cheaper to buy a double-sided drive from another company than it is to buy a single-sided drive from Radio Shack. More and more people already have them and are not using the second side because Disk BASIC does not allow them to do so. I will show you a couple of ways to access the second side. One is software and the other is hardware. Use the method that suits you best. Either way, you will want to build the project if you have double-sided drives.

The first thing to do to use the double-sided drive is make sure you have one! You must connect it to the Radio Shack controller. Remember, I said there were pins missing in the Radio Shack cable and that will give us problems.

The side select pin is only present on a four-drive cable, and then only on the fourth drive. You must add another connector for every double-sided drive you are adding to your system. (They are available at your nearest Radio Shack Computer Center.) The connector is a 34-pin edge card connector. If you don't know how to install it on your cable, ask your dealer to do it for you. Have him press the new connector about an inch and a half away from the old connector.

The disk drive now has to be configured to which drive number you want. There are jumpers inside the drive you must set. In the owner's manual of the drive there will be instructions on how to do that.

Now you have a double-sided drive on line, but you will still need a way to access it. The first way is in software. The way Disk BASIC selects the drive is by using four "mask" bytes. Each byte contains the necessary data in order to activate that drive number. There are four bits that control each of the output pins as seen in Figure 1.

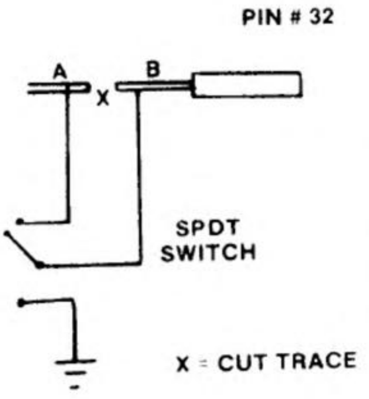


Figure 2
Double-sided drive switcher

In the controller, there is a memory-mapped byte that controls the output of these pins. It is at $FF40 or 65344. Try this:

```
POKE 65344,1
```

The select light on Drive 0 turned on.

Now try the values two, four and 64 instead of one. This will turn on drive numbers 1, 2 and 3, respectively. The last value of 64 activates Drive 3 (if you have four drives), but remember on our double-sided drive that is the side select. By changing the values on the four mask bytes we can access the second side of the drive. By changing the mask data, you can access the second side of the drive as another drive.

Example: If you have one single-sided Radio Shack drive and one double-sided drive with the right changes to the mask byte, you will have three drives on line. The Radio Shack drive is the first, the first side of the double-sided drive is the second and the second side of the double-sided drive is third. If you had two double-sided drives, it would be as if you had four separate drives. Two double-sided drives is the maximum you can have with Disk BASIC because there are only four mask bytes.

There are two ways to change the mask byte in software. One is to burn the new mask byte into an EPROM. The second is to use the 64K mode of the computer and make the changes in RAM. I'll leave that part up to you, but what I will do is tell you how to change the mask byte.

The four mask bytes correspond to four drives. Since we are using the last drive number as a side select, we can no longer use it as a drive but only as a side select. That leaves us with three other mask byte values. The values are 1, 2 and 4. The side select value is 64. Any combination of this will work (maximum of four).

First example: Your first drive is a Radio Shack single-sided drive. You want it to be Drive 0, so the value of the first mask byte is '1.' Your second drive is a double-sided drive; they will be Drive 1 and Drive 2. The second mask byte will be '2' and the third byte will be 65. The fourth byte will be untouched.

Second example: You have two double-sided drives. Drive 0 will be the normal side of the first drive; Drive 1 will be the normal side of the second drive. Drive 2 will be the second side

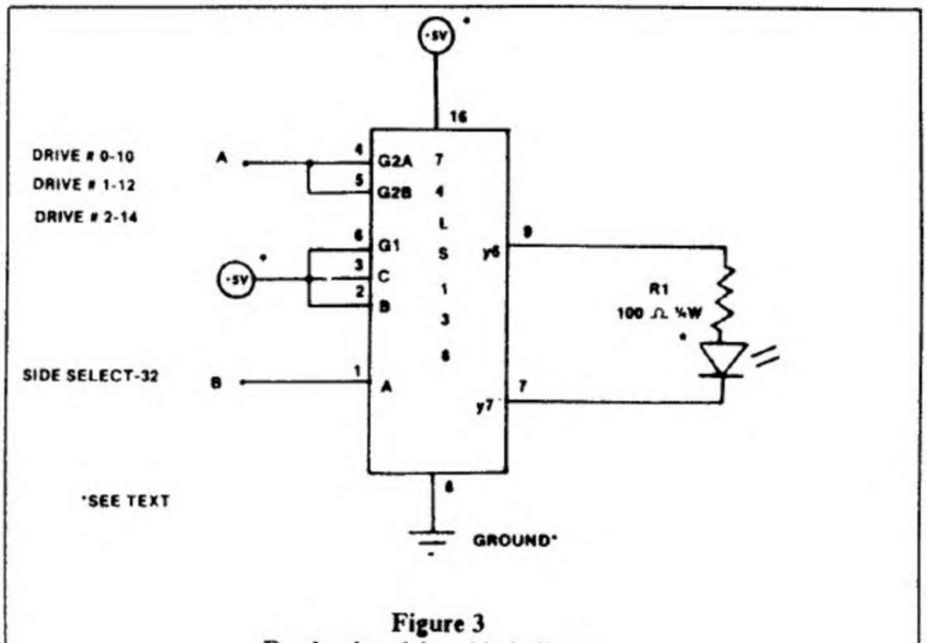


Figure 3
Dual color-drive side indicator

of the first drive and Drive 3 will be the second side of the second drive. The four mask bytes are 1, 2, 65, 66.

Radio Shack has two versions of DOS: 1.0 and 1.1. The memory address of the four mask bytes for DOS 1.0 is $D7AA (55210); the address mask bytes for DOS 1.1 is $D89D (55453), plus the next three bytes for the other three values.

If all that doesn't thrill you, you can select the other side by adding a small switch to your disk controller. Figure 2 shows how to hook up the switch to your controller. You must cut the foil between points 'A' and 'B.' Drill a suitable hole in the cover of the controller to mount the switch. When the switch is in the up position, the normal sides of all double-sided drives are accessed. When the switch is in the down position, the second side is accessed. Never change the switch when doing I/O to disk since it will ruin both sides. Again, remember, you must not use the fourth drive on a four-drive connector.

To some, it is easier to install the switch than to do it in software, but it is a little more difficult to manually flip the switch. In any case, visual cue as to what side of the disk you are really on is almost a necessary option.

Figure 3 is a schematic for a circuit that will tell you what side of the drive you are using by lighting a different color LED for each side. This circuit goes inside the disk drive and replaces the "active drive" select LED. The heart of the circuit is the Radio Shack Tri-Color LED (part #276-035). This LED glows one of three colors. We will be using only two of these colors, red and green. The circuit uses a 74LS138 decoder.

When no drive is selected, the two outputs used are logical level one and the LED is off. When the drive in question is selected, the 'A' (drive select) input goes low, therefore activating the chip. If the 'B' (side select) is high (first side of the drive), the Y7 output goes low. This will cause a positive voltage to appear across the Tri-Color LED which makes the LED glow red. If the 'B' input is low, the Y6 output goes low, in which case there will be a negative voltage across the LED. Then the LED will glow green. When the 'A' input is high (drive not selected) the chip is disabled and both Y7 and Y6 are high, the LED will be off. I put red as the first side because it is the color of a single-sided drive. That way when I see green, I automatically know I'm on the other side.

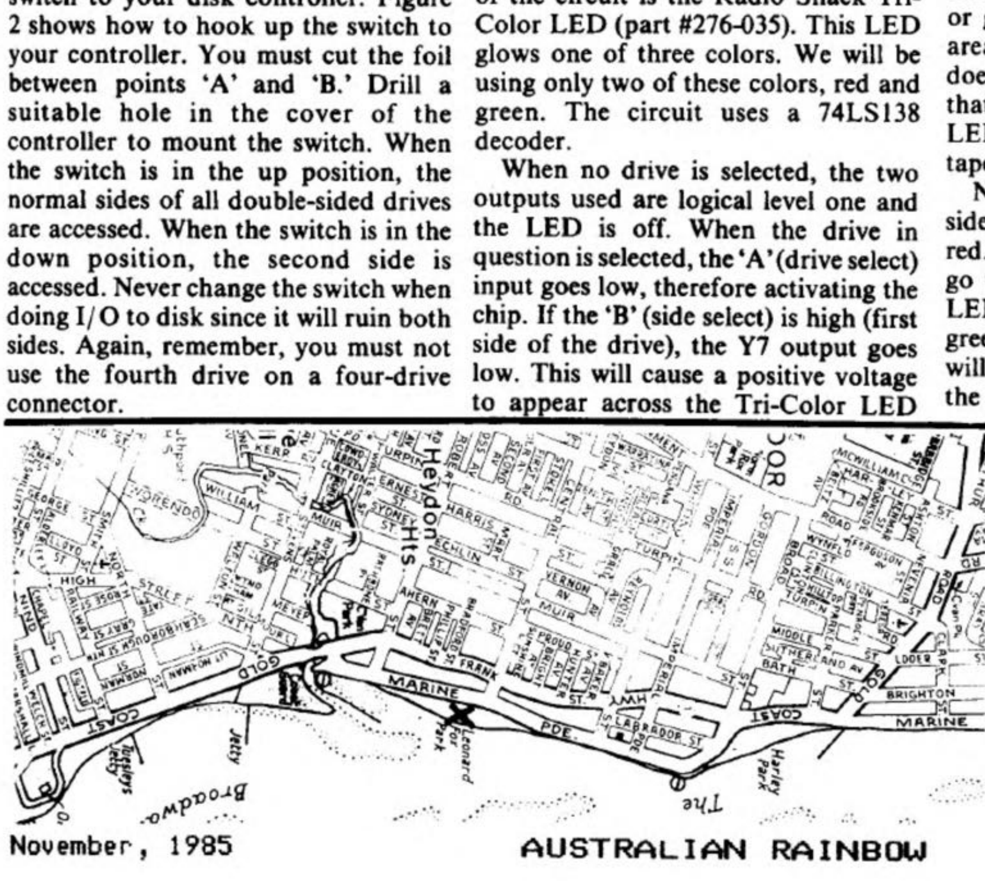
There are just a few things to consider when hooking up this circuit inside the drive. The first is where to get the five volts and ground needed to run the circuit. The easiest place to get a ground is pin #1 of the drive cable connector. Pin #1 is on the side of the connector that has all the pins connected together. They are all the odd-numbered pins. The drive connector pins are numbered on each end.

Five volts can be taken from the last pin of any 74LSXX chip. Use a volt meter to check the voltage. This is either pin #14 or #16 depending on how many pins there are on that chip.

The second thing to watch for is to make sure the 'A' input matches that of the drive selected. This means if the 'A' wire goes on Drive 0, make sure the drive configuration block is set to Drive 0, otherwise the LED will never light.

The actual construction of the circuit can be done on a small perf board. Tape or glue down the board in an unused area of the disk drive. Make sure it doesn't get in the way of the diskette that enters the drive. Remove the old LED. Replace it with the new one. Use tape or glue to hold it down.

Now, try the drive and access the first side of the drive. The LED should be red. If it is green, reverse the wires that go to the LED. When all is OK, the LED will glow red for the first side and green for the second side. This way you will always know which side of the drive the software is accessing.



BAR - B - QUE

Sat. 9th NOV. 1985

11.30 am.

B.Y.O. everything

' X ' marks the spot!!

Interfacing Machine Language With BASIC

By R. Bartly Betts

While the topic of this month's column is interfacing machine language programs with BASIC, it covers other territory as well. Along with another method of initializing machine language programs, it introduces a few ROM routines to use.

Finally, everything is tied up in a neat machine language program that lets you type a selected number of characters on the screen. The characters you type are in reverse video but, when you press ENTER, they flip back to regular video and are stored in a string variable that you specify.

Last month the feature program demonstrated one way to interface BASIC to machine language. The program this month shows you another way to interface BASIC with machine language, and also how to interface machine language with BASIC.

Push and Pull

But first, some lessons. Two new instructions for you to learn are PSHS and PULS. PSHS (Push a Register on the Hardware Stack) is used like a good filing system. In effect, issuing an instruction like PSHS Y is like filing the sales figures for yoyos in a file drawer. The next time you need to know how many yoyos are sold, open the drawer and pull the yoyo file.

Likewise, if you have some information in Register Y and are afraid it might get lost or destroyed if you leave it there, you can file its contents on the hardware stack with a PSHS Y instruction. In fact, PSHS can be used to save any number of .the registers. To test this, type Listing 1, then single-step through it to observe the results:

Listing 1: DEMO
(A demonstration of the PSHS and PULS instructions.)

```
ØØ1ØØ START    LDA     #9
ØØ11Ø          LDB     #8
ØØ12Ø          LDX     #7
ØØ13Ø          LDU     #6
ØØ14Ø          PSHS    A,B,X,U
ØØ15Ø          CLRA
ØØ16Ø          CLRB
ØØ17Ø          LDX#Ø
ØØ18Ø          LDU#Ø
ØØ19Ø          PULS    A,B,X,U
ØØ2ØØ DONE     SWI
ØØ21Ø          END
```

As you can see, this program does nothing but load values into four registers, then clears them to zero. Before clearing them, however, Line 140 causes them to be pushed onto the hardware stack. After the registers are clear, a PULS instruction is issued in Line 190 to restore the values. In this demonstration program, four registers are saved. PSHS can, however, save as many or as few registers as you wish, using only one PSHS instruction.

Reading the Registers

To see the program in action, enter the ZBUG mode by typing Z and ENTER. To begin single-stepping, type START, (type a comma after START). Press 'R' to see the condition of the registers. Register A contains the value '9'. To step to the next instruction, press the comma again. Again press 'R' to see the registers. Register A still contains '9' and Register B contains '8'.

Continue through the program by pressing the comma to advance. Examine the registers each time. You see the four registers A, B, X and U loaded with values. The subsequent PSHS instruction does not cause any change. Next, the registers are cleared. Finally the PULS instruction loads all the original values back into the four registers.

When using the PSHS and PULS instructions, remember that you must keep track of the order in which registers are loaded onto the stack. The stack operates in a "last in, first out" manner. Therefore, if you push the value in Register Y onto the stack, followed by the value in Register X, a PULS Y instruction does not load Y's original value back into Y, it loads the value from X.

Relating to ROM

Last month I discussed writing to the text screen by storing character values directly into the screen memory. This column introduces you to a different approach.

In BASIC, characters are displayed on the video screen by a subroutine in

ROM (Read Only Memory). ROM has hundreds of routines that BASIC uses to do its work. For instance, there are routines to turn on your cassette recorder or disk drives. There are routines to do arithmetic. There are routines to read key presses when you type, and there are routines to display characters on the screen. A great many ROM routines are available for you to use from machine language. All you need to know is what they do and where they are.

Listing 2 makes use of two of these subroutines to 1) get a keypress and 2) display the generated character on the video screen. It also introduces you to the USR method of interfacing to BASIC.

The origin of this program is set to $E00 to be compatible with all computers. The next instruction, in Line 110, jumps to a ROM subroutine that loads Register D with the value of the argument, or parameter, included in the BASIC USR command. For instance, if you issue the command A=USRØ(1Ø1), Line 110 stores the value 101 in the 'D' register. Therefore, in the case of this program, the characters you type begin at screen location 101.

"A great many ROM routines are avqilable for you to use from machine language. All you need to know is what they do and where they are."

Line 120 of the program adds 1024 to the value now stored in Register D. This is because the text screen memory begins at 1024. When this value is added to the screen location value you set in the USR command, the result is the proper screen memory location. The new value of 'D' is then transferred to Register X for safe keeping.

A second ROM routine is used in Line 140. Here, the routine at $A1C1 is used to capture the keys you type. It stores your keystrokes in Register A. Routine $A1C1 does not produce a cursor when you type. I have used the label GKEY to mark the beginning of the routine that handles the characters you type. Here is how the routine works:

- Line 150 compares a keystroke stored in Register A to a value of zero. Zero in Register A means that no key is pressed and the routine loops back to check again.
- Line 160 compares the keystroke to a space (you pressed the space bar). If it is a space, Line 170 branches to a routine to display a space (character 96). The normal character 32 (generated when you press the space bar) causes a black block to appear on the screen when stored in text screen memory.
- Line 180 compares the keystroke to a backspace (character 8). If you press a backspace, Line 190 branches to a routine that decreases Register X by one and stores 96 in place of the previously typed character, erasing it.
- Line 200 compares Register A to the value generated by a carriage return. If it is a carriage return, 210 causes the program to end.
- If you press a key, but it was not a carriage return, space or backspace, Line 220 stores the character value you typed into text screen memory at the location pointed to by Register X and increases Register X by one.
- Line 230 then causes the program to always branch back to the GKEY routine to look for the next keystroke.

You can see that using ROM routines is easy. A jump-to-subroutine instruction (JSR) is all that is required. Each ROM routine ends with a RTS (Return From Subroutine) instruction that causes it to return control to your program, beginning execution at the line following the JSR.

Listing 2: SCRNTYPE

(A program to read keyboard input and display it on the screen. The location for the display is controlled by the USR function.)

```
                     ØØØ1Ø ************************
                     ØØØ2Ø *     SCREEN TYPE      *
                     ØØØ3Ø * BY R. BARTLY BETTS   *
                     ØØØ4Ø ************************
                     ØØØ5Ø *
                     ØØØ6Ø *
ØEØØ                 ØØ1ØØ        ORG    $EØØ
ØEØØ BD   B3ED       ØØ11Ø START  JSR    $B3ED
ØEØ3 C3   Ø4ØØ       ØØ12Ø        ADDD   #1Ø24
ØEØ6 1F   Ø1         ØØ13Ø        TFR    D,X
ØEØ8 BD   A1C1       ØØ14Ø GKEY   JSR    $A1C1
ØEØB 27   FB         ØØ15Ø        BEQ    GKEY
ØEØD 81   2Ø         ØØ16Ø        CMPA   #32
ØEØF 27   ØC         ØØ17Ø        BEQ    SPACE
ØE11 81   Ø8         ØØ18Ø        CMPA   #8
ØE13 27   ØE         ØØ19Ø        BEQ    BACK
ØE15 81   ØD         ØØ2ØØ        CMPA   #13
ØE17 27   12         ØØ21Ø        BEQ    DONE
ØE19 A7   8Ø         ØØ22Ø        STA    ,X+
ØE1B 2Ø   EB         ØØ23Ø        BRA    GKEY
ØE1D 86   6Ø         ØØ24Ø SPACE  LDA    #96
ØE1F A7   8Ø         ØØ25Ø        STA    ,X+
ØE21 2Ø   E5         ØØ26Ø        BRA    GKEY
ØE23 3Ø   1F         ØØ27Ø BACK   LEAX   -1,X
ØE25 86   6Ø         ØØ28Ø        LDA    #96
ØE27 A7   84         ØØ29Ø        STA    ,X
ØE29 2Ø   DD         ØØ3ØØ        BRA    GKEY
ØE2B 39              ØØ31Ø DONE   RTS
          ØØØØ       ØØ32Ø        END

ØØØØ TOTAL ERRORS
```

"When using the PSHS and PULS instructions, remember that you must keep track of the order in which registers are loaded onto the stack. The stack operates in a 'last in, first out' manner."

Joining Forces with BASIC

To use Listing 2 with BASIC, all you need are a few short lines that establish the beginning of the machine language routine and call it with a USR command that contains a valid argument. Listing 3 does this:

Listing 3: INTRFACE
(A BASIC program to interface the previous machine language program with BASIC.)

```
 1Ø CLS
 2Ø DEFUSRØ=&HEØØ
 3Ø A=USRØ(1Ø1)
 4Ø END
```

Line 20 establishes the beginning of your machine language program with the DEFUSR statement. Line 30 calls the machine language program, passing it an argument of 101. Thus, when the program is run, the characters you type begin to appear at screen location 101.

To use the programs, simply LOAD them using the proper loading commands, type RUN and ENTER (because the machine language program is initialized by the BASIC program, you do not use the EXEC command). The screen clears and you can begin typing. To return to BASIC, press ENTER.

Type and Invert

Listing 4 uses some of the preceding features in a different way. The current cursor position establishes where the characters you type are displayed. The characters appear in reverse but, when you finish and press ENTER, they immediately invert to normal uppercase characters.

This program also uses two new ROM routines. The routine at $A1B1

Listing 4: REVTYPE
(A program to display keyboard input on the screen in reverse video, then invert the characters when you press ENTER.)

```
                        ØØØ1Ø **********************
                        ØØØ2Ø *     REVERSE TYPE    *
                        ØØØ3Ø *    BY  CHRIS BONE   *
                        ØØØ4Ø **********************
                        ØØØ5Ø *
                        ØØØ6Ø *
7ØØØ                    ØØ1ØØ         ORG     $7ØØØ
7ØØØ E6     84          ØØ11Ø START   LDB     ,X
7ØØ2 1ØAE   Ø2          ØØ12Ø         LDY     2,X
7ØØ5 34     2Ø          ØØ13Ø         PSHS    Y
7ØØ7 9E     88          ØØ14Ø         LDX     $88
7ØØ9 BF     7Ø5F        ØØ15Ø         STX     BEGIN
7ØØC 3A                 ØØ16Ø         ABX
7ØØD BF     7Ø5D        ØØ17Ø         STX     LENGTH
7Ø1Ø BD     A1B1        ØØ18Ø LOOP    JSR     $A1B1
7Ø13 81     ØD          ØØ19Ø         CMPA    #$ØD
7Ø15 27     2D          ØØ2ØØ         BEQ     DONE
7Ø17 81     2Ø          ØØ21Ø         CMPA    #32
7Ø19 25     ØC          ØØ22Ø         BLO     CONTRL
7Ø1B 26     Ø8          ØØ23Ø         BNE     FLIP
7Ø1D 9E     88          ØØ24Ø         LDX     $88
7Ø1F A7     8Ø          ØØ25Ø         STA     ,X+
7Ø21 9F     88          ØØ26Ø         STX     $88
7Ø23 2Ø     Ø7          ØØ27Ø         BRA     CONTRØ
7Ø25 88     2Ø          ØØ28Ø FLIP    EORA    #$2Ø
7Ø27 BD     A3ØA        ØØ29Ø CONTRL  JSR     $A3ØA
7Ø2A 9E     88          ØØ3ØØ         LDX     $88
7Ø2C BC     7Ø5F        ØØ31Ø CONTRØ  CMPX    BEGIN
7Ø2F 24     Ø7          ØØ32Ø         BHS     NOTBEF
7Ø31 BE     7Ø5F        ØØ33Ø         LDX     BEGIN
7Ø34 9F     88          ØØ34Ø         STX     $88
7Ø36 2Ø     D8          ØØ35Ø         BRA     LOOP
7Ø38 BC     7Ø5D        ØØ36Ø NOTBEF  CMPX    LENGTH
7Ø3B 25     D3          ØØ37Ø         BLO     LOOP
7Ø3D 86     Ø8          ØØ38Ø         LDA     #$8
7Ø3F BD     A3ØA        ØØ39Ø         JSR     $A3ØA
7Ø42 2Ø     CC          ØØ4ØØ         BRA     LOOP
7Ø44 BE     7Ø5F        ØØ41Ø DONE    LDX     BEGIN
7Ø47 35     2Ø          ØØ42Ø         PULS    Y
7Ø49 A6     84          ØØ43Ø DONELP  LDA     ,X
7Ø4B 88     4Ø          ØØ44Ø         EORA    #$4Ø
7Ø4D A7     8Ø          ØØ45Ø         STA     ,X+
7Ø4F 81     6Ø          ØØ46Ø         CMPA    #$6Ø
7Ø51 26     Ø2          ØØ47Ø         BNE     NOTSPC
7Ø53 86     2Ø          ØØ48Ø         LDA     #$2Ø
7Ø55 A7     AØ          ØØ49Ø NOTSPC  STA     ,Y+
7Ø57 BC     7Ø5D        ØØ5ØØ         CMPX    LENGTH
7Ø5A 26     ED          ØØ51Ø         BNE     DONELP
7Ø5C 39                 ØØ52Ø         RTS
7Ø5D                    ØØ53Ø LENGTH  RMB     2
7Ø5F                    ØØ54Ø BEGIN   RMB     2
            ØØØØ        ØØ55Ø         END
ØØØØØ TOTAL ERRORS
```

is similar to the routine at $A1C1, except the normal cursor is on the screen as you type. The second routine at $A304 causes the characters typed to be displayed on the screen. This is the routine your computer uses in BASIC, and it replaces the STA instruction we used in the earlier program.

Listing 5 is commented to help you follow its logic. The following notes describe some of the key points:

- The RMB instructions in lines 530 and 540 set aside storage space that points to the beginning and end of the string.
- Memory location $88 stores the current screen cursor position.
- The ABX instruction adds the 'X' and 'B' registers. 'X' and 'B' are the only registers that allow this 16- and 8-bit addition.
- The subroutine at $A1B1 gets whatever key you press and stores it in Register A. This subroutine provides a normal cursor while waiting for a key press.
- The subroutine at $A30A displays the character currently in Register A on the screen at the current cursor position.
- If you are not familiar with the ASCII values of characters (used in lines 190, 210, 310, 360, 380, 460 and 500), there is an ASCII chart in your Color BASIC manuals.

Listing 5: CALLER
(BASIC program to call the *Reverse Type* program. The number of spaces between the quotes in Line 120 establish the maximum number of characters you can type.)

```
1Ø DEFUSRØ=&H7ØØØ
2Ø A$=USRØ("          ")
3Ø END
```

Listing 6 is a sample of how the program might be handled from BASIC.

The program asks you to complete three sentences. The characters you type are in reverse video. When you complete your entry and press the ENTER key, the typed characters flip back to normal video. To show that the characters you type are stored in B$, Line 130 combines them with the partial sentence.

After using the program, LIST it. The last characters you typed are now part of the program and are displayed between the quotation marks in Line 120.

Listing: 6

A BASIC program to show how REVTYPE can be used.

```
10 CLS
20 CLEAR 1000,&H6FFF: REM change
 &H6FFF to &h3FFF for a 16K mach
ine
25 LOADM"REVTYPE": REM change LO
ADM to "CLOAD for a tape system
30 DEFUSR0=&H7000: REM change &h
7000 to &H3000 for a 16K machine
40 PRINT TAB(5) "COMPLETE THE SE
NTENCE
50 FOR T=1 TO 3
60 READ A$
70 S$(T)=A$
80 NEXT T
90 FOR T=1 TO 3
100 PRINT
110 PRINT S$(T)" ";
120 B$=USR0("
 ")
130 C$=S$(T)+" "+B$
140 PRINT
150 PRINT "this is what you wrot
e:"
160 PRINT C$
170 NEXT T
180 DATA ROSES ARE,MY BONNIE LIE
S,HOME IS WHERE THE
```

Forging Ahead

Now, put your imagination to work. Use some of the techniques you have learned to write programs of your own. You can, for instance, combine last month's program with the one for this month and create a program that lets you type characters in reverse video, flip them when you are through, then flash them.

You probably realize that you now have the knowledge to write a simple word processor, or create your own editing routine for a program.

You should feel that assembly language isn't too difficult. You should have the heady feeling of accomplishment that comes with realizing you are writing routines which could be used in full-fledged machine language games and utilities.

We have covered a lot of ground; now take your time and explore the new territory during the next month. It's a territory filled with more treasures. When you find some, write me, and we will share them with others. My address is 2251 Lipscomb, Fort Worth, TX 76110.

SOUND SYNTHESIS

64K
Disk

For The Analog-To-Digital And Back Again

By Stephen Gunn

After seeing Jeremy Spiller's "Analog-to-Digital And Back Again" article (June 1985, Page 36), I just had to try it! I typed it in and it worked; the only problem was that it would only give about 13 seconds of sound. The program only used 32K, so I modified it for 64K.

The assembly language listing is rather short, so it has been modified. Also included are four routines for moving blocks of memory from the upper page of RAM into the lower page and back again, therefore the BASIC SAVEM and LOADM commands can be used.

How the New Program Works

The BASIC program no longer pokes the routine into memory, but loads it from disk (or cassette). This makes changing the assembly language routine easier. The BASIC program then sets a flag in RAM, so even if you press BREAK and then RUN the program again, it won't bother reloading the machine language routine. The BASIC program also turns off the interrupts which allows for better sound quality.

You are presented with a menu similar to the old version's. Selection one is the same as in the original program. Selection two is different in that you can now also control the speed at which the tape is read in. Selecting a speed of one will give a higher sampling rate, hence better sound but less playing time (about 15 seconds). Selecting 15 as the speed will give much lower sound quality, but the playback time is 75 seconds, five times longer than the original program.

(One note: If you want something to sound the same when it plays back, select the same speed of playback as when it is read in. For example, if you read something in at the speed of 11, to have it sound the same it must play back at the speed of 11.)

Selection three is playback, same as in the original version. Selection four lets you save what is in memory to disk, but be warned that a disk will hold only two complete dumps from memory. The filename has to be seven characters or less because the program saves memory out in three files, adding the numbers '1', '2' and '3' to the filename given. After you save memory out, what is in memory is more or less destroyed by the saving process, so to listen to the same file use Selection five to reload it. (Selection five loads files back into memory.)

The modifications were created for a disk system, but I see no reason why they shouldn't work for cassette if you change all of the LOADMs to CLOADMs and all of the SAVEMs to CSAVEMs.

Loading Instructions

First, using an editor/assembler, type in and assemble the ML program. If you have a disk system and are using the cartridge *EDTASM*, assemble it to tape, then reconnect the disk controller and transfer it to disk. If you are using disk *EDTASM*, just assemble it to disk. Make sure you use the filename *SOUND.*

If you have the old BASIC version of Jeremy's program, you can just add my routines and changes to his program, otherwise you will have to type in the whole program (it isn't too long).

I enjoyed modifying *A/D DRVR* for 64K, and I hope you will enjoy using it! If you should have any questions regarding this new version, you may contact me at 944 Wisconsin, Oak Park, IL 60304. Please include an SASE. □

```
170 .......17
340 ......151
END .....102
```

Listing 1: 64KDRVR

```
0 'WHEN YOU SAVE A FILE,
   WHATEVER IS IN MEMORY IS
   RUINED BY THE SAVING PROCESS,
   SO THAT IF YOU WANT TO LISTEN
   TO THE FILE YOU JUST SAVED,
   YOU WILL HAVE TO RELOAD IT.
10 CLS:GOTO460
20 CLEAR100,&H1AFF:POKE65283,52:
IFPEEK(512)=0THENLOADM"SOUND":PO
KE512,85
30 POKE65344,0
40 CLS
50 PRINT"  ANALOG TO DIGITAL CON
VERTER"
60 PRINT"        BY JEREMY SPILLER
"
70 PRINT"        MODIFIED FOR 64K
"
80 PRINT"        BY STEPHEN GUNN"
90 PRINT
100 PRINT"1. TEST VOLUME OF TAPE
 RECORDER"
110 PRINT"2. READ TAPE (DIGITALI
ZE)"
120 PRINT"3. SAY DATA (PLAYBACK)
"
```

Listing 2: SOUND

```
                    ØØ1ØØ ****************************
                    ØØ11Ø *  SIMPLE A/D CONVERTER *
                    ØØ12Ø *   BY JEREMY SPILLER    *
                    ØØ13Ø *          1985          *
                    ØØ14Ø ****************************
1BØØ                ØØ15Ø         ORG     $1BØØ
1BØØ 7F   FFDF      ØØ16Ø START   CLR     $FFDF
1BØ3 8E   1CØØ      ØØ17Ø         LDX     #$1CØØ
1BØ6 7F   FF2Ø      ØØ18Ø         CLR     $FF2Ø
1BØ9 C6   Ø8        ØØ19Ø STO1Ø   LDB     #$8
1BØB 1Ø8E ØØØ6      ØØ2ØØ TIME    LDY     #6
1BØF 31   3F        ØØ21Ø TIME2   LEAY    -1,Y
1B11 26   FC        ØØ22Ø         BNE     TIME2
1B13 74   FF2Ø      ØØ23Ø         LSR     $FF2Ø
1B16 49             ØØ24Ø         ROLA
1B17 5A             ØØ25Ø         DECB
1B18 26   F1        ØØ26Ø         BNE     TIME
1B1A A7   8Ø        ØØ27Ø         STA     ,X+
1B1C 8C   FFØØ      ØØ28Ø         CMPX    #$FFØØ
1B1F 25   E8        ØØ29Ø         BLO     STO1Ø
1B21 7F   FFDE      ØØ3ØØ         CLR     $FFDE
1B24 39             ØØ31Ø         RTS
                    ØØ32Ø ****************************
                    ØØ33Ø *  RECONSTRUCTS SOUNDS   *
                    ØØ34Ø ****************************
1B25 7F   FFDF      ØØ35Ø PLAY    CLR     $FFDF
1B28 8D   26        ØØ36Ø         BSR     SOUND
1B2A 8E   1CØØ      ØØ37Ø         LDX     #$1CØØ
1B2D A6   8Ø        ØØ38Ø SOUND1  LDA     ,X+
1B2F C6   Ø8        ØØ39Ø         LDB     #8
1B31 48             ØØ4ØØ SOUND2  LSLA
1B32 76   FF2Ø      ØØ41Ø         ROR     $FF2Ø
1B35 1Ø8E ØØØ6      ØØ42Ø TIMER   LDY     #6
1B39 31   3F        ØØ43Ø TIMER2  LEAY    -1,Y
1B3B 26   FC        ØØ44Ø         BNE     TIMER2
1B3D 5A             ØØ45Ø         DECB
1B3E 26   F1        ØØ46Ø         BNE     SOUND2
1B4Ø 8C   FFØØ      ØØ47Ø         CMPX    #$FFØØ
1B43 25   E8        ØØ48Ø         BLO     SOUND1
1B45 7F   FFDE      ØØ49Ø         CLR     $FFDE
1B48 39             ØØ5ØØ         RTS
                    ØØ51Ø ****************************
```

```
130 PRINT"4. SAVE DATA"
140 PRINT"5. LOAD DATA"
150 PRINT
160 INPUT" PICK A NUMBER";A
170 ON A GOTO 190,240,290,330,39
0
180 GOTO 40
190 CLS
200 PRINT"TO GET BACK TO MENU PR
ESS RESET THEN RUN THIS PROGRAM.
"
210 PRINT
220 PRINT"SET YOUR TAPE RECORDER
 TO THE   CORRECT VOLUME"
230 MOTORON:EXEC &H1B49
240 CLS:INPUT"TYPE A NUMBER 1-15
 TO CONTROL   THE SPEED.  ENTER=
DEFAULT OF 6. =)";S:IFS=0 THEN S
=6
250 IF S)15THEN240
260 POKE&H1B0B+3,S
270 PRINT"READING TAPE":MOTORON:
EXEC &H1B00:MOTOROFF
280 GOTO40
290 CLS:INPUT"TYPE A NUMBER 1-15
 TO CONTROL   THE SPEED.  ENTER=
DEFAULT OF 6. =)";S:IF S=0 THEN
S=6
300 IF S)15 THEN290
310 CLS:PRINT"SAYING":POKE&H1B35
+3,S:EXEC &H1B25
320 GOTO40
330 CLS:LINEINPUT"ENTER FILE NAM
E: ";A$:IFLEN(A$))7THEN330
340 PRINT"SAVING"
350 SAVEM A$+"1",&H1C00,&H7FFF,0
360 EXEC &H1B69:SAVEM A$+"2",&H1
C00,&H7FFF,0
370 EXEC &H1B80:SAVEM A$+"3",&H1
C00,&H3700,0
380 GOTO30
390 CLS:LINEINPUT"FILE TO LOAD:
";A$
400 PRINT"LOADING"
410 LOADM A$+"3":EXEC&H1B97
420 LOADM A$+"2":EXEC&H1BAF
430 LOADM A$+"1"
440 GOTO30
450 '*PCLEAR 0*
460 POKE&H3C0,&H5F:POKE&H3C1,&H5
C
470 POKE&H3C2,&H96:POKE&H3C3,&HB
C
480 POKE&H3C4,&H1F:POKE&H3C5,&H0
2
490 POKE&H3C6,&H7E:POKE&H3C7,&H9
6:POKE&H3C8,&HA3
500 EXEC &H3C0
510 GOTO20
```

```
                        ØØ52Ø *  VOLUME CHECK ROUTINE *
                        ØØ53Ø *************************
1B49 8D      Ø5         ØØ54Ø CHECK    BSR      SOUND
1B4B 76      FF2Ø       ØØ55Ø VOLUME   ROR      $FF2Ø
1B4E 2Ø      FB         ØØ56Ø          BRA      VOLUME
1B5Ø B6      FFØ1       ØØ57Ø SOUND    LDA      $FFØ1
1B53 84      F7         ØØ58Ø          ANDA     #$F7
1B55 B7      FFØ1       ØØ59Ø          STA      $FFØ1
1B58 B6      FFØ3       ØØ6ØØ          LDA      $FFØ3
1B5B 84      F7         ØØ61Ø          ANDA     #$F7
1B5D B7      FFØ3       ØØ62Ø          STA      $FFØ3
1B6Ø B6      FF23       ØØ63Ø          LDA      $FF23
1B63 8A      Ø8         ØØ64Ø          ORA      #8
1B65 B7      FF23       ØØ65Ø          STA      $FF23
1B68 39                 ØØ66Ø          RTS
                        ØØ67Ø **************************
                        ØØ68Ø * LOAD/SAVE ADDED BY     *
                        ØØ69Ø * STEPHEN GUNN FOR 64K   *
                        ØØ7ØØ **************************
1B69 7F      FFDF       ØØ71Ø SAVE1    CLR      $FFDF
1B6C 8E      8ØØØ       ØØ72Ø          LDX      #$8ØØØ
1B6F 1Ø8E    1CØØ       ØØ73Ø          LDY      #$1CØØ
1B73 A6      8Ø         ØØ74Ø LOOP1    LDA      ,X+
1B75 A7      AØ         ØØ75Ø          STA      ,Y+
1B77 8C      E3FF       ØØ76Ø          CMPX     #$E3FF
1B7A 23      F7         ØØ77Ø          BLS      LOOP1
1B7C 7F      FFDE       ØØ78Ø          CLR      $FFDE
1B7F 39                 ØØ79Ø          RTS
1B8Ø 7F      FFDF       ØØ8ØØ SAVE2    CLR      $FFDF
1B83 8E      E4ØØ       ØØ81Ø          LDX      #$E4ØØ
1B86 1Ø8E    1CØØ       ØØ82Ø          LDY      #$1CØØ
1B8A A6      8Ø         ØØ83Ø LOOP2    LDA      ,X+
1B8C A7      AØ         ØØ84Ø          STA      ,Y+
1B8E 8C      FEFF       ØØ85Ø          CMPX     #$FEFF
1B91 23      F7         ØØ86Ø          BLS      LOOP2
1B93 7F      FFDE       ØØ87Ø          CLR      $FFDE
1B96 39                 ØØ88Ø          RTS
1B97 7F      FFDF       ØØ89Ø LOAD1    CLR      $FFDF
                        ØØ9ØØ **CORRESPONDS TO SAVE 2**
1B9A 8E      1CØØ       ØØ91Ø          LDX      #$1CØØ
1B9D 1Ø8E    E4ØØ       ØØ92Ø          LDY      #$E4ØØ
1BA1 A6      8Ø         ØØ93Ø LOOP3    LDA      ,X+
1BA3 A7      AØ         ØØ94Ø          STA      ,Y+
1BA5 1Ø8C    FEFF       ØØ95Ø          CMPY     #$FEFF
1BA9 23      F6         ØØ96Ø          BLS      LOOP3
1BAB 7F      FFDE       ØØ97Ø          CLR      $FFDE
1BAE 39                 ØØ98Ø          RTS
1BAF 7F      FFDF       ØØ99Ø LOAD2    CLR      $FFDF
                        Ø1ØØØ **CORRESPONDS TO SAVE 1**
                        Ø1Ø1Ø **($8ØØØ - $E3FF)**
1BB2 8E      1CØØ       Ø1Ø2Ø          LDX      #$1CØØ
1BB5 1Ø8E    8ØØØ       Ø1Ø3Ø          LDY      #$8ØØØ
1BB9 A6      8Ø         Ø1Ø4Ø LOOP4    LDA      ,X+
1BBB A7      AØ         Ø1Ø5Ø          STA      ,Y+
1BBD 1Ø8C    E3FF       Ø1Ø6Ø          CMPY     #$E3FF
1BC1 23      F6         Ø1Ø7Ø          BLS      LOOP4
1BC3 7F      FFDE       Ø1Ø8Ø          CLR      $FFDE
1BC6 39                 Ø1Ø9Ø END      RTS
             ØØØØ       Ø11ØØ          END
ØØØØØ TOTAL ERRORS
```

ASSEMBLY FILE

BY KEVIN

Last month we took a look at the general outline of the CoCo memory map. This month lets have a closer look at the actual contents of certain blocks of memory, primarily BASIC's system use RAM. We'll start with a list of the more important locations within this block of RAM.

0019 001A	Start of Basic Program Pointer	
001B 001C	End of Basic Program; Begin Variables Pointer	
001D 001E	Top of Variables; Start Arrays Pointer	
001F 0020	End of Arrays; Start of Free Memory Pointer	
0021 0022	Start of String Space Pointer	
0023 0024	Next usable String Space Pointer	
0025 0026	Pointer BASIC Memory Limit	
0027 0028	Start of CLEARed memory pointer	
0037 0038	Current Variable Name	
0068 0069	Current Program Line	
006E	I/O Type Flag	
006F	I/O Device Flag	00= Screen
		FF= Cassette
		FE= Printer
		01-15=Disk File
0071 0072	Power on Reset Setup Flag	
0074 0075	Pointer to End of RAM Memory	
007E 007F	Program End Address after CLOAD	
0088 0089	Pointer to Current Cursor Position	
0095 0096	Printer Baud Rate Code (norm =57)	
0116 0117	Seed for RND Function	
011A	Shift Lock Flag	
0120-013C	Token Table Directory	
	Byte 1 = No. of Keywords	
	Byte 2,3 = Address of Table	
	Byte 4,5 = Address of Subroutines	
0120-0124	BASIC Commands	
0125-0129	BASIC Functions	
012A-012E	Extended BASIC Commands	
012F-0133	Extended BASIC Functions	
0134-0138	Disk BASIC Commands	
0139-013C	Disk BASIC Functions	
013E-0151	Address for USR Functions 1 thru 9	
0152	Keyboard Strobe	8 0 X P H @
0153	Keyboard Strobe	9 1 Y Q I A
0154	Keyboard Strobe	* 2 Z R J B
0155	Keyboard Strobe	+ 3 ^ S K C
0156	Keyboard Strobe	< 4 [T L D
0157	Keyboard Strobe	- 5 U M E
0158	Keyboard Strobe	> 6] V N F
0159	Keyboard Strobe	/ 7 W O G
015A	Right Joystick Horiz Value	
015B	Right Joystick Vert Value	
015C	Left Joystick Horiz Value	
015D	Left Joystick Vert Value	
015E-0160	Open Device Hook	
0161-0163	Device Number Check	
0164-0166	Return Device Hook	
0167-0169	Character Output Hook	
016A-016C	Input Character Hook	
016D-016F	Check File Open Hook	
017F-0181	BREAK Key & SHIFT@ Hook	
0182-0183	Get Line from Keyboard Hook	
0191-0193	Error Routine Hook	
01A0-01A2	CLS Hook	
01A5-01A7	List Hook	
01D2-01D9	Cassette File Name	
01DA-02D8	Cassette Buffer	
01DA-01E1	CLOADM Filename	
01E5 01E6	EXEC Address From Tape	
01E7 01E8	ORG address from Tape	
02DD 03FF	Keyboard Line Input Buffer	
0400-05FF	Text Screen RAM	
A000 A001	Address of Keyboard Scan Routine	
A002 A003	Address of Character Output Routine	
A027	Primary Reset	
FF00-FF03	PIA 1	
FF20-FF23	PIA 2	
FFD4	Clear P1	0-7FFF RAM = Page 0
FFD5	Set P1	0-7FFF RAM = Page 1
FFD6	Clear R0	Reset CPU Rate
FFD7	Set R0	Set CPU Rate (Normal speedup POKE)
FFD8	Clear R1	Reset CPU Rate
FFD9	Set R1	Set CPU Rate (Extra Fast Speedup)
FFDE	Clear TY	Map Type 0. CPU Addresses ROMs
FFDF	Set TY	Map Type 1. CPU Addresses 64K RAM

Naturally enough this is again only a small portion of the entire CoCo Memory Map. However it does comprise the main substance of the blocks of memory we will be working with in the near future plus a bit extra for good measure.

Play around with the memory locations referred to in this list. Write BASIC routines to PEEK these memory locations. EXEC the addresses formed by the second and third bytes of the 'Hook' values. Is the first byte of these 'Hook' addresses always $7E, the Machine Code instruction for JMP. Play around with $FFD8 and $FFD9. Try timing this sort of short program:

```
10 POKE &HFFD9,0
20 FORX=1TO1000:NEXT
30 POKE &HFFD8,0
```

Then change line 10 to read: 10 POKE &HFFD8,0.
What do you think is happening?

Exercise your natural curiosity and next month we will make use of some of these gems of information.

16K & COCOCONNECTION

THE COCOCONNECTION

by John Poxon & Graham Morphett

Last month we looked at a simple program to work with the CoCoConnection, which would act as a quiz master.

One of the disadvantages of the way it was set up, was its need to have the input buttons physically numbered.

If the computer can take inputs from external sources such as these buttons, it follows there should be no need to number them.

So we have made some changes to the program and come up with the following program, which now allows the holder of the button to input his/her name without the need to know the number of the button.

The major changes occured in the lines 300 - 395, where the fairly standard input provided originally, was replaced with a routine which:

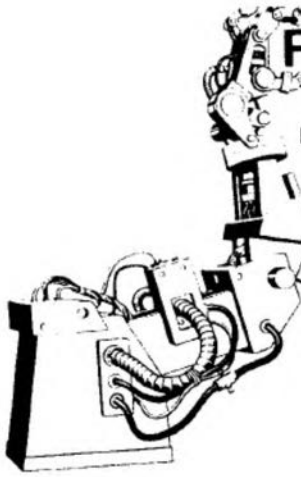
- checks for a button push (line 350);
- checks which button it was (GOSUB500);
- keeps a record of the first button pushed (line 360);
- if the first button is pushed again, then proceed to next part of program (line 363);
- if not, then INPUT a name (line 365);
- do it all again (line 370).

We also moved the subroutine at line 500 around a bit.

Originally this routine only had to service the question / answer section of the program, but now it had to be made available to the Names Input section as well. Not a difficult job, and we saved some K in the process!

THE LISTING:

```
1 '*COCOCONNECTION DEMONSTRATION
   ***************GAME*************
   *JOHN POXON & GRAHAM MORPHETT
   *******   26/8/85   **********
2 GOTO10
3 SAVE"CONDEMO2":DIR:STOP
10 CLS4:POKE65495,0
20 CLEAR2000:DIMZ$(8),ZZ(8)
40 PRINT@104,"*****************";
50 PRINT@136,"***SWITCHGAME***";
60 PRINT@168,"***    BY    ***";
70 PRINT@200,"***JOHN POXON***";
80 PRINT@232,"*****************";
90 FORT=0TO70:PLAY"O1;L255;ABCDE
FG":NEXTT
100 '***BEGIN***
130 CLS
135 PRINT@64," THIS IS A SMALL D
EMONSTRATION":PRINT"   OF THE AB
ILITIES OF THE":PRINTTAB(8)"COCO
CONNECTION.":PRINTTAB(9)"IF IT I
S NOT":PRINT"     INSERTED INTO T
HE COLOUR           COMPUTER, SWITC
H OFF THE           COMPUTER BEFORE
 INSERTING.":GOSUB185
136 PRINT"ENSURE ALL WIRES ARE F
IRMLY     CONNECTED AND ARE NOT
LIKELY TO PULL THE COCOCONNECTIO
N FROM THECOMPUTER."
139 GOSUB185
140 GOSUB200
150 GOSUB300
160 GOTO400
180 END
185 PRINT@448,"PRESS <ENTER> TO
CONTINUE";:PLAY"O1;L255;ABCDEFGA
BCDEFG":I$=INKEY$:IFI$=""THEN185
ELSE CLS:RETURN
190 '
200 '***INITIALISATION***
220 POKE 65409,0:POKE 65408,0:PO
KE 65409,60
230 POKE 65415,0:POKE 65414,0:PO
KE 65415,60
240 RETURN
290 '
300 '***STORE STUDENT NAMES***
310 'IT WOULD BE HANDY TO BE ABL
E TO LOAD IN NAMES FROM A STUDEN
T FILE HERE TO NEGATE THE NEED T
O INPUT THE SAME NAMES EACH TIME
!
330 CLS:PRINT@69,"STUDENT NAME I
NPUT."
340 PRINT"DECIDE WHO IS GOING TO
 PUT THEIR NAME IN 1ST, 2ND & SO
 ON, THEN PUSH YOUR BUTTONS IN T
HAT ORDER TO INPUT NAMES"
345 PRINT"    WHEN ALL NAMES ARE
 ENTERED,  THE FIRST PERSON SHOU
LD PRESS  THEIR BUTTON AGAIN TO
START THE GAME."
350 A=PEEK(65408):B=255-A:IFB=0T
HEN350
355 GOSUB500
360 NO=NO+1:IFNO=1THENC=X:GOTO36
5
362 IFX=C THEN FOR T=1TO300:NEXT
T:RETURN
363 FORT=1TO8:IFC(T)=X THEN 350
ELSE NEXTT
364 FORT=1TO8:IF C(T)=0 THEN C(T
)=X ELSE NEXTT
365 LINEINPUT"PLEASE TYPE IN YOU
R NAME            ";Z$(X)
370 GOTO350
395 RETURN
399 '
400 '***ASK QUESTIONS***
410 READQ$,AN$
420 IFQ$="Q"THEN700
430 CLS:PRINTQ$
450 A=PEEK(65408)
460 B=255-A
465 PLAY"O1;L255;ABCDEFG"
470 IF B=0 THEN 450
480 GOSUB500:GOSUB650:GOTO410
490 GOTO450
495 '
500 '**FIND FIRST BUTTON PUSHED*
520 IFB=32THENX=6
530 IFB=64THENX=7
540 IFB=128THENX=8
550 IFB=1THENX=1
560 IFB=2THENX=2
570 IFB=4THENX=3
580 IFB=8THENX=4
590 IFB=16THENX=5
595 RETURN
599 '
640 '
650 '*PRINT NAME & ASK QUESTION*
660 CLS:PRINT@96,Z$(X);" YOU WER
E FIRST":PRINTQ$:LINEINPUTI$
670 IFI$=AN$THEN PRINT"WELL DONE
 ";Z$(X):FORT=1TO2500:NEXT:ZZ(X)
=ZZ(X)+1:CLS:GOTO695
690 PRINT"I'M SORRY "Z$(X)" EVER
YONE TRY AGAIN!":FORT=1TO2500:NE
XT:CLS:GOTO430
695 IFHS<ZZ(X)THEN HS=ZZ(X)
696 GOTO410
699 '
700 '***FINAL SCORES***
710 CLS:PRINTTAB(9)"YOUR RESULTS
"
715 PRINT"   NAME","SCORE"
720 FORX=1TO8
730 PRINTZ$(X),ZZ(X);:IF ZZ(X)=H
S THEN PRINT"*" ELSE PRINT
740 NEXTX
745 PRINT:PRINTTAB(5)"'*' - WINN
ER!"
750 GOSUB185
760 PRINT"THIS UNIT ALLOWS THE C
OCO TO":PRINT"CONTROL MOST ELECT
RICAL OR MECH-ANICAL ITEMS. THIN
```



ROBOTICS

by Graham Morphett

What is a Robot?

When we think of Robots, we tend to conjure up images of Hollywood's monsters, remorselessly plodding toward the doomed heroine. (Of course just in time she invariably is rescued by the invariably handsome star - why can't heros be old and ugly like me?)

The reality is very different, and ever so slowly, our society is beginning to understand just what a Robot really is.

For a start we know that Robot arms are being used in many factories around the world.

The Automotive Industry has led the world in this type of technology - and they themselves have been led by Italy (Fiat) and Japan (Various).

Robot arms were first used in the Nuclear Industry, where they are indispensible.

Today there are a very wide range of uses for them, but they are still only one type of Robot.

Most of today's Robots are not perceived as such. Have you ever thought as you got into the lift at work, that you were walking into the hand of a Robot?

Well you are - this type of Robot is built to lift you from one floor of a building to another.

Other Robots stop you from crossing a railway level crossing when a train is approaching. And your dot matrix printer is really a very specialized Robot.

So what is a Robot?

In many ways, the term is still being defined. But the term really refers to mechanical devices which are given the ability to make choices.

The case of the classic Robot is obvious - he chooses to chase the heroine, and then proceeds to do so.

Interfacing the CoCoConnection to a Robot is quite simple. In fact it is only one step on from the Quiz Master program we discussed above.

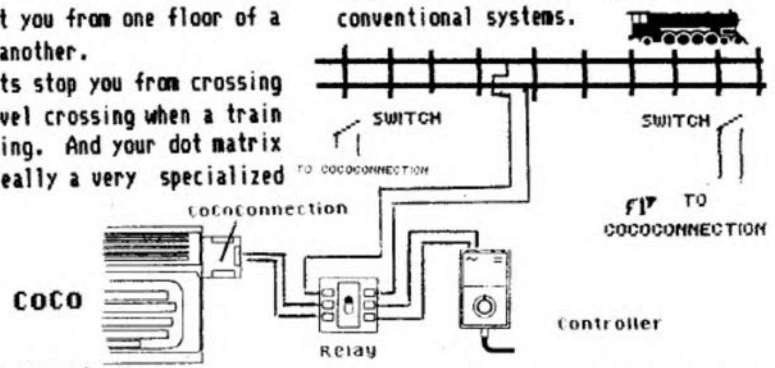
A simple Robot might be represented by a model train on a model train track.

Without the computer, the train requires a voltage of up to 12 volts DC to make it go. Positive current is carried in one rail, negative in the other. The train picks up the current through its wheels which are insulated from each other.

To make it go, we need a length of track, the train has to have all its wheels on the track, we need a power supply, which needs to be connected to the track, and we need to apply the power from the power supply to the train.

This sounds simple, and is.

If we introduce a CoCo to this environment, we complicate the system, but we are able to automate the operation of the train with an ease hitherto not possible in conventional systems.



In the diagram we have a straight piece of track, a train, and a controller. If we turn on the controller, the train will go; but if we don't turn it off in time, it will run off the end of the track.

In real life, buffers are a valuable aid in such circumstances.

But, we have CoCo!

We need to place a normally open switch at each end of the track, connected to the appropriate inputs of the CoCoConnection. If the switch is depressed, CoCo will detect it and we can tell CoCo to stop the train, reverse the polarity, and send the train back the other way.

When the train reaches the other end of the track, the same proceedure can apply.

Next month we will look at the hardware needed to put this simple project into action, and to add a new feature, a siding.

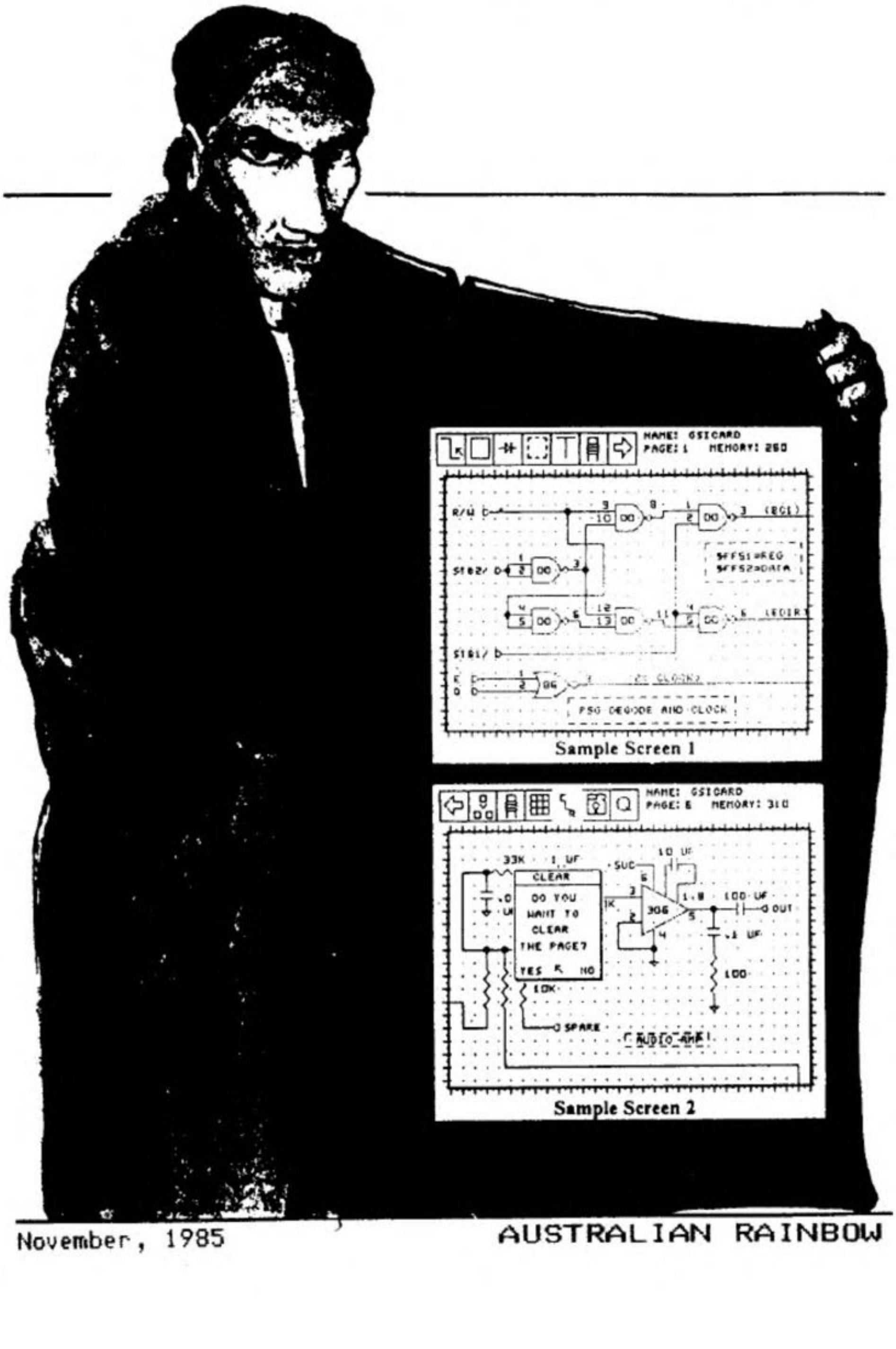
The important point to remember is that most physical tasks can be accomplished using the CoCoConnection and a CoCo, but as with programming, the steps needed to achieve the task need to be fully explored.

```
GS LIKE ROBOTSMODEL TRAINS BURGU
LAR ALARMS ANDSCIENCE EXPERIMENT
S ARE SUITABLESUBJECTS FOR THE"
765 PRINTTAB(8)"COCOCONNECTION."
:GOSUB185
770 PRINT"THANK YOU FOR PARTICIP
ATING                     BYE!":PO
KE65494,0:FORT=1TO2000:NEXTT:RUN
799 '
800 '***DATA FOR QUESTIONS***
810 DATA"WHAT IS THE CAPITAL OF
                    AUSTRALIA?","CANB
ERRA"
820 DATAWHAT IS THE OPTIMUM NUMB
ER OF    STUDENTS PER CLASS,0
830 DATA"WHICH TOWN IN QUEENSLAN
D HAS THE BRIGHTEST STUDENTS?",S
OUTHPORT
840 DATA"HOW DO YOU SPELL 'BEER'
?","XXXX"
850 DATA"AT WHICH SHOP IN WARWIC
K CAN YOUBUY AUSTRALIAN RAINBOW
MAGAZINE?","TANDY"
860 DATA WHAT DUST HAS TRADITION
ALLY BEENASSOCIATED WITH TEACHER
S?,CHALK
870 DATA WHY TEACH?,MONEY
880 DATA WHAT IS THE NICK NAME O
F THE    TANDY COLOUR COMPUTER?,
COCO
890 DATAWHAT IS 5 X 6?,30
990 DATA"Q","Q",
```

CoCocad:

The Schematic Scoundrel

By Peter Kerckhoff



Sample Screen 1

Sample Screen 2

A "mini-cad" (computer-aided design) program for the Color Computer, CoCoad aids you in drawing schematics. It requires a 32K Color Computer, one disk drive, a printer with bit mapped graphics ability and a positioning device (Radio Shack color mouse, joystick, X-pad or touch tablet). CoCoad is icon menu driven, and the keyboard is seldom used. Some functions make use of a "pop-up" menu for their options.

Instead of digging into the program and describing how I got each function to work, I have provided a highly commented listing (COCOAD.REM); as they say, "leave the proof to the student." A word of caution: The commented listing is rather long and you'll have to PCLEAR 1 just to load it in. You'll also have to remove all of the remark statements and resave the program (use the name COCOAD.BAS) before you try to run it. If you are keying in the program, just leave the remarks out and don't do the PCLEAR.

I've also included a short program listing, REMOVREM, to remove the remarks for you. Just remember to reload and resave the new CoCoad program as the remark remover program generates an ASCII file (program) that takes a while to load.

Once you've got an uncommented copy of the program, load it into the computer and run it. The screen should clear and a header line followed by a question dealing with clearing memory should appear at the top of the screen. The memory clear question is so you can restart the program with a schematic in memory without losing the schematic. Since you do not have a schematic in memory, you should answer 'Y'. After a very short delay the screen will again clear and the work sheet screen will appear.

The work sheet screen is where almost all of the action takes place. At the upper left-hand corner of the screen is a series of boxes with small figures (icons) in them. I've provided a couple of screen dumps. Sample Screens 1 and 2, that show the work sheet screens. These boxes are used to call the various functions into use (more about this in a moment). In the upper right-hand portion of the display is a display of the current filename, the current page number and the memory remaining in the page.

A page is a sub-page of the whole schematic. The schematic is made up of nine sub-pages arranged in an array three across and three down. Page zero is the upper left-hand corner of the finished schematic. Page one is just below page zero and page two just below page one. Page eight is the lower right-hand corner.

The memory for a page is used for storing each function you use to draw the schematic. If you start running out of memory space, the computer will warn you by beeping.

The main portion of the screen is taken up by the schematic work area. The sides of the work area have grid mark on them to ease the drawing of lines between pages. The work area itself is stippled with dots every eight pixels. When you are in the schematic work area, your cursor will "snap" to either the dots or to the area halfway between the dots. This makes the positioning of line end points, components and text easier.

Somewhere on the screen there should appear an arrow. The arrow should be under positional control by your mouse (X-pad, joystick, touch pad , etc). The arrow cursor appears when the computer wants you to select an item from the menu (or submenu, as we'll see in a minute). To select an item, position the arrow on the item and click the mouse button (press down on the button, then release).

Let's select our first menu item. Move the mouse so the arrow is in the menu box which contains the icon that looks like the letter 'Z'. Now click the mouse button. The sides of the box should disappear, letting you know the computer has accepted your command.

To activate the menu item, move the mouse down into the schematic work area. A new cursor will appear in the work area. This cursor should be shaped like a box whose corners have been flipped inwards. This is the end point (or point) cursor. The point cursor is used to identify an end point or a spot in the work area to the computer.

To draw the line, position the point cursor to the spot in the work area where you want one end of the line to be and click the mouse button. The cursor should flicker and be replaced by a new cursor shaped like a box. This cursor identifies the first end point for your line. If you move the mouse you'll see that you still have control over the point cursor. Move the cursor to the second end point and click the mouse button. A couple of things will happen: The cursors will disappear, a line will be drawn and the memory remaining display will be updated. You still have control of the point cursor, so you can draw another line by identifying end points and so on.

To exit the line function (or any other), move the cursor up into the main menu. The point cursor will disappear and be replaced with the menu-selecting arrow cursor. Whenever you have control of the action on the screen with the mouse you can exit by going up into the main menu. This can come in handy, for example, if you've selected the wrong first end point for a line and you'd like to start over.

OK, a quick review: To select a function from the menu, move the arrow cursor to the function and click, then move down into the schematic work area;to exit a function move the cursor back into the main menu. Easy isn't it?

The next menu selection is the box function. We'll include in our discussion of the box function the fourth menu item (dashed box) and the fifth menu item (erase), as these functions are very similar.

Select the box function and move the mouse so the point cursor is in the schematic work area. Move the cursor to a point where you want one corner of the box to appear and click the mouse button. The cursor will vanish and, if you look closely, there will be a single pixel blinking on and off at the point where the cursor was. Now move the mouse; a box will follow the mouse's movement. When you are satisfied with the box, click the mouse button again so you'll be all set to draw another box.

The dashed box and erase functions work much in the same way. The differences are that with the dashed box, when the box has been defined, a dash box will be drawn. With the erase, the area in the box defined will be erased.

The third menu item, the diode, is used to select one of the preprogrammed components from the component library. Get the arrow cursor back and group and select the component function. A pop-up menu will appear on the screen in the schematic work area. At the top of the menu is the menu title, in this case "components." In the center of the menu is the currently active component. Down at the bottom of the menu on the left are the words "PREV" and "NEXT." Placing the arrow cursor on either of these words and clicking the mouse button will change the active component displayed in the center of the menu. Holding down the mouse button while the arrow cursor is over one of these words will cycle you through the components.

At the right of the menu on the bottom is the word "USE." Clicking the mouse button while the cursor is over this word will clear the menu from the screen and give you the active component to use. The component will be under mouse control and move about in the schematic work area as you move the mouse. When you have placed the component to the desired position, click the mouse button and the component will be drawn onto the screen. If you move the mouse again, you'll notice that you still have control over the component so that another one can be drawn onto the screen. To select another component or another menu item, just move up into the main menu.

There are two more menu items, the bit 'T' and the right pointing arrow. The 'T' function is used to put text on the screen. Get the arrow cursor back and select the text function. When you move the mouse back down into the work area you will notice there is a new cursor, a small horizontal line; this is the text cursor. Move the text cursor where you want the text to appear and click the mouse button. The cursor will move upwards two pixels and no longer be under mouse control. The cursor moves up so that if you are labelling a line or an end point of a line, the text will line up properly.

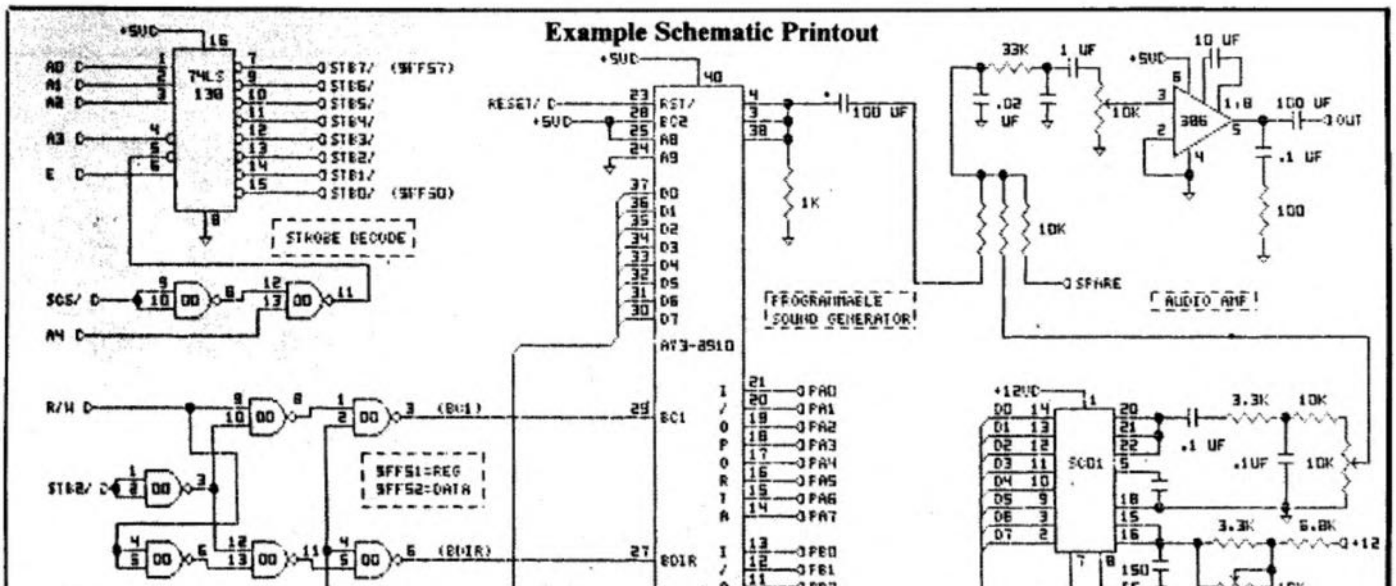
Enter text by typing on the keyboard. The backspace (left arrow) key has been implemented so if there is a typing error, use it to correct the error. When you finish with the test entry, press the ENTER key and the page memory will be updated with your text. The text cursor will be under mouse control again so you can move the cursor to the next spot and enter more text.

The final menu item is the right pointing arrow. This function will switch the current menu to the second menu. Try out the function by placing the arrow cursor in the box and clicking the mouse button (don't forget to move the mouse downwards to execute the function). The old menu will disappear and be replaced with a new one.

The first item in this new menu is a left pointing arrow, and you guessed it, will get you back to the original menu. The third menu item is another copy of the erase function and is identical to the one on the original menu. The remaining menu items are new.

The second menu item, the one with the three little squares, is the copy function, Select the function and move down into the work area. Move the point cursor to the area near the object you wish to copy and click the mouse button. Move the mouse so the box that is under mouse control surrounds the object you want to copy and click the mouse button again. The box will disappear and if you move the mouse you'll notice that a copy of the

Example Schematic Printout



item you boxed will move. When the copy click has been positioned, the mouse button and the copy will be left on the screen and you'll get the point cursor back. Note the size of the boxed area will directly affect how fast the image will move -the larger the box the slower the move.

The fourth menu item, the one with the nine little squares, is the page switch function. When you select the function, a blinking cursor will appear as the current page number (upper right-hand portion of the display). Press a key between zero and eight to select the page. Remember, page zero is in the upper left and page eight is in the lower right of the final schematic. When a page number is selected, the current page will be cleared and the selected page displayed.

The fifth menu item, the box with the "CLR" in it, is the clear page function. Selecting the function will produce a pop-up menu asking if you really want to clear the page. Answer by positioning the cursor over your answer and clicking the mouse button.

The sixth menu item, the box with the little floppy disk, is for disk I/O and will display a pop-up menu when the function is selected. You must enter a file name before you can load or save. Once again, select the commands from the pop-up menu by positioning the cursor over the command and clicking the mouse button.

The last menu item, the large 'Q', is used to quote the program with the option of getting a printout of the schematic in memory. if you do decide to get a printout the computer will check to see if there is enough room on the disk for the temporary storage of the sub-page image files and, if there is, will load and run the schematic printout program (Cadprint) after saving the image files.

The best and easiest way to get familiar with CoCoad is to use the program. You'll find the mouse really does work well with the program and that selecting and using the various functions follow a logical order. Before going in and modifying the program, make sure you look it over. There are quite a few subroutines you may want to use.

For X-pad users, you will have to change the mouse scan routine to read the X-pad. You will also notice the pen status is stored in the variable 'P' and can be read directly from the X-pad (guess what positioning device I started writing the program with!).

In closing, if you use CoCoad to draw a schematic you hope to have published, please put a little note somewhere on it denoting CoCoad was used to draw the schematic - my ego will get a kick out of it!

43	172	436	57	831	213
90	134	460	167	874	89
110	252	494	143	960	81
141	42	530	117	1100	49
170	237	562	206	1280	153
211	225	610	125	1580	254
241	115	641	4	1730	217
280	38	690	17	1780	95
314	240	720	223	1860	50
345	69	747	72	1966	97
400	42	773	200	END	23

Listing 1: COCOCAD

This is a commented listing of the program CoCocad. To load the program you must first PCLEAR 1. To run the program you must remove all of the comment lines. Don't forget to PCLEAR 4 before running.

```
0 '
1 '      COCOCAD    VR 2.0
2 ' (C) 1985 BY P.KERCKHOFF
3 ' 4335 HENDRIX WAY SAN JOSE CA
4 '
5 ' A cad SYSTEM FOR THE COLOR
6 ' COMPUTER. REQUIRES 32K, DISK
7 ' X-PAD AND A GRAPHICS PRINTER
8 '
9 '
10 CLEAR 500,&H64FF:CLS:INPUT"CO
COCAD VR 2.0  P.KERCKHOFF     CL
EAR PAGE MEMORY (Y/N)";A$:GOTO12
0
20 X=JOYSTK(0)*4:Y=JOYSTK(1)*4:P
=ABS((1 AND PEEK(&HFF00))*3-3):I
F X<3 THEN X=3 ELSE IF X>252 THE
N X=252
30 IF Y<3 THEN Y=3 ELSE IF Y>188
 THEN Y=188
40 RETURN
50 GET(0,0)-(6,6),C1,G:LINE(0,0)
-(6,6),PRESET,BF:DRAW"BM3,3XC$(N
);":GET(0,0)-(6,6),C2,G:PUT(0,0)
-(6,6),C1,PSET:RETURN
60 GET(X-3,Y-3)-(X+3,Y+3),C1,G:P
UT(X-3,Y-3)-(X+3,Y+3),C2,AND:RET
URN
70 PUT(X-3,Y-3)-(X+3,Y+3),C1,PSE
T:RETURN
80 IF ASC(A$)>62 THEN 90 ELSE ON
 ASC(A$)-31 GOTO 1120,1130,1140,
1150,1160,1170,1180,1190,1200,12
10,1220,1230,1240,1250,1260,1270
,1280,1290,1300,1310,1320,1330,1
340,1350,1360,1370,1380,1390,140
0,1410,1420
90 ON ASC(A$)-62 GOTO 1430,1440,
1450,1460,1470,1480,1490,1500,15
10,1520,1530,1540,1550,1560,1570
,1580,1590,1600,1610,1620,1630,1
640,1650,1660,1670,1680,1690,170
0
100 FOR C=1 TO LEN(T$):DRAW"BM=T
X;,=TY;":A$=MID$(T$,C,1):GOSUB 8
0:TX=TX+5:NEXT C:RETURN
110 LINE(221,9)-(236,13),PRESET,
BF:T$=STR$(768-AD(PG)+A(PG)):T$=
RIGHT$(T$,LEN(T$)-1):TX=221:TY=1
3:GOSUB100:IFLEN(T$)=1THENPLAY"E
E03EE02EE01EE04EE":RETURN ELSE R
ETURN
120 DIM C$(3),A(8),AD(8),C1(1),C
2(1),C3(1),L1(6),L2(6),L3(6),L4(
6),CM(15),CO(15),MD(255),MO(255)
:B$="V31L10004B":NF$="NONE"
130 C$(0)="BHNL2U2BR2D2R2BD2L2D2
BL2U2L2":C$(1)="BH3ND2R2BR2R2D2B
D2D2L2BL2L2U2":C$(2)="R3":C$(3)=
"NF2H2ND3R3":FOR X=0 TO 8:A(X)=X
*768+&H6500:IF A$="Y" THEN POKEA
(X),0:NEXT ELSE NEXT
140 PG=0:GOSUB1710:MN=0:N=3:GOSU
B50
150 GOSUB20:GOSUB60:IF P<>3 OR Y
>20 THEN GOSUB 70:GOTO 150 ELSE
GOSUB 70: F=INT(X/20):IF F>6 THE
N 150 ELSE LINE(F*19+1,1)-(F*19+
20,20),PRESET,B:F=F+1:PLAYB$
160 GOSUB 20:IF Y<20 OR P=3 THEN
 160 ELSE IF MN=0 THEN ON F GOTO
 190,290,440,290,520,290,180 ELS
```

```
E ON F GOTO180,580,290,680,700,7
40,820
170 PLAYB$:F=F-1:LINE(F*19+1,1)-
(F*19+20,20),PSET,B:N=3:GOSUB50:
GOTO150
180 LINE(0,0)-(134,20),PRESET,BF
:FORX=0TO6:LINE(X*19+1,1)-(X*19+
20,20),PSET,B:NEXT:IF MN=1 THEN
MN=0:GOSUB 1760:GOTO170 ELSE MN=
1:GOSUB 1770:GOTO170
190 N=0:GOSUB50:LX=-1
200 GOSUB20:IFP=3THEN200
210 GOSUB20:GOSUB60:IFY>20THEN22
0ELSEGOSUB70:IFLX=-1THEN170ELSEP
UT(LX-3,LY-3)-(LX+3,LY+3),C3,PSE
T:GOTO170
220 GOSUB70:IFP<>3THENX1=X:Y1=Y:
GOTO210ELSEPLAYB$:X=X1:Y=Y1
230 IF LX=-1 THEN LX=X:LY=Y:GET(
LX-3,LY-3)-(LX+3,LY+3),C3,G:DRAW
"BM=LX;,=LY;XC$(1);":GOTO200
240 PUT(LX-3,LY-3)-(LX+3,LY+3),C
3,PSET:IF Y<27 THEN 170
250 X=INT(X/4)*4:Y=INT(Y/4)*4
260 LINE(X,Y)-(LX,LY),PSET:POKEA
D(PG),52:POKEAD(PG)+1,X:POKEAD(P
G)+2,Y:POKEAD(PG)+3,LX:POKEAD(PG
)+4,LY:AD(PG)=AD(PG)+5:POKEAD(PG
),0:GOSUB110:LX=-1:GOTO200
270 GET(PX,PY)-(LX,PY),L1,G:GET(
PX,PY)-(PX,LY),L2,G:GET(LX,LY)-(
PX,LY),L3,G:GET(LX,LY)-(LX,PY),L
4,G:LINE(PX,PY)-(LX,LY),PSET,B:R
ETURN
280 PUT(PX,PY)-(LX,PY),L1,PSET:P
UT(PX,PY)-(PX,LY),L2,PSET:PUT(LX
,LY)-(PX,LY),L3,PSET:PUT(LX,LY)-
(LX,PY),L4,PSET:RETURN
290 N=0:GOSUB50
300 GOSUB20:GOSUB60:IF Y<20 THEN
 GOSUB70:GOTO170 ELSE IF P<>3 TH
EN GOSUB70:GOTO300
310 GOSUB70:LX=X:LY=Y:PX=LX:PY=L
Y
320 GOSUB20:IFP=3THEN320
330 GOSUB270:GOSUB 20:IF Y<20 TH
EN GOSUB 280:GOTO 170 ELSE IF P<
>3 THEN GOSUB 280:PX=X:PY=Y:GOTO
 330
340 POKEAD(PG)+1,LX:POKEAD(PG)+2
,LY:POKEAD(PG)+3,PX:POKEAD(PG)+4
,PY:POKEAD(PG)+5,0:IF F=2 THEN P
OKE AD(PG),53:AD(PG)=AD(PG)+5:GO
SUB110:GOTO290
350 IF F<>4 THEN 410 ELSE LINE(L
X,LY)-(PX,PY),PRESET,B:POKEAD(PG
),55:
360 X=PX:Y=PY:IF X>LX THEN PX=LX
:LX=X:X=PX
370 IF Y>LY THEN PY=LY:LY=Y:Y=PY
```

```
380 AD(PG)=AD(PG)+5:GOSUB110:FOR
 X1=X TO LX STEP 8:Y1=X1+4:IF Y1
>LX THEN Y1=LX
390 LINE(X1,Y)-(Y1,Y),PSET:LINE(
X1,LY)-(Y1,LY),PSET:NEXT:FOR X1=
Y TO LY STEP8:Y1=X1+4:IF Y1>LY T
HEN Y1=LY
400 LINE(X,X1)-(X,Y1),PSET:LINE(
LX,X1)-(LX,Y1),PSET:NEXT:GOTO290
410 LINE(LX,LY)-(PX,PY),PRESET,B
F:IF PX>LX THEN X=PX:PX=LX:LX=X
420 IF PY>LY THEN Y=PY:PY=LY:LY=
Y
430 PX=INT(PX/8)*8:PY=INT(PY/8)*
8+4:FOR X=PX TO LX STEP 8:FOR Y=
PY TO LY STEP 8:PSET(X,Y):NEXT Y
,X:POKEAD(PG),54:AD(PG)=AD(PG)+5
:GOSUB110:GOTO290
440 T$="COMPONENTS":GOSUB850:T$=
"NEXT   USE":TX=55:TY=115:GOSUB1
00:T$="PREV":TX=55:TY=107:GOSUB1
00:N=3:GOSUB50:N=1:DRAW"BM65,65"
:GOSUB870
450 GOSUB20:GOSUB60:GOSUB70:IF Y
<27 THEN PLAYB$:PUT(50,50)-(110,
120),MO,PSET:GOTO170 ELSE IF Y<1
00 OR Y>118 OR X<55 OR X>110 OR
P<>3 THEN 450 ELSE PLAY B$
460 IF X>80 THEN GET(65,65)-(65+
XW,65+YW),CM,G:PUT(50,50)-(110,1
20),MO,PSET:GOTO480 ELSE DRAW"BM
65,65C5":GOSUB870:DRAW"BM65,65C0
":IF Y<109 THEN 470 ELSE N=N+1:I
F N<25 THEN GOSUB870:GOTO450 ELS
E N=1:GOSUB870:GOTO450
470 N=N-1:IF N>0 THEN GOSUB870:G
OTO450 ELSE N=24:GOSUB870:GOTO45
0
480 GOSUB20
490 GET(X,Y)-(X+XW,Y+YW),CO,G:PU
T(X,Y)-(X+XW,Y+YW),CM,AND:LX=X:L
Y=Y:GOSUB20:IF P<>3 AND Y>26 THE
N PUT(LX,LY)-(LX+XW,LY+YW),CO,PS
ET: GOTO 490
500 IF Y<27 THEN PUT(LX,LY)-(LX+
XW,LY+YW),CO,PSET:GOTO 170
510 PLAYB$:POKEAD(PG),N:POKEAD(P
G)+1,LX:POKEAD(PG)+2,LY:AD(PG)=A
D(PG)+3:POKEAD(PG),0:GOSUB110:GO
TO480
520 N=2:GOSUB50
530 GOSUB20:GOSUB60:GOSUB70:IF P
<>3 AND Y>26 THEN 530 ELSE Y=Y-2
540 IFY<27 THEN 170 ELSE PLAYB$:
POKEAD(PG),51:POKEAD(PG)+1,X:POK
EAD(PG)+2,Y:AD(PG)=AD(PG)+3:GOSU
B110:X1=AD(PG)
550 GOSUB60:A$=INKEY$:GOSUB70:IF
 A$="" THEN 550 ELSE PLAY B$:IF
A$=CHR$(13) THEN POKE AD(PG),255
```

```
:AD(PG)=AD(PG)+1:POKEAD(PG),0:GO
SUB110:GOTO520
560 IF A$=CHR$(8) THEN IF AD(PG)
>X1 THEN AD(PG)=AD(PG)-1:X=X-5:L
INE(X,Y)-(X+5,Y-6),PRESET,BF:GOT
O550 ELSE GOTO550
570 POKEAD(PG),ASC(A$):AD(PG)=AD
(PG)+1:DRAW"BM=X;,=Y;":GOSUB80:X
=X+5:GOSUB110:GOTO550
580 N=0:GOSUB50
590 GOSUB20:GOSUB60:IF P<>3 AND
Y>26 THENGOSUB70:GOTO590ELSEGOSU
B70:IF Y<27 THEN170
600 PLAYB$:LX=X:LY=Y:PX=LX:PY=LY
610 GOSUB20:IFP=3THEN610
620 GOSUB270:GOSUB20:IFY<26THENG
OSUB280:GOTO170ELSEIFP<>3THENGOS
UB280:PX=X:PY=Y:GOTO620
630 GOSUB280:PLAYB$:XW=ABS(PX-LX
):YW=ABS(PY-LY):IF (XW*YW)>10000
 THEN 580 ELSE GET(LX,LY)-(PX,PY
),MD,G:POKEAD(PG)+1,PX:POKEAD(PG
)+2,PY:POKEAD(PG)+3,LX:POKEAD(PG
)+4,LY
640 GOSUB20:GET(X,Y)-(X+XW,Y+YW)
,MO,G:PUT(X,Y)-(X+XW,Y+YW),MD,AN
D:PUT(X,Y)-(X+XW,Y+YW),MO,PSET:I
FY<27THEN170ELSEIFP<>3THENX1=X:Y
1=Y:GOTO640ELSEX=X1:Y=Y1:PUT(X,Y
)-(X+XW,Y+YW),MD,AND
650 POKEAD(PG),56:POKEAD(PG)+5,X
:POKEAD(PG)+6,Y:AD(PG)=AD(PG)+7:
POKEAD(PG),0:GOSUB110:GOTO580
680 LINE(166,13)-(171,8),PSET,BF
:A$=INKEY$:LINE(166,13)-(171,8),
PRESET,BF:IF A$<"0" OR A$>"8" TH
EN 680
690 PG=VAL(A$):DRAW"BM166,13":GO
SUB80:LINE(8,28)-(247,183),PRESE
T,BF:GOSUB1790:FORX=8TO248STEP8:
FORY=28TO184STEP8:PSET(X,Y):NEXT
Y,X:GOSUB110:GOTO170
700 T$="  CLEAR":GOSUB850:T$="YE
S     NO":TX=55:TY=115:GOSUB100:
N=3:GOSUB50:TX=63:TY=70:T$="DO Y
OU":GOSUB100:TX=60:TY=80:T$="WAN
T TO":GOSUB100:TX=63:TY=90:T$="C
LEAR":GOSUB100:TX=55:TY=100:T$="
THE PAGE?":GOSUB100
710 GOSUB20:GOSUB60:GOSUB70:IFY<
27 THEN PLAYB$:PUT(50,50)-(110,1
20),MO,PSET:GOTO170 ELSE IF Y<10
0 OR Y>120 OR X<55 OR X>110 OR P
<>3 THEN 710 ELSE PLAY B$
720 IF X>80 THEN PUT(50,50)-(110
,120),MO,PSET:GOTO170
730 POKEA(PG),0:AD(PG)=A(PG):GOS
UB110:LINE(8,28)-(247,183),PRESE
T,BF:FORX=8TO248STEP8:FORY=28TO1
84STEP8:PSET(X,Y):NEXTY,X:GOTO17
```

```
0
740 T$="   DISK":GOSUB850:T$="NA
ME":TX=70:TY=70:GOSUB100:T$="LOA
D":TX=70:TY=80:GOSUB100:T$="SAVE
":TX=70:TY=90:GOSUB100:N=3:GOSUB
50
750 GOSUB20:GOSUB60:GOSUB70:IF Y
<27 THEN PLAYB$:PUT(50,50)-(110,
120),M0,PSET:GOTO170 ELSE IF Y<6
0 OR Y>90 OR X<70 OR X>90 OR P<>
3 THEN 750 ELSE PLAY B$
760 IF Y>70 THEN 770 ELSE CLS:LI
NEINPUT"FILE NAME =>";NF$:SCREEN
1,1:LINE(170,5)-(210,0),PRESET,B
F:TY=5:TX=170:T$=NF$:GOSUB100:GO
TO750
770 IF NF$="" OR NF$="NONE" THEN
 750 ELSE IF Y>80 THEN 780 ELSE
TX=60:TY=110:T$="LOADING":GOSUB1
00:LOADM NF$:A$="0":GOTO 690
780 IF Y>90 THEN 750 ELSE TX=60:
TY=110:T$="SAVING":GOSUB100:SAVE
M NF$,&H6500,&H7FFF,&H6500:LINE(
60,110)-(90,105),PRESET,BF:GOTO7
50
820 T$="   QUIT":GOSUB850:TX=60:
TY=70:T$="SAVE TO":GOSUB100:TX=7
0:TY=80:T$="DISK":GOSUB100:T$="Y
ES     NO":TX=55:TY=115:GOSUB100
:N=3:GOSUB50
830 GOSUB20:GOSUB60:GOSUB70:IF Y
<27 THEN PLAYB$:PUT(50,50)-(110,
120),M0,PSET:GOTO170 ELSE IF Y<1
00 OR Y>120 OR X<50 OR X>110 OR
P<>3 THEN 830 ELSE PLAY B$
840 IF X<80 THEN PUT(50,50)-(110
,120),M0,PSET:GOTO740 ELSE CLS:I
NPUT"DO YOU WANT A PRINTUUT (Y/N
)";A$:IF A$="Y" THEN 1970 ELSE P
RINT "OK...BYE.":STOP
850 GET(50,50)-(110,120),M0,G:LI
NE(50,50)-(110,120),PRESET,BF:LI
NE(51,51)-(108,118),PSET,B:LINE(
52,119)-(109,119),PSET:LINE-(109
,52),PSET:LINE(51,60)-(108,60),P
SET:TX=54:TY=57:GOSUB100:RETURN
870 ON N GOTO 880,890,900,910,92
0,930,940,950,960,970,980,990,10
00,1010,1020,1030,1040,1050,1060
,1070,1080,1090,1100,1110
880 DRAW"BD15U15R13FRF4DFD2GDG4L
GL13":XW=23:YW=18:RETURN:'AND
890 DRAW"BD15U15R12FRF4DFD RE2RF
2G2LH2L DGDG4LGL12":XW=28:YW=18:
RETURN:'NAND
900 DRAW"BD16RE2UEU2EU2 HU2HUH2L
R13FR2FRF2RF4 G4LG2LGL2GL12":XW=
25:YW=16:RETURN:'OR
910 DRAW"BD16RE2UEU2EU2HU2HUH2LR
13FR2FRF2RF4 E2R2F2G2L2H2 G4LG2L
```

```
GL2GL13":XW=31:YW=17:RETURN:'NOR
920 DRAW"BD16U16FRFRFRFRFRFRFRFG
LGLGLGLGLGLGLG":XW=15:YW=17:RETU
RN:'BUFFER
930 DRAW"BD15U15FRFRFRFRFRFRFRF
RER2FDGL2HU GLGLGLGLGLGLGLGL":XW
=20:YW=17:RETURN:'INVTR
940 DRAW"BD17RE2UEU2EU3HU2HUH2LB
R4R8FR2FRF2RF4DG4LG2LGL2GL7RE2UE
U2EU3HU2HUH2L":XW=24:YW=17:RETUR
N:'XOR
950 DRAW"BD16BR2RE2UEU2EU2HU2HUH
2LBR4R8FR2FRF2RF4 RE2RF2G2LH2L G
4LG2LGL2GL8RE2UEU2EU2HU2HUH2L":X
W=32:YW=17:RETURN:'XNOR
960 DRAW"3D21U19FRFRFRFRFRFRFRFR
FRFGLGLGLGLGLGLGLGLGLGU6 BR3R2LD
U2 BU7LR2":XW=19:YW=22:RETURN:'O
PAMP
970 DRAW"BD8R6NU5ND5E6U2D2G6F5U2
G2R2FD2":XW=12:YW=16:RETURN:'NPN
980 DRAW"BD8R6NU5ND5E6U2D2G2U2F2
L2G4F6D2":XW=12:YW=16:RETURN:'PN
P
990 DRAW"BD4R4E2F4E4F4E4F4E2R4":
XW=32:YW=8:RETURN:'H-REST
1000 DRAW"BR4D4F2G4F4G4F4G4F2D4"
:XW=8:YW=32:RETURN:'V-REST
1010 DRAW"BD4R4NU4ND4BR4NU4ND4R4
":XW=12:YW=8:RETURN:'H-CAP
1020 DRAW"BR4D4NL4NR4BD4NL4NR4D4
":XW=8:YW=12:RETURN:'V-CAP
1030 DRAW "BD4BR8 L4H2L2R2F2G2L2
U4":XW=7:YW=10:RETURN:'INPUT
1040 DRAW"BD4R4E2R2D4L2H2":XW=10
:YW=6:RETURN:'OUTPUT
1050 DRAW"BD3ER2FD2GL2HU2":XW=8:
YW=8:RETURN:'NOT DOT
1060 DRAW "BD3BR2END2RND2RND2FDN
L2DNL2GLNU2LNU2HUR2UR2":XW=8:YW=
8:RETURN:'CON DOT
1070 DRAW"BR4D4L3F3E3L3":XW=8:YW
=8:RETURN:'GND
1080 DRAW"BD4R5ND4U4F4NG4RNU4ND4
R5":XW=15:YW=8:RETURN:'>DIODE
1090 DRAW"BD4R5ND4NU4RNF4E4D4ND4
R5":XW=15:YW=8:RETURN:'<DIODE
1100 DRAW"BD8R2E2F4E4U6D5NH2NE2D
F4E4F4E2R2":XW=28:YW=10:RETURN:'
H-POT
1110 DRAW"BR4D2F2G4F4R6L5NE2NF2L
G4F4G4F2D2":XW=14:YW=28:RETURN:'
V-POT
1120 DRAW"BR4":RETURN
1130 DRAW"RBU2U2LD2":RETURN
1140 DRAW"BU2U2BR2D2":RETURN
1150 DRAW"BRU4BR2D4URL4RU2LR4":R
ETURN
1160 DRAW"R3U2L3U2R3LD4":RETURN
1170 DRAW"E4BL3DBR3BD2D":RETURN
```

```
1180 DRAW"BR2NU4H2R3":RETURN
1190 DRAW"BRBU4D2":RETURN
1200 DRAW"BR2HU2E":RETURN
1210 DRAW"BREU2H":RETURN
1220 DRAW"BUE2GHF2":RETURN
1230 DRAW"BU2R2LUD2":RETURN
1240 DRAW"BRU2":RETURN
1250 DRAW"BU2R3":RETURN
1260 DRAW"BRURD":RETURN
1270 DRAW"UE2U":RETURN
1280 DRAW"U4R3D4L3":RETURN
1290 DRAW"R2LU4G":RETURN
1300 DRAW"NR3U2R3U2L3":RETURN
1310 DRAW"R3U2NL2U2L3":RETURN
1320 DRAW"BU4D2R3U2D4":RETURN
1330 DRAW"R3U2L3U2R3":RETURN
1340 DRAW"NU2R3U2L3U2R3":RETURN
1350 DRAW"BU4R3DGD2":RETURN
1360 DRAW"NU4R3U2NL3U2L3":RETURN
1370 DRAW"R3U2L3U2R3D2":RETURN
1380 DRAW"RULBU2RUL":RETURN
1390 DRAW"BU4RDLBD2RD":RETURN
1400 DRAW"BU2NE2F2":RETURN
1410 DRAW"BUR3BU2L3":RETURN
1420 DRAW"E2H2":RETURN
1430 DRAW"BU3UR3D2LBDD":RETURN
1440 DRAW"NR3U3ER2BD2LDRU":RETUR
N
1450 DRAW"U3ERFDNL3D2":RETURN
1460 DRAW"U4R2FGNL2FGL2":RETURN
1470 DRAW"BRREGLHU2ERF":RETURN
1480 DRAW"U4R2FD2GL2":RETURN
1490 DRAW"NR3U2NR2U2R3":RETURN
1500 DRAW"U2NR2U2R3":RETURN
1510 DRAW"BU4BR3L2GD2FREUL":RETU
RN
1520 DRAW"U4D2R3U2D4":RETURN
1530 DRAW"R2LU4NLR":RETURN
1540 DRAW"BUFREU3":RETURN
1550 DRAW"U2RNF2NE2LU2":RETURN
1560 DRAW"NR3U4":RETURN
1570 DRAW"U4FRED4":RETURN
1580 DRAW"U4F3DU4":RETURN
1590 DRAW"BUU2ERFD2GLH":RETURN
1600 DRAW"U4R2FGL2":RETURN
1610 DRAW"BUU2ERFD2GNUNRLH":RETU
RN
1620 DRAW"U4R2FGL2RF2":RETURN
1630 DRAW"R2EHLHER2":RETURN
1640 DRAW"BU4R2LD4":RETURN
1650 DRAW"NU4R3U4":RETURN
1660 DRAW"BU4D3FREU3":RETURN
1670 DRAW"NU4ERFU4":RETURN
1680 DRAW"UE2UDGHUDF2D":RETURN
1690 DRAW"BU4DFEUDGD2":RETURN
1700 DRAW"NR2UE2UL2":RETURN
1710 PMODE4,1:SCREEN1,1:COLOR0,1
:PCLS:DRAW"C0S4":TX=140:TY=5:T$=
"NAME: "+NF$:GOSUB100:TX=140:TY=
13:T$="PAGE:":GOSUB100:TX=184:TY
```

```
=13:T$="MEMORY:":GOSUB100
1720 FORX=0TO6:LINE(X*19+1,1)-(X
*19+20,20),PSET,B:NEXT:LINE(7,27
)-(248,184),PSET,B:LINE(6,26)-(2
49,185),PSET,B:GOSUB1760
1730 FORX=8TO248STEP8:LINE(X,27)
-(X,23),PSET:LINE(X+4,27)-(X+4,2
5),PSET:LINE(X,184)-(X,188),PSET
:LINE(X+4,184)-(X+4,186),PSET:NE
XT:LINE(252,23)-(252,191),PRESET
1740 FORY=28TO184STEP8:LINE(7,Y)
-(3,Y),PSET:LINE(7,Y+4)-(5,Y+4),
PSET:LINE(248,Y)-(252,Y),PSET:LI
NE(248,Y+4)-(250,Y+4),PSET:NEXT
1750 GOSUB1790:FORX=8TO248STEP8:
FORY=28TO184STEP8:PSET(X,Y):NEXT
Y,X:GOSUB110:A$=RIGHT$(STR$(PG),
1):DRAW"BM166,13":GOSUB80:RETURN
1760 DRAW"BM4,4R6D12R6 BM24,4R12
D12L12U12 BM43,10R4NU3D3E3NH3RNU
3ND3R3 BM62,4R2BR3R2BR3R2D2BD3D2
BD3D2L2BL3L2BL3L2U2BU3U2BU3U2 BM
81,4R12L6D12 BM102,4D4NR6GFNR6GF
NR6GFND4R6ND4EHEHEHU4L6 BM119,8N
D4R6U4F6G6U4L6":RETURN
1770 DRAW"BM4,10E6D4R6D4L6D4H6 B
M100,4R12D3L2DR2D8L6U3E2H2LG2F2D
3L5U12D3R6U3 BM28,8U4R4D4L4BD2F2
E2BD4R4D4L4U4BL4L4D4R4U4 BM46,4D
4NR6GFNR6GFNR6GFND4R6ND4EHEHEHU4
L6 BM62,4R4ND12R4ND12R4D4NL12D4N
L12D4L12U12 BM82,4NR3D4R3BDD4R3B
DND4R3D2L2NF2L"
1780 DRAW"BM122,6R4F2D4GNHFRLHGL
4H2U4E2":RETURN
1790 AD(PG)=A(PG)
1800 D=PEEK(AD(PG)):IF D=0 THEN
RETURN
1810 X=PEEK(AD(PG)+1):Y=PEEK(AD(
PG)+2):AD(PG)=AD(PG)+3:IF D>50 T
HEN 1830
1820 DRAW"BM=X;,=Y;":N=D:GOSUB87
0:GOTO1800
1830 IF D<>51 THEN 1850
1840 D=PEEK(AD(PG)): AD(PG)=AD(P
G)+1:IF D=255 THEN 1800 ELSE A$=
CHR$(D):DRAW"BM=X;,=Y;":GOSUB80:
X=X+5:GOTO 1840
1850 LX=PEEK(AD(PG)):LY=PEEK(AD(
PG)+1):AD(PG)=AD(PG)+2
1860 IF D=52 THEN LINE(X,Y)-(LX,
LY),PSET:GOTO 1800
1870 IF D=53 THEN LINE(X,Y)-(LX,
LY),PSET,B:GOTO1800
1880 IF D=54 THEN LINE(X,Y)-(LX,
LY),PRESET,BF:GOTO1800
1890 IF D<>55 THEN 1950
1900 IF LX<X THEN X1=LX:LX=X:X=X
1
1910 IF LY<Y THEN Y1=LY:LY=Y:Y=Y
1
1920 FOR X1=X TO LX STEP 8:Y1=X1
+4:IF Y1>LX THEN Y1=LX
1930 LINE(X1,Y)-(Y1,Y),PSET:LINE
(X1,LY)-(Y1,LY),PSET:NEXT:FOR X1
=Y TO LY STEP8:Y1=X1+4:IF Y1>LY
THEN Y1=LY
1940 LINE(X,X1)-(X,Y1),PSET:LINE
(LX,X1)-(LX,Y1),PSET:NEXT:GOTO18
00
1950 IF D<>56 THEN AD(PG)=AD(PG)
-5:POKE AD(PG),0:RETURN
1960 GET(X,Y)-(LX,LY),MD,G:LX=AB
S(X-LX):LY=ABS(Y-LY):X=PEEK(AD(P
G)):Y=PEEK(AD(PG)+1):PUT(X,Y)-(X
+LX,Y+LY),MD,AND:AD(PG)=AD(PG)+2
:GOTO1800
1970 PRINT:PRINT"CHECKING FREE D
ISK SPACE":X=FREE(0):IF X<28 THE
N PRINT "NOT ENOUGH ROOM ON THE
DISK.":STOP ELSE PRINT "CREATING
 SUB PAGE IMAGE FILES.":FOR X=1
TO 1000:NEXT
1980 FOR PG=0 TO 8:FL$="PRT.PG"+
RIGHT$(STR$(PG),1):PMODE4,1:SCRE
EN1,1:COLOR0,1:PCLS:GOSUB1790:SA
VEM FL$,&H0E00,&H25FF,&H0E00:NEX
TPG
1990 RUN "CADPRINT"
```

30 233
90 133
END 170

Listing 2: CADPRINT

This program is called by the program CoCocad when a schematic printout is preformed. The program was written for use with the Gemini-10 printer and will have to be modified if another printer is used.

```
0 '
1 ' CADPRINT   VR 1.0
2 ' (C) 1985 BY P.KERCKHOFF
3 ' 4335 HENDRIX WAY SAN JOSE CA
4 '
5 ' THE PRINTOUT PART OF THE
6 ' PROGRAM COCOCAD, USED WITH
7 ' THE GEMINI 10 OR 10X PRINTER
8 '
9 '
10 CLEAR100,&H379A:CLS:PRINT"cad
print RUNNING":CS=0:FOR X=&H379B
 TO &H37FF:READ A$:D=VAL("&H"+A$
):CS=CS+D:POKE X,D:NEXT:READ X:I
F CS<>X THEN PRINT "ERROR IN DAT
A STATEMENTS.":STOP
11 '
12 '
13 ' PICK YOUR BAUD RATE, 1=
14 ' 9600 BAUD
15 '
20 POKE150,1
21 '
22 '
23 ' SETUP LINEFEED LENGTH TO
24 ' 16/144THS OF AN INCH
25 '
30 PRINT#-2,CHR$(27);CHR$(51);CH
R$(16)
31 '
32 '
33 ' FOR EACH SUB PAGE OF THE
34 ' SCHEMATIC (LOOP)
35 '
40 FOR PG=0 TO 8 STEP 3
41 '
42 '
43 ' GET THE PAGES FROM DISK
44 ' AND STORE THEM IN MEMORY
45 ' (UPPER MEMORY)
46 '
50 LOADM "PRT.PG"+RIGHT$(STR$(PG
),1),&H2A00
60 LOADM "PRT.PG"+RIGHT$(STR$(PG
+1),1),&H4200
70 LOADM "PRT.PG"+RIGHT$(STR$(PG
+2),1),&H5A00
71 '
72 '
73 ' CALL THE MACHINE CODE
74 ' PRINTOUT ROUTINE POKED
75 ' IN EARLIER
76 '
80 EXEC &H379B
81 '
82 '
83 ' ERASE THE PAGES FROM THE
84 ' DISK AND LOOP UNTIL DONE
85 '
90 KILL "PRT.PG"+RIGHT$(STR$(PG)
,1):KILL "PRT.PG"+RIGHT$(STR$(PG
+1),1):KILL "PRT.PG"+RIGHT$(STR$
(PG+2),1):NEXT PG
100 PRINT:PRINT"DONE.":STOP
50000 DATA 86,FE,97,6F,86,1,34,2
,86,1B,AD,9F,A0,2,86,4B,AD,9F,A0
,2,86,D4,AD,9F,A0,2,86,1,AD,9F,A
0,2,35,2,8E,7E,E0,30,86,34,2,8D,
26,35,2,8E,66,E0,30,86,34,2,8D,1
B,35,2,8E,4E,E0,30,86,34,2,8D,10
,86,D,AD,9F,A0,2,35,2,4C,81,1F,2
6,B8,F,6F,39,C6,9C
50010 DATA A6,84,43,34,14,AD,9F,
A0,2,35,14,30,88,E0,5A,26,EF,39
50020 DATA 9974
```

Listing 3:

This program is a short machine language printer driver that is poked into memory and called by the *Cadprint* program.

```
                     ØØ1ØØ
379B                 ØØ11Ø        ORG $379B
379B  7E      379B   ØØ12Ø BEGIN  JMP      START
379E          ØØ65   ØØ13Ø        FDB      DONE-BEGIN
                     ØØ14Ø
                     ØØ15Ø
                     ØØ16Ø        TITLE    CADCODE
                     ØØ17Ø
                     ØØ18Ø * A ROUTINE TO QUICKEN
                     ØØ19Ø * UP THE PRINTOUT
                     ØØ2ØØ
379B                 ØØ21Ø        ORG $379B
                     ØØ22Ø
                     ØØ23Ø
              AAØ2   ØØ24Ø CHROUT EQU      $AØØ2  COCO'S OUTPUT ROUTINE
              ØØ6F   ØØ25Ø DEVNUM EQU      $ØØ6F  --2 FOR PRINTER
                     ØØ26Ø
                     ØØ27Ø
                     ØØ28Ø * PRINT THREE SUB PAGES
                     ØØ29Ø * OUT TO THE PRINTER
                     ØØ3ØØ * ASSUMES BAUD RATE HAS
                     ØØ31Ø * BEEN SET AND PRINTER
                     ØØ32Ø * IS ALL SET
                     ØØ33Ø
379B 86  FE          ØØ34Ø START  LDA      #$FE   SET OUTPUT TO
379D 97  6F          ØØ35Ø        STA      DEVNUM THE PRINTER
379F 86  Ø1          ØØ36Ø        LDA      #1     SET X POS
37A1 34  Ø2          ØØ37Ø MLOOP  PSHS     A      SAVE IT, MAIN LOOP
37A3 86  1B          ØØ38Ø        LDA      #27    SET PRINTER UP
37A5 AD  9F AØØ2     ØØ39Ø        JSR      [CHROUT]
37A9 86  4B          ØØ4ØØ        LDA      #75    ESC,K (GRAPHICS MODE)
37AB AD  9F AØØ2     ØØ41Ø        JSR      [CHROUT]
37AF 86  D4          ØØ42Ø        LDA      #212   468 BYTES AT A TIME
37B1 AD  9F AØØ2     ØØ43Ø        JSR      [CHROUT]
37B5 86  Ø1          ØØ44Ø        LDA      #1     468/256
37B7 AD  9F AØØ2     ØØ45Ø        JSR      [CHROUT]
37BB 35  Ø2          ØØ46Ø        PULS     A      GET X POS
37BD 8E  7EEØ        ØØ47Ø        LDX      #$7EEØ BOTTOM SUB PAGE -
37CØ 3Ø  86          ØØ48Ø        LEAX     A,X    CALC ADDRESS->X REG
37C2 34  Ø2          ØØ49Ø        PSHS     A      SAVE X POS
37C4 8D  26          ØØ5ØØ        BSR      PRINT  GO PRINT BYTES
37C6 35  Ø2          ØØ51Ø        PULS     A      GET X POS
37C8 8E  66EØ        ØØ52Ø        LDX      #$66EØ MIDDLE SUB PAGE
37CB 3Ø  86          ØØ53Ø        LEAX     A,X
37CD 34  Ø2          ØØ54Ø        PSHS     A
37CF 8D  1B          ØØ55Ø        BSR      PRINT
37D1 35  Ø2          ØØ56Ø        PULS     A
37D3 8E  4EEØ        ØØ57Ø        LDX      #$4EEØ TOP SUB PAGE
37D6 3Ø  86          ØØ58Ø        LEAX     A,X
37D8 34  Ø2          ØØ59Ø        PSHS     A
37DA 8D  1Ø          ØØ6ØØ        BSR      PRINT
37DC 86  ØD          ØØ61Ø        LDA      #13    A CR/LF
37DE AD  9F AØØ2     ØØ62Ø        JSR      [CHROUT]
37E2 35  Ø2          ØØ63Ø        PULS     A      GET X POS
37E4 4C              ØØ64Ø        INCA
37E5 81  1F          ØØ65Ø        CMPA     #31    DONE?
37E7 26  B8          ØØ66Ø        BNE      MLOOP  NOPE, LOOP
37E9 ØF  6B          ØØ67Ø        CLR      DEVNUM YEP, SET OUTPUT FOR SCREEN
37EB 39              ØØ68Ø        RTS             EXIT TO BASIC
                     ØØ69Ø
37EC C6  9C          ØØ7ØØ PRINT  LDB      #156   Y COUNTER
37EE A6  84          ØØ71Ø PLOOP  LDA      ,X     GET BYTE
37FØ 43              ØØ72Ø        COMA            INVERT BITS
37F1 34  14          ØØ73Ø        PSHS     B,X    SAVE REG'S
37F3 AD  9F AØØ2     ØØ74Ø        JSR      [CHROUT] PRINT BYTE
37F7 35  14          ØØ75Ø        PULS     B,X
37F9 3Ø  88 EØ       ØØ76Ø        LEAX     -32,X  NEXT Y ADDRESS
37FC 5A              ØØ77Ø        DECB            DONE ALL Y'S?
37FD 26  EF          ØØ78Ø        BNE      PLOOP  NOPE, LOOP
37FF 39              ØØ79Ø        RTS             YEP, RETURN
                     ØØ8ØØ
                     ØØ81Ø
         38ØØ        ØØ82Ø DONE   EQU      *
         ØØØØ        ØØ83Ø        END
```

Listing 4: REMOVREM

This program will remove lines not evenly divisible by 10 from a program on disk saved in the ASCII format. (The program is used to remove the comment lines from COCOCAD.REM.)

```
1 ' REMOVREM
2 ' A VERY SIMPLE REMARK
3 ' REMOVER PROGRAM. REMOVES
4 ' LINES NOT EVENLY DIVISABLE
5 ' BY TEN.
6 '
7 ' (C) 1985 P.KERCKHOFF
8 '
9 '
10 CLEAR 500
11 '
12 '
13 ' OPEN THE OLD PROGRAM AS
14 ' AN ASCII FILE
15 '
20 OPEN "I",#1,"COCOCAD.REM"
21 '
22 '
23 ' OPEN A FILE FOR THE NEW
24 ' PROGRAM TO GO INTO
25 '
30 OPEN "O",#2,"COCOCAD.BAS"
31 '
32 '
33 ' IF END OF THE FILE THEN
34 ' WE'RE ALL DONE
35 '
40 IF EOF(1) THEN 80
41 '
42 '
43 ' GET THE DATA FROM THE OLD
44 ' FILE (PROGRAM)
45 '
50 LINEINPUT #1,A$
51 '
52 '
53 ' IF THE LINE NUMBER IS NOT
54 ' EVENLY DIVIS. BY TEN THEN
55 ' DON'T SAVE IT IN THE NEW
56 ' FILE (PROGRAM)
57 '
60 IF VAL(A$)<>INT(VAL(A$)/10)*1
0 THEN PRINT "REMOVED LINE: ";VA
L(A$):GOTO 40
61 '
62 '
63 ' ELSE THE LINE IS GOOD SO
64 ' SAVE IT IN THE NEW FILE
65 '
70 PRINT #2,A$:GOTO 40
71 '
72 '
73 ' ALL DONE. CLOSE IT AND
74 ' STOP.
75 '
80 CLOSE
90 PRINT "NOW LOAD 'COCOCAD.BAS'
 AND       RESAVE (THE LOAD WILL
TAKE QUITEAWHILE)."
100 STOP
```

FORTH

FORTH FORUM

by John Poxon

During the last couple of months I have presented the program BOILER in the languages BASIC, PASCAL and FORTH. While discussing the FORTH BOILER last month I remarked that although fast, it could be speeded up by using look-up routines, and added that the program suffered from a cycling problem at high and low values of water in the drum. (This prevents D1 and D2 from becoming sufficiently close to escape BOILER).

This article examines the application of a look-up table, explains its construction, and answers the more-than-trivial question: "How does one put together the data for the look-up table and get it into the program?".

The question of cycling is left for another issue.

The new BOILER program is listed immediately below. I suggest that you examine it carefully, noting all differences from the the old program. Note that the FORTH word 2NDTM has been altered, both in its name (to 1ST-TM) and in its definition (a D1 cubed term has become D1 squared). The changes to 1ST-TM are not due to super devious clever programming, but to a realisation that:

a) 2NDTM actually came first in the mathematics of line V1, and

b) I had made a mistake while translating between programmes.

Other changes to the programme are due to the new form of ARCSIND1, which now uses a look up table instead of a recursive routine.

Let us look first at the structure of the look-up table itself. The word CREATE preceds (and creates, at compile time) the name AD1 which, when called, puts its address on the stack. You could say, in a sense, that AD1 is a "header" for the array of numbers which follow. The commas which lie between each number are FORTH words, each having the function of storing the immediately preceding (two byte) number in the next highest position in memory (byte values could be stored using the FORTH word C,).

If for instance, the 10th element of the array from AD1 (9, in this case) were to be accessed, the sum (20 + address of AD1) followed by a @ would put the value 9 on the stack.

Put a little more clearly, if each successive number in the array is two bytes further on from the beginning of the last number, the Nth number commences N*2 bytes on from AD1, or at an absolute address of address (AD1 + N*2).

So, I could define the word NVALUE to fetch and print the Nth value on the stack, as follows:

```
        : NVALUE AD1 SWAP 2* + @ . ;
```

Use of NVALUE would of course require that N be available on the stack.

The definition of ARCSIN clearly copies NVALUE

```
        : ARCSIN AD1 SWAP 2* + @ ;
```

and is subsequently used in ARCSIND1. Study of ARCSIND1

```
        : ARCSIND1 D1 @ DUP 0 > IF
        ARCSIN ELSE DNEGATE ARCSIN
        DNEGATE THEN ;
```

will show that the remainder of the definition merely deals with possible negative values for D1.

The table listed for AD1 is obviously not complete, since it must contain a value for each millimetre radius of boiler drum, i.e.1000 values (incidentally the values in the table are of ARCSIND1 in radians). The complete table has been omitted for the sake of space, since it can easily be generated using a calculator or of course, the CoCo.

Manual generation (sequentially punching each value of D1 into a pocket calculator) of the AD1 array is not a chore I would wish on anyone, so I have provided for automatic generation of the data (commas included) by two means. The first is the use of the BASIC program listed below, which not only will calculate the ARCSIND1 values for you, but also the SQRT values (more about that later).

The second is a listing of both sets of numbers to Graham, who will supply them via Rainbow on Tape.

To use the program, prepare your tape recorder and type RUN. The program run time will depend on whether you choose to use the optional screen display.

When the run is finished, an alarm will sound until you <BREAK> the program run. I have provided this feature since ARRAYS takes a few minutes to run, sufficient to make yourself a cup of tea or something.

FORTH LISTING FOR BOILER.

```
DECIMAL
0 VARIABLE D1
0 VARIABLE D2
0 VARIABLE Q

CREATE AD1 0 , 1 , 2 , 3 , 4 , 5
, 6 , 7 , 8 , 9 , 10 , 11 , 12 ,
13 , 14 , 15 , 16 , 17 , 18 , 19
,     ( Extra numbers here)     ,
1422 , 1429 , 1437 , 1444 , 1452
, 1461 , 1471 , 1481 , 1493 ,
1508 , 1526 , 1571 ,

: *PI 355 113 */ ;
: 1-D1*D1 10000 D1 @ 10 / DUP
  * - 10 / ;
: SQRT DUP 0 > NOT IF DROP 0
  LEAVE THEN DUP 2/ 2/ 2/ 2/ 1+
  BEGIN 2DUP / 2DUP - ABS 1 >
  WHILE + 2/ REPEAT ROT 2DROP ;
```

```
: SQRT(1-D1*D1) 1-D1*D1 10 *
  SQRT 10 * ;
: D1*ROOT D1 @ SQRT(1-D1*D1)
  1000 */ ;
: ARCSIN AD1 SWAP 2* + @ ;
: ARCSIND1 D1 @ DUP 0 > IF
  ARCSIN ELSE DNEGATE ARCSIN
  DNEGATE THEN ;
: D1E3/3 D1 @ DUP DUP 1000 */
  3000 */ ;
: LASTOFV1 D1E3/3 D1 @ - *PI ;
: V1 Q @ NEGATE 11519 + D1*ROOT
  ARCSIND1 + 6 * + LASTOFV1 - ;

: 1ST-TM 12000 NEGATE D1 @ DUP *
  M* 12000 M+ SQRT(1-D1*D1) M/ ;
: V2 1-D1*D1 *PI 1ST-TM + ;
: D22 D1 @ V1 1000 V2 */ - DUP
  D2 ! ;
: BOILER Q ! 40 0 DO D22 D1 @ -
  ABS 15 < IF ." DONE! " CR CR
  ." LEVEL DEVIATION = " D2 @ .
  ." M.METRE." LEAVE THEN D2 @
  D1 ! LOOP ;
```

BASIC LISTING.

```
4 '*********************
5 '*  ARRAY GENERATOR  *
6 '*   BY JOHN POXON   *
7 '*    07-208-0893    *
8 '*********************
10 CLEAR 1000
20 A=0:Z=0:A$="":B$=" ,"
30 CLS:PRINT"WOULD YOU LIKE A SC
REEN DISPLAY? THE DISPLAY MAY BE
ENGAGED OR   DISENGAGED AT ANY T
IME BY        PRESSING <Y> OR <N>
AND HOLDING MOMENTARILY."
40 OPEN"O",-1,"ARRAYS"
50 GOSUB110
60 GOSUB210
70 GOSUB460
80 GOSUB510
90 END 100 '
110 '***GENERATE ROOT VALUE***
120 A=A+.001
130 GOSUB 320
140 X=INT(1000*SQR(A))
150 IF Z = 1 THEN D=INT(1000*A):
PRINT"A="D"ROOT A="X
160 GOSUB380
170 IF A >= 1 THEN  RETURN
180 GOTO120
190 '***************************
200 '
210 '***GENERATE ARCSIN VALUE***
220 A=0:C$=C$+"     "+"0"
230 IF A >= 1 THEN X = 1571 : GOTO260
240 GOSUB 320
250 X=INT(1000*ATN(A/SQR(-A*A+1)))
260 IF Z = 1 THEN D=INT(1000*A):
PRINT "A="D"ARCSIN(A)="X
270 GOSUB380
280 IF A >= 1 THEN  RETURN
290 A=A+.001:GOTO230
300 '***************************
310 '
320 '***DISPLAY TOGGLE***
330 Z$=INKEY$
340 IF Z$ = "Y" THEN Z = 1 ELSE
IF Z$ = "N" THEN Z = 0
350 RETURN
360 '*******************
370 '
380 '***PROCESS X (WRITE TO CASS
ETTE?)***
390 A$=STR$(X)
400 C$=C$+A$+B$
410 Y=LEN(C$)
420 IF Y > 240 THEN  :  PRINT#-1,C$ :C$=""
430 RETURN
440 '*************************
450 '
460 '***CLOSE ARRAYS***
470 CLOSE-1
480 RETURN
490 '********************
500 '
510 '***SOUND END OF PROGRAM RUN***
520 S=50
530 CLS:PRINT"RUN FINISHED: PRES
S <BREAK> ANYTIME."
540 SOUND (S+150),2 : GOTO540
550 RETURN
560 '******************************
```

Testing of BOILER will reveal that the new ARCSIND1 routine reduces run-time considerably. Unfortunately the cycling at extreme values of Q (water content) continues unabated. Since BOILER works fine in PASCAL and BASIC, i.e. does not get hung-up by cycling, I am forced to conclude that the FORTH BOILER program is insufficiently accurate at extreme Q. I feel that the major culprit is SQRT, and am reasonably confident that either a look-up table for SQRT or double length calculations here and there will solve the problem. The sqrt (D1) values are provided so that you can experiment by replacing SQRT with a look-up routine, and thereby learn about FORTH by the very best method: by doing.

At this point I should re-iterate a comment which I made last month: that it is not terribly significant if BOILER is inaccurate at extremes of Q, provided that it is very accurate at small deviations from midline. My interest in the cycling is more intense from the viewpoint of anticipating hang-ups, and from an (unfortunate) tendency towards perfectionism.

That's it for this month. I hope that the article was useful to you. I am always happy to talk to anyone who would like to pursue specific issues further. Feel free to call me on 07-208-0893. Regards, John Poxon.

NOW! is the time to subscribe to Australian RAINBOW

Copies of back issues can be obtained, subject to the availability of stocks, by using this order form and marking clearly which issues you require to be sent to you.
Each issue costs $4.50 including postage and packing. Please enclose your cheque / money order made payable to: Australian Rainbow Magazine, PO Box 1742, Southport, 4215.

RATES

AUSTRALIAN CoCo / MiCo / softgold		AUSTRALIAN RAINBOW
□ $3.45	Latest per copy	$4.50 □
□ $19	6 months	$24.75 □
□ $31	12 months	$39.95 □

BACK ISSUES

MiCo —
First Issue Oct '83
□
$3 / copy

CoCo / MiCo / softgold
First Issue Aug '84
□
$3.45 / copy

AUSTRALIAN RAINBOW		
Feb '85-Apr '85	□	$3.95
Aug '84-Jan '85	□	$3.25
to July '84	□	$3.00

MiCoOz
(MiCo on Tape) $8 ea □ First Issue Dec '83

DISKS & TAPES

CoCoOz
(Aust CoCo on Tape)
□ Latest $8
□ 6 months $42
□ 12 months $75
First Issue Mar '83
or Debit my Credit Card Monthly □

Blank tapes 12 for $18 or $1.70 ea □
Cassette Cases , 10 for $5 □
Disks — $3.50 ea □ 10 for $29.99 □

Rainbow on Tape (program listings) $12 for month of □
Please note that RAINBOW on TAPE is issued irregularly
or Annually $144 □ or Debit my Credit Card Monthly □

BOOKS

Byte
SORRY
OUT OF STOCK

Help
Medium $9.95 □

Facts
Advanced $11.95 □

MiCo Help (MC-10 computer).
Medium $9.95 □

PLUS

□ Best of CoCoOz # 1
(Education)
□ Best of CoCoOz # 2
(Games)
□ Part 1 for 16K ECB
□ Part 2 for 32K ECB
(Some 16k CB on P1)
TAPE $ 10.00, DISK $ 21.95

BULLETIN BOARD

Let your computer talk to ours!
CoCoLink —
Annual Sub $29 □

□□□□□□□□□□□□□□

□ VISA BANKCARD □
MASTERCARD □

________________ Signature

□ CASH
□ CHEQUE
□ MONEY ORDER

Authorised
Amount $ ________________

BLOCK CAPITALS PLEASE

If you already subscribe to either Australian Rainbow or Australian CoCo please place Subscription No
□□□□□□□

Complete the section below with one letter figure or space per square

FIRST NAME SECOND NAME
□□□□□□□□□□□□□□□□□□□□

Address
□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□ PC □□□□

Telephone Number STD Code □□□□ Local Number □□□□□□□□□□□□

New Subscription □ Renewal □

PARIS RADIO ELECTRONICS

CALL
(02) 344 9111 VOICE
(02) 344 9511 BBS
TO ORDER

CoCo OS-9™FLEX ™

Free Diskette with Each $50 Purchase

COMPILERS

K-BASIC — A BASIC language to MACHINE language Compiler, includes an Extended Macro Assembler CCF — $292.00

PL/9 — by Graham Trott. A combination Editor/Compiler/Debugger Structured Programming at the "almost Assembly Language" Level. CCF — $292.00

INTROL C Compiler — Full Featured C Compiler. Linking Loader includes full Library Manager. CCF — $549.00

*Forth — Forth language for CoCo CCR — $99.95
CCF — $99.95

MICROWARE BASIC09 — for CoCo OS-9 systems. CCO — $149.95

MICROWARE C Compiler — for CoCo OS-9 Systems. CCO — $149.95

DYNA C — C Compiler from the authors of Dynastar and Dynaforms.
CCO — $99.95
CCF — $99.95

DATA BASE'S

XDMS — "Mainframe" User's say: "We don't have anything NEARLY as powerful as XDMS;" pure Assembly Language, F-A-S-T and small enough to operate on a single-sided 5" disk.
XDMS (V) I — CCF — $199.95 XDMS (V) II — CCF — $299.95
XDMS (V) III — CCF — $$399.95

RMS — Machine Language Data Base Manager — Super Fast.
CCF — $220.00
CCO — $180.00

OPERATING SYSTEMS

XEX — The latest Flex Operating System from Frank Hogg Laboratory USA. Includes hi res screen drivers, user definable keyboard, supports 128k upgrades, and 35, 40, and 80 track drives. CCF — $149.95

OS-9 for the CoCo. The hottest selling software to hit Australia. CCO — $99.95

DISASSEMBLERS

SUPER SLEUTH — Interactive: extremely POWERFUL!! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator. Label "Name Changer," and Files of "Standard Label Names" for different Operating Systems included. CCF, Obj. Only $89.95
CCF, w/Source $145.00 CCO, Obj. Only $89.95

DYNAMITE + — Excellent standard "Batch Mode" Disassembler Includes XREF Generator and "Standard Label Names" Files.
CCF, Obj. Only $89.95 CCO, Obj. Only $89.95

WORD PROCESSING

STYLOGRAPH III — A full Screen-oriented WORD PROCESSOR (what you see is what you get); also supports Daisy Wheel proportional printers and WORDPAK
CCF — $165.00
CCO — $165.00

STYLO-SPELL Spelling checker to suit stylo CCF — $99.95
CCO — $99.95

STYLO-MAIL Mail merge program for Stylograph CCF — $89.95
CCO — $89.95

DYNASTAR & DYNAFORMS — Another full screen Editor and Word Processor for the CoCo
CCF — $149.95
CCO — $149.95

XWORD — A powerful word processor that can be run under O-PAK, XSCREEN hi res drivers and WORD-PAK 80 CCO — $99.95

SPREAD SHEETS

DYNACALC CoCo's best and fastest Spreard Sheet system
CCR — $149.95

UTILITIES

UTILIX is a unix like utilities package for OS-9. It includes 15 different utilities to aid you in the manipulation of text files. CCO — $75.95

FILTERS KIT No. 1 — Eleven utilities used as filters for OS-9
CCO — $39.95

FILTERS KIT No. 2 — Ten more utilities, APPEND, CONFIRM, FF FORCERROR, MACGEN, NULDEVICE, REP, SIZE, TOUCH, and unload
CCO — $39.95

SDISK and BOOTFIX — Use 40 or 80 track, single or double sided drives with OS-9. Create a bootable double sided system disk. CCO — $59.95

HACKERS KIT No. 1 — This package contains a set of programs useful to anyone working with assembly language, trying to unravel the operating system or other assembly language programmes, or customizing their own.
CCO — $59.95

XSCREEN — Hi Res Screen driver for OS-9 CCO — $39.95

O-PAK — Hi Res Screen driver for OS-9 CCO — $56.95

SEARCH AND RESCUE — disk utilities to search and retrieve lost data.
CCO — $56.95

XTERM — A communications program for use with XSCREEN.
CCO — $82.95

MEMORY MINDER — Disk diagnostics program. The most comprehensive program available to analyse your disk drives
DOUBLE SIDED CCR — $149.95
SINGLE SIDED CCR — $139.95

XMENU — This is a program that creates a menu driven environment for the coco under OS-9. CCO — $69.95

RAM-DISK — This program simulates a disk drive, utilizing spare RAM memory. Very fast execution of commands. For use with 64K or 128K memory, Flex, XEX, or OS-9
CCF — $49.95
CCO — $49.95

MAGAZINES and BOOKS

AMERICAN RAINBOW — Direct from the USA the same month of publication for the same price as the newsagents. Why wait! $6.80

AMERICAN HOT COCO — Airfreighted direct each month. Subscription available. $5.80

THE OFFICIAL BASIC09 TOUR GUIDE $29.95

HARDWARE

PBJ WORKPAK 80 — an 80 column card to fit in the expansion port of the CoCo. Will run under Flex or OS-9. $199.95

PBJ WORDPAK II — The ultimate wordpak. Includes smooth scrolling, soft ware controlled video switch, improved character set, hardware inverse video and up to 8K video RAM. $249.95

PBJ CC-BUS — A six slot software selectable expansion bus. Will take your disk controller, WORDPAK 80, GAMES Cards etc. $219.95

PBJ PC-PAK — This is a dual function cartridge which contains a centronics compatible parallel printer port and/or a battery backed real time clock
w/ real time clock $199.95
w/o real time clock $ 99.95

PBJ 2SP-PAK — A 2 port RS-232 serial interface with programmable baud rates up to 19.2K. Each port can also be interrupt driven. $119.95

DISK CONTROLLERS

J&M Disk Controller with JDOS Disk Basic	$219.95
RADIO SHACK Disk Controller Ver 1.1	$199.95

MISC

CC-VID Monochrome video output board	$ 35.00
MC6809EP Processor chip	$ 24.00
MC6883 SAM chip	$ 24.00
4164-150 nSec, 64K chips	each $ 7.80

T/M FLEX is a trademark of Technical Consultants
T/M OS9 is a trademark of Microware
PRICES SUBJECT TO ALTERATION
WITHOUT NOTICE

PARIS RADIO ELECTRONICS
161 BUNNERONG RD, KINGSFORD 2032
P.O. BOX 380, DARLINGHURST 2010
Ph: 344 9111

Availability Legends —

CCF	Color Computer FLEX
CCO	Color Computer OS-9
CCR	Color Computer RS DOS

PRICES DO NOT INCLUDE
POSTAGE AND PACKAGING

AUSTRALIAN RAINBOW MAGAZINE
REGISTERED BY AUSTRALIAN POST —
PUBLICATION NO. QBG 4009
AUSTRALIAN COCO/softgold
REGISTERED BY AUSTRALIAN POST —
PUBLICATION NO. QBG 4007.
PO BOX 1742,
SOUTHPORT. QLD. 4215.

user group CONTACTS

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

ADELAIDE	JOHN HAINES	08 278 3560
ADELAIDE NTH	STVN EISENBERG	08 250 6214
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	TOM STUART	067 72 8162
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
BANKSTOWN	CARL STERN	02 646 3619
BLACKTOWN	KEITH GALLAGHER	02-627-4627
BLACKWATER	ANNIE MEIJER	079.82.6937
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BOWEN	TONY EVANS	077 86 2220
BRASSALL	BOB UNSWORTH	07 201 8659
BRIGHTON	GLENN DAVIES	08 296 7477
BRISBANE EAST	ROB THOMPSON	07 848 5512
BRISBANE SH	PATRICK SIMONIS	07 209 3177
BRISBANE SW	GRAHAM BUTCHER	07 376 3400
BRISBANE WEST	BRIAN DOUGAN	07 30 2072
BROKEN HILL	DEAN PARADICE	080 6701
BUNBURY	Call Tandy for local details.	
BUNDABERG	JIM McPHERSON	071 72 8329
CAMBERWELL	TONY BALDWIN	03 728 3676
CAMDEN	KEVIN WINTERS	046.66.8068
CAMPBELLTOWN	LEO GINLEY	02 605 4572
CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 88 9226
CAULFIELD	JEFF SHEEN	03 528 3724
CHATSWOOD	BILL O'DONNELL	02 411 3336
' or	MARK ROTHWELL	02 817 4627
CHURCHILL	GEOFF SPOWART	051 22 1389
COLYTON TEENS	DWAYNE MANSON	02 623 5805
COOMA	ROSS PRATT	0648 23 065
COORANBONG	GEORGE SAVAGE	049 77 1054
DANDENONG	DAVID HORROCKS	03 793 5157
DARWIN	BRENTON PRIOR	089.81.7766
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DONCASTER	JUSTIN LIPTON	03 857 5149
DUBBO	GRAEME CLARKE	068 89 2095
FORBES	JOHANNA VAGG	068 52 2943
FORSTER	GARY BAILEY	065 54 5029
FRANKSTON	BOB HAYTER	03.783.9748
GIPPSLAND STH	PAT KERMODE	056 74 4583
GLADSTONE	ALBERT VAN GORKUM	079 72 2353
GOLD COAST	SHERYL BENTICK	075-39-2003
GOSFORD	PETER SEIFERT	043 32 7874
GOULBURN VALLEY	TONY HILLIS	058 59 2251
GRAFTON	DAVID HULME	066.42.0627
GREENACRES	BETTY LITTLE	08 261 4083
HASTINGS	MICHEAL MONCK	059.86.8288
HERVEY BAY	LESLEY HORWOOD	071 22 4989
HILLS DIST	DENNIS CONROY	02 671 4065
HOBART	BOB DELBOURGO	002 25 3896
HORNSBY	ATHALIE SMART	02 848 8830
IPSWICH	MILTON ROWE	07 281 4059
JUNEE	PAUL MALONEY	069 24 1860
KALGOORLIE	TERRY BURNETT	090.21.5212
KENMORE	GRAHAM BUTCHER	07 376 3400
LEETON	CHRIS NAGLE	069 53 2969
LITHGOW	DAVID BERGER	063 52 2282
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACKAY	LEN MALONEY	079511333x782
MACLEOD	ROBIN ZIUKELIS	03 450211x465
MacQUARIEFIELDS	KIETH ROACH	02 618 2858
MAFFRA	MAX HUCKERBY	051 45 4315
MAITLAND	LYN DAWSON	049 49 8144
MARYBOROUGH	NORM WINN	071 21 6638
MELBOURNE	JEFF SHEEN	03 528 3724
MELTON	MARIO GERADA	03 743 1323
MILDURA	SCOTT HOWISON	050 23 6016
MORPHETTVALE	KEN RICHARDS	08 384 4503
MOREE	ALF BATE	067 52 2465
MORWELL	GEORGE FRANCIS	051 34 5175
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280
MUDGEE	BRIAN STONE	063-72-1958
MURGON	PETER ANGEL	071 68 1628
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NEWCASTLE	' LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
ORANGE	STEVE LOVETT	063.62.4025
' or	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PENRITH	ALEX SCHOFIELD	047 31 5303
PERTH	IAN MACLEOD	09 448 2136
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
PORT MacQUARIE	RON LALOR	065 83 8223
PORT NOARLUNGA	ROB DALZELL	08 386 1647
PORT PIRIE	KEVIN GOWAN	086 32 1368
RINGWOOD	ANDREW RAWLINGS	03 700 4498
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
ROSEVILLE	KEN UZZELL	02 467 1619
SALE	BRYAN McHUGH	051 44 4792
SANDGATE	MARK MIGHELL	07 269 5090
SCARBOROUGH	PETER MAY	07 203 6723
SEACOMBE HTS	GLENN DAVIS	08 296 7477
SHEPPARTON	ROSS FARRAR	058 25 1007
SMYTHESDALE	TONY PATTERSON	053 42 8815
SPRINGWOOD	DAVID SEAMONS	047 51 2107
STURT	MARY DAVIS	08 296 7477
SUNBURY	JACK SMIT	03.744.1355
SUTHERLAND	IAN ANNABEL	02 528 3391
SWAN HILL	BARRIE GERRAND	050.32.2838
SYDNEY TEENS	ROD HOSKINSON	02 48 5948
SYDNEY EAST	JACKY COCKINOS	02 344 9111
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
TONGALLA	TONY HILLIS	058 59 2251
TOOWOOMBA	GRAHAM BURGESS	076 30 4254
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
TRARALGON	MORRIS GRADY	051 66 1331
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
WAGGA WAGGA	CES JENKINSON	069 25 2263
WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
WONTHAGGI	PAT KERMODE	056 74 4583
WYNYARD	ANDREW WYLLIE	002 35 1839
YARRAWONGA	KEN SPONG	057 44 1488
SPECIAL INTEREST GROUPS		
' BUSINESS		
BRIZBIZ	BRIAN BERE-STREETER	07 349 4696
' TDP - TELEWRITER, DYNACALC PROCOLOR		
BRISBANE	GEOFF TOLPUTT	07 44 6084
' OS9 GROUPS		
BANKSTOWN	CARL STERN	02 646 3619
BRISBANE	JACK FRICKER	07 262 8869
KALGOORLIE	TERRY BURNETT	090.21.5212
MONARO	FRED BISSELING	0648 23263
PENRITH	BOB THOMSON	047 30 2468
SYDNEY EAST	JACKY COCKINOS	02.344.9111
SYDNEY NTH	MARK ROTHWELL	02 817 4627
BLAXLAND 128K	BOB THOMSON	047 30 2468
' MiCo GROUPS		
CARLISLE	STUART HALL	08 361 1922
LITHGOW	DAVID BERGER	063 52 2282
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
ROCKHAMPTON	TIM SHANK	079 28 1846
SYDNEY	RAJA VIJAY	02 519 4106
' TANDY 1000 / MS DOS		
BRISBANE	BRIAN DOUGAN	07 30 2072
SYDNEY	ROGER RUTHEN	047.39.3903
' FORTH		
PORT LINCOLN	JOHN BOARDMAN	086 82 2385

November's AUSTRALIAN CoCo

- * Quick on the Draw
- * Water Wars
- * Tape Jacket Maker
- * Catapillar
- * Number Run
- * CADding About
- * Going Troppo
- * Certificate or Award
- * Wazzdere

PLUS
Martha's Quickies,
User Friendlier,
Scoreboard,
MiCo - for yourMC 10
"softgold" - for your
T1000, T2000, T100
& T200!