

AUSTRALIAN COCO

\$3.25

RAINBOW

GO CO

MICO



AUSTRALIAN CoCo

RAINBOW

GoCo

MiCo

SCREEN DUMP2	ZERO SLASHER20	MAGIC WAND64	LETTERS74
AUSTRALIAN COCO4	CIRCUIT INTERFACE .22	FORTH67	DEMON76
CHRIS6	COMPUTER SIMULATIONS 29	TANDY TO	THE BRIAN MCLACHLIN
LETTERS7	UNCLE BERT31	APPLE68	VIDEO SHOW .79
COMPUTER EXPO9	COOKING WITH COCO ...34	2000 REVIEW70	
HEARD ON THE WIRE9	PAY THE RENT41		
ON THE NEWSTANDS10	ANGLES OF BASIC		
CCR-82 REVIEW11	& LOGO50		
OS-811	DUALING CASSETTES ..53		
MONITOR MOD12	OPERATING SYSTEMS 55		
LITTLE GEM13	DEATH CAVERNS ...57		
ORBQUEST14	REVERSE59		
NOUGHTS & CROSSES15	VARIABLES REVISITED .60		
TV SCRIBBLER17			
FORTH17			
SCOREBOARD18			
CLUB NEWS19			

AUGUST 84
No 38



SCREEN DUMP

AUSTRALIAN COCO
**RAINBOW
GOCO
MICO**

Dedicated with Love
to the Memory of
Greg Wilson.

AUSTRALIAN
EDITOR & PUBLISHER
Graham Morphett

CO-EDITOR
Kevin Mischewski

EDITOR'S ASSISTANT
Christine Lucas

AND GRATEFUL ASSISTANCE
FROM
*Peggy Annabel Tina Lucas
Rod Hoskinson Sheryl Bentick
Annette Morphett Greg Hains
Bram Dougan
Project Internal Linings
Richard & Judy*

OS-9.

Kevin HOLMES is the man to contact for information on OS-9. All Rainbow OS-9 content is being sent to him and in turn to those interested, along with a monthly newsletter.

Kevin has joined the U.S. users group and wants to form a local branch of that group here to give you access to all their Public Domain software and keep you up to date with the latest news.

Kevin appreciates any assistance you can provide in the form of software or hints.

His address is:-

39 PEARSON ST.,
NARARA. N.S.W. 2250.

Printed by Australian Rainbow
P.O.Box 1742,
SOUTHPORT, QUEENSLAND.

As you will be aware from last issue, our friend Greg Wilson, passed away on 8th June, under most unpleasant circumstances.

This issue in particular, and all further issues too, is / will be dedicated to his memory.

Greg was a man with a purpose. He was dedicated to assisting every CoCo owner, and to this end, he worked tirelessly for you. He shall be missed. I know that you would all have me pass on to Helga our collective condolences. She has been very brave, and is coping quite well, considering the circumstances.

Helga has allowed the publication of a number of letters in this issue, that she has received since Greg's death. If you would like to record your feelings for Greg too, we will make space available in the September issue for you to do so. September is close to being ready for the printer, so you will need to respond QUICKLY.

With other people in another part of this magazine continuing on this note, I will now go on to tell you what's happening.

Firstly, Rainbow lives on! There was no way that we were prepared to let Greg's love go down the drain. From this issue, the magazine is being operated by myself with an enormous amount of assistance from Kevin Mischewsky. We are members of the Gold Coast group, and have had our CoCo's for about two years. We are still learning, like everyone else, and have a very long way to go before we can say that we know the answers to your questions. So the first change that you'll notice about the magazine will be our greater reliance on you! Frankly, your assistance is really needed to continue this job.

We are to be ably assisted by Christine, Greg's secretary, who has packed her things, and moved up to the sunny state to continue her work for us. (Which, if you know Christine, means mainly telling us why we can't do things this way or that!)

To be a proposition, we have decided that the magazine has to undergo some minor format changes.

In the first instance, our printer can't do all the tricks that Greg's printer could with the funny size, so I figure that you've already noticed the first change! We apologise to those that have a system for keeping the larger format; we think that the new size will be, in the long run, more in keeping with standard practice, and hence will certainly help to maintain costs.

The next immediate change is the incorporation of MiCo and GoCo magazines into the Australian Rainbow for now.

There are several definite advantages to such a format, apart from the saving in dollars. These include the fact that you will now get all the news! How many missed the great article on Modems in GoCo recently? Not to mention the great little games that are often in MiCo! So, nothing will be lost, but, we feel that you may gain quite a bit -- 'cos, the other change is that we are going to at least 88 pages! At the time of writing, details were still sketchy, so it may yet be that next month's final size will be more like 96 pages!

After we learn how to do that, we will be setting up an electric noticeboard and ideas bank. To this end, those of you who sell communications programs or modems, are invited to submit proposals for such a system.

We figure that we all have the technology to send and receive information by modem, so why not use it to improve Australian Rainbow. This service is likely to be in place by Christmas, and should speed your article and program submissions. In addition, it will ultimately reduce the cost of CoCoOz to those with modems, because we will be in a position to send CoCoOz by modem.

The next change was one that Greg had already foreshadowed. He had planed for an Australian version of Rainbow. In fact, it happens that we of the Gold Coast Group were to be presenting that first magazine, with all the copy and programs coming from our club. That will not happen now. What we will do, however, is blend the more interesting parts of the American Rainbow with Australian programs to produce what we think will be a most interesting magazine.

So, how about it? We need not only your program, not only an article telling us not only how the program works, but also how it is that you came to write the monster in the first place! As Greg would have said, let's see you get your bums into gear and write those articles and programs!

Finally, we announce that there will be a Color Computer Conference on the Gold Coast next year. We haven't set a date yet, but June looks good if we can get the accommodation! Again, let us know your thoughts on this matter.

OK, as my CoCo would say, that's about it for now, this issue has been the work of some very frantic and hard work by many people, most have been named already, however Peter Dance from Project Internal Linings, Sheryl Bentick, Chris's 'rats' (I really don't know why she calls them that!), Greg and Tina, and of course, Kevin, deserve a special mention at this point. Lastly, Christine, thank you especially both for staying on, and for your special assistance this issue - we hope that eventually you will get used to us!



Introducing:

AUSTRALIAN COCO

Welcome to Australian CoCo!

or How to Move a Magazine and Survive!

Little did we realize that as we wrote some of the articles in this magazine that we were really practicing for the day when we would have to do the lot!

We heard of Greg's death from our friend Patrick Simonis on the following Saturday, and I guess like every one, we were a little stunned, and quite frankly, felt a little sorry for ourselves too.

After it had sunk in, we started to think about the magazines. Kevin works nights and I have my own business and we were a little like most others in that we were prepared to 'leave it to someone else', but when after a couple of weeks we heard that no way had been found to keep the magazines going, we decided to talk to Helga.

Kevin and myself traveled to Sydney and saw Helga. It wasn't an easy thing to decide but we eventually arranged to take the magazines over.

When we got back from talking to Helga, I immediately set about looking for a place to house 'Rainbow'. Just four weeks previously a Real Estate salesman had been winging to me about how hard it was to shift rental properties. Yes, you guessed it! Murphy struck!

So a week and a half later, I still hadn't seen a rentable house!

Then one became available that Goldilocks would have said was 'just right', except that it stank. Literally, a family of dogs had lived in it, and had had a litter of puppies in it which no one had gotten around to cleaning up after!

We were desperate, and at this stage a good friend of mine, Peter Dance, of Project Internal Linings came forward and 'took over' the house. He arranged to clean the house top to bottom, paint it entirely inside, recarpet and revinyl the interior, and remove a solid block of rubbish, 20ft X 8ft X 8ft! Then to top it off, he supplied a truck and driver to cart the magazine and Chris up to the Gold Coast.

So we think he's a pretty top guy!

The day the truck arrived we unloaded, and started - admittedly slowly at first - on this issue.

Chris has had to get used to the Queensland culture - it's a touch different to Kings Cross! - but we're all working well together and I'm very pleased that the mag is out at all!

I believe in giving credit to those who assist in any enterprise, but the list of those who have lent their talents to



this one is already becoming formidable! I have to say that if I've left your name off the following list, and you've helped us in the last three weeks, that we love you too, it's just that my memory gets effected by lack of sleep!

To Kevin, Nicholas, Glen, Chris, Peter, Greg, Tina, Helga, Peggy, Sheryl & Jim, Richard & Judy, my father (Bill), Brian (or is it Brain?), Patrick, Alan & Lisa, Fiona, Greg Birch, Bill McDougall, Peter Issacs, and especially Annette, my lovely wife, thank you all for your assistance. It goes without saying that the magazine only exists because you did your part.



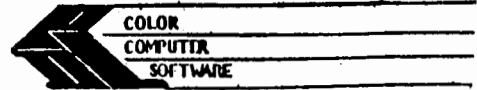
Necessarily, we're a little out of gear with Australian CoCo this month. We promise to improve!

A new operating system to beat both OS9 and Flex has become available. Called OS8, those who have used it say that there are definite similarities to both those other systems. You will find the first hints on how to utilise this very effective system overleaf.

Also, we welcome Martha Gritwhistle and her very handy hints. We are unlikely to be able to fit much of her work in this issue, however I'm sure that we can include at least something of her work. If you have similar thoughts from time to time, I'm sure Martha will be pleased to include your thoughts in her columns in the future.

Other new columns include the "Club News" and "Heard on the Wire" columns. These are intended to keep you abreast of the latest in the Australian CoCo World.

Hope also to have a Tandy News column - but gotta get the news before we can rite it if ya get me drift!



**NEW!!!
AUSTRALIAN
GEOGRAPHY**

The first 3 programmes in a set aimed at all geographic aspects of each individual state. Programmes cover general agriculture, weather, towns, industry, rivers etc. in tutorial and questionnaire form. A splendid aid to teaching Australian geography in school or home.

Two programmes on each tape.
All 16Kecb \$14.95 each.

**SOUTH AUSTRALIA
NEW SOUTH WALES
QUEENSLAND**

Plus all the latest arcades and utilities at the best prices.

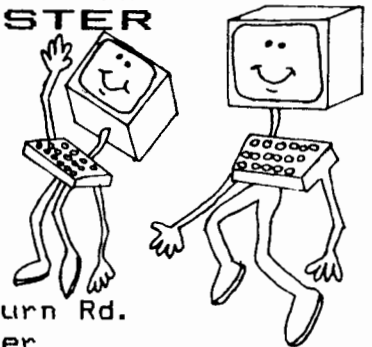
R.D.L. COLOR COMPUTER SOFTWARE

31 NEDLAND CRES.,
PT. NOARLUNGA STH.,
S.A. 5167
(08) 386 1647



**Spectrum Software
DONCASTER**

- Adventures
- Arcade
- Business
- Utilities



Mail orders

1/234 Blackburn Rd.
East Doncaster
Melb. Vic. 3109.
Call PETE
03-842-1205



Most of you people out there who read Rainbow never met Greg. Some of you spoke to him on the phone. A few of you knew him personally. Then there those who worked with him. I was one of those.

I have worked with Greg since last August. We both wondered what had struck us. I had several years of Business Management behind me. Greg made me feel like I was straight out of kindergarten. In fact, Greg made it quite clear that I WAS straight out of kindergarten. However, as the months went past we came to compromise on different issues. I tried so hard to absorb what Greg was trying to teach me and Greg tried so hard to be patient. I was successful in learning (however slowly) and Greg was successful with patience and in doubling his teaching efforts. Not only did we work together, but there was empathy. A rare commodity in my life.

Now I am up here. My thanks go to so many people:

Graham and Kevin - who must be mad to contemplate a Kiwi working on an Australian mag. (Greg thought I was mad too).

Then those people who helped pack Rainbow and my household ready for moving with just two weeks warning.

Fiona - who worked for Greg part-time for two years - helped with studio gear. Bill Morphett and Mrs. Morphett spent a freezing cold day in one of the garages starting with packing mags into cartons. 20,000 mags and books is one whole heap of books. During the first weekend my friend Greg Birch packed and stacked cartons (nearly 200 of them) into the other garage ready for the Removal truck.

Then came Sunday 15th July. Truck arrived. Graham, Bill, Lisa, Allan, Greg B. and my son Greg stacked Rainbow gear. Then to my house. We ran out of room and Greg B. hired another truck, organised a few days off work, stacked 2nd truck and drove me up here. Tina, my daughter (my kids, Greg and Tina hereon referred to as "rats") came up in the car with Graham and our two cats. Lisa and Allen came up in big truck. Greg B and I had our dog with us. My elder rat (Greg) went on the train with my car. We all met up here sometime Monday.

I think Bill went home and shook his head in wonderment at the lunacy.

Once here everyone set to unpacking.

Now I am flat out helping with this issue. I miss Greg Wilson. I feel that he is still with us all and hope that you can understand that the people involved with Rainbow loved Greg very much and we all intend for Greg's work and enthusiasm to continue.

I know nothing about computers (Graham is about to change that - this is 'SCRIPSIT' and I seem to be mastering it) so this little epistle (like it Graham?) is non-computer. But then again, this issue is a different one isn't it?

I would like to say a special thank you to my two children Greg and Tina, Greg Birch and to Greg Wilson. I love you all.

Christine



LETTERS

GREG

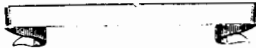
To whom it may concern,

I was saddened to hear of Greg's death in July issue. Although I have never met him in person, I feel that I know him, if only a little bit, from his comments, articles etc. in Australasian Rainbow. He was a great man who did his best to further his and others interests in the Color Computer. Everyone owes him a lot for his unselfish contribution to others of his time, materials and knowledge. He will be sorely missed by many.

My deepest commiserations and best wishes for the future.

Yours faithfully,
Raymond Hilder.

ps: I hope Rainbow will continue for many year to come in memory of Greg, so that others can draw CoCo users together, as he sought to.



To Greg's Family and Friends,

Please accept my sincere condolences on your deep loss.

Although I never had the fortune to meet Greg I recognised his deep commitment, expert knowledge and sense of fair play.

He will be greatly missed by us all.

Yours sincerely,
Graham Fenton.



And from Adam and Joanne Gilbert we received a card which read -

"We never met, but he brought sunshine into our lives. We will miss him."
The envelope was addressed to -

"To those who loved Greg."

The following letter especially touched our hearts:

Dear Team,

I have just heard the terrible news about Greg. Although I am only a new member of the "family", I have appreciated all that Greg has done (with your help) to make learning to live with the Color Computer a truly delightful experience.

Last month, I ordered a year's back copies of "Rainbow" (foolishly forgetting to enclose my order for the ensuing year). I have devoured them all, and I feel able to communicate with real integrity as to the warm genuine human being he was.

So I find myself grieving deeply with you, and with hundreds of users across the nation, whose hearts will be heavy at this time.

Please accept my deepest sympathy, as one who represents the "beginners" whom Greg sought so generously to help.

He has left behind him a marvellous memorial - a living memorial in the hearts of his wide "family", and in the many users' groups, as well as the "Australian Rainbow" and associated publications. This is a truly remarkable achievement. No doubt, when our Users' Group next meets, we will be trying to put together a more tangible expression of our regard for Greg's memory.

I hope you will be able to carry on the good work. As a token of my confidence in you, and my support for you, I have enclosed a subscription form for the ensuing twelve months.

With warmest regards and deepest sympathy,

Doug Brandon
University of Queensland
ST LUCIA

Australian CoCo

Dear Helga,

Enclosed is a cheque for renewal for Rainbow and subscription for Aust. CoCo. I hope the amount's right, if not let me know. Yes I did see the free trial, etc. but I know I will want it so it's easier to pay for it all at once.

Yours sincerely,
Geoff Spowart.

Madness and the Minotaur

Dear Greg,

Last holidays I bought 'Madness and the Minotaur' by Spectral Associates from Tandy. I am having trouble solving it as I suppose many people do. Any clues or hints to help solve it? In the March '83 issue of the Rainbow you listed a program called 'Ad-dicts', I typed that in and ran it, but the printer printed rubbish all over the paper.

Do you know how to slow the CPU of the CoCo?

I recall in one of the issues of the Rainbow, a review which said that 'Planet Invasion', by Spectral Associates could be played by one or two players. How?

Thanks. Keep up the good work and keep those great programs rolling in!!

Many thanks,
Adam Robinson.

Chess

Dear Mr. Wilson,

I have recently purchased a Tany TRS80 standard colour computer and have been using their chess game cassette. Have found that at Level 8 there is an insufficient challenge. I am informed that you have a Rainbow mag which will improve the standard.

Could you please send me some information in regard to this matter.

Thanking you in advance for your assistance.

Yours sincerely,
J.A.McILRATH

Dear Greg,

I have written the following program, hopefully to be put into the Rainbow.

It has lines going down to the bottom of the screen from random co-ordinates. It could be Skyscrapers, Fireworks or little dots going down the screen (I don't know).

```
10 PMODE4:SCREEN1,1:PCLS
20 X=RND(252):Y=RND(191)
30 CIRCLE(X,Y),1
40 Y=Y+2
45 IFY>191THEN20
50 GOTO30
```

Yours Sincerely,

Adam Albert. (age 11).
Yanderra. N.S.W.

**D & L****WILSONS**

serving melbourne's east

FROM SPECTRAL: Robot Battle, Space Invaders, Whirly Bird Run.
FROM COMPUTERWARE: Doodle Bug, Eloc Head, Megapede, Shark Treasure
Also TOM MIX,
MARK DATA,

\$ 28.95 ea

JULY SPECIAL!! FREE BLANK TAPE FOR YOUR BACK-UP PROGRAM

doug & louise wilson -
6 stafford st -
blackburn - phone 898-4521



HEARD ON THE WIRE

Your Phone Calls This Month

COMPUTER EXPO '84

THE GOLD COAST COMPUTER FAIR

The Gold Coast meet was invited to be a part of this Expo which was held in June.

Patrick Simonis came down and assisted by displaying many of his programs and we had help from a heap of children from a nearby school too.

And what a show! About 6000 people came through on the day and saw virtually every major personal computer on the market. There were a lot of peripherals and a large variety of software being shown.

Of course the trade stands weren't too pleased to see us - we were lent 4 computers with monitors by Tandy Sundale and had an additional computer from within our club. And we had people! They stood three deep whilst other stands had few to none! The Commodore man had a permanent scowl fixed on his face all day!

Next year the Expo will be held over 2 days and will be bigger than ever. We had many enquiries for membership of our club this year, next year we'll be more prepared.

Additionally, next year, the CoCoConf will be held in parallel with the Expo and all CoCoConf delegates will have free passes into Expo.



Of course this month has been a very sad and very busy month. And the phone calls have reflected that aspect too.

We appreciate the trouble many of you went to obtain the new phone number and update subscriptions or order back issues. All orders taken this month by phone or mail will be attended to at the beginning of August.

Barry Cawley of the East Brisbane meet called to say that he has taken over CoCoBug, the Brisbane user's group magazine. His phone no. is 07-390-7946.

Tony Hillis - the Tongala contact, phone 058-59-2251 needs a Typing tutor program - can anyone help?

Whilst on that subject, has anyone given any thought to a word processing program for the MC-10? We could use one here particularly if it saved to ASCII.

Many Agents have been kept waiting for their July supplies. We found it impossible to extract the agents' addresses from Greg's data base - we will attend to this as a matter of urgency.

Your response to Australian CoCo was beyond wildest expectations. When we split the magazine again, it is likely that this will be the first mag to be produced separately.

Patrick Simonis, one of the growing number of Software Agents, had an accident on his motor bike recently. Hope all's well mate.

Finally, may I say that you all deserve a box of chocolates each for your understanding and kindness - both to Helga and to Annette and myself. You've all done yourselves proud. Moving into Greg's shoes is not going to be easy, but by creating a family atmosphere about the magazine and it's subscribers, he sure made it a lot less tough than it could have been!

ON THE NEWSTANDS

By **Bill McDougall**

The Rainbow is, undoubtedly, the premier publication as far as COCO users are concerned, but there are several other magazines devoted to our machine which, contain much material of merit. I am particularly fond of COLOR COMPUTER MAGAZINE, which is gaining strength, probably because of the influence of Dennis Kitz, who has been with us Tandy enthusiasts since the very first issue of 80-MICRO. Dennis swung over to the COCO about two years ago and has presented much in hardware, especially in music synthesis, and software to both the Model I/III and the COCO.

As I am a regular peruser of computer literature, I have volunteered to contribute a general overview of items of interests in some of these other magazines. If you find you are interested in a closer look at something in particular, then it is up to you to obtain a copy of said issue.

Paris Radio at Bunnerong, N.S.W., regularly receives air freighted shipments of HOT-COCO from the U.S.A., hot off the presses and for only a few cents more than you pay for a stale issue from the newsagents. I don't know of a source, other than the newsagencies, for COLOR COMP. MAG. at present, but if there is enough demand I hope Jacky will make that available too.

Luckily, I happen to have the MAY issues of both these magazines at the moment, so they will be the primary interests of this review.

HOT COCO reviews the following software:

SuperstKt - a statistical analysis package on cassette.

Master Writer - M/L word-processing at a low price.

Mathmenu - engineering and scientific math functions, including an RPN calculator.

Color-80 BBS - run your own bulletin board service.

Speak-Up! - software-based speech synthesis.

Keyboard Beeper Cartridge - emits a beep with each key pressed, plus other enhancements.

The "theme" of the May issue is science. Here we have a good selection of programmes and articles, including one on building a twelve bit A/D converter and using COCO and standard chemical lab equipment to conduct and record experiments. Another, physics orientated, on random distribution and point setting for star clusters, and an astronomy programme to calculate sidereal time, with adjustments for GMT.

A chemical programme sets test questions and rewards by drawing the BOHR orbital model of the element on the screen.

If you can imagine up to five pendulums of differing lengths, joined end to end, and set to swing with a variety of initial velocities, then you may also be able to calculate their individual positions, angles and velocities at any given time. If not, then Multiple Pendulums, your COCO and printer will do it for you.

I would like to go back a month and mention an article of note in the April issue of HOT COCO. Called DISK DECISION, it discusses the whys and wherefores of purchasing a non-Tandy disk operating system. There are several bugs in Disk Basic mentioned, some good, good reasons to stay with Tandy, and some good ones to go elsewhere. Well worth reading if you are going to disks, especially as at least one other controller is available locally, and there is money to be saved.

Before leaving the May issue of HOT COCO, read its realistic review if you are thinking hard in terms of OS-9. It will reinforce what Frank Hogg said in Rainbow, but if you think he may have presented a view with a vested interest, this will give you second thoughts.

Now to COLOR COMPUTER magazine, and the second part of Dennis Kitz's Eprom Burner. This part contains the software driver, the hardware was presented in March, and the whole package is available in either kit form, built and tested, or you can buy just the PCB. Of course, you can just follow this series and hardware it. This issue also presents an alternate Eprom burner to build.

Jake Commander, another long-time TRS-80 buff and former editor of 80-MICRO, is well in to his series on ROM disassembly and several pages of completely dissected code are included. The Edit, FN and USR commands of ECB are

only a portion of this month's article.

The bulk of the magazine is concerned with Languages, with "C" being well featured. It includes a published listing for TIC-TAC-TOE written in this language. LOGO also is well covered.

These issues of these two publications should be at the newagencies by the time you read this, so keep your eye out for them. Don't forget, there is also Color Micro Journal dedicated to the COCO and several others devoted to 6800 systems, which are including more and more on our machine, particularly with Flex and OS-9 giving us more in common with the larger systems. The number, and presentation, of these magazines reflects the growing strength of the Tandy Color Computer.

PRODUCT REVIEW

CCR-82 CASSETTE RECORDER

A new cassette recorder for use with CoCo has been a long time coming. For some time Greg was buying any of the old CTR80A cassette recorders that he could lay his hands on - and with good cause! They were really the only ones that could stand the work that he gave them.

But now comes CCR-82, certainly in the small time that we've had to review it, the best little deck we've seen for CoCo.

Why?

Well er, price - \$59.95!!! Just like the price of the cheap computer's recorders - but unlike their recorders, it gives you a heap of additional features.

Some of these features include:

1. A set volume level for loading programs. This level can be adjusted for really stubborn tapes.

2. Switches to monitor the output of the tape and to control the pause function.

3. Low battery light.

4. A data light which indicates when data transmission is taking place.

The unit itself is quite small and therefore fits into place on a crowded desk well. We liked the 'feel' of the keys and the definite lock on each key.

We also liked the addition of a computer to recorder cord in the packaging, and as is usual with Tandy, the very full and informative set of instructions with the unit. (There aren't many around these days that will sell you a recorder that has a full set of specifications included with the price!)

We thought that Tandy could have included the power pack, as they have previously, but this aside, found little to criticize in the unit.

As we said - a sweet little unit that we at Rainbow intend to adopt. We'll let you know how she performs in the longer term!

Our thanks to Peter at Tandy, Sundale for the use of his one and only CCR-82 in this review - thanks mate, but don't expect it back, we'll buy it!

Graham

OS-8

Have you heard the one about the sadist and the masochist?
The masochist said to the sadist
"Beat me, Beat me."
The sadist said "No"

Why were the flies playing soccer in the saucer?
They were playing for the cup.

MONITOR MOD FOR YOUR T. V.

Peter Blackman

An article in Australian Rainbow, February 1984 explained quite simply how to extract unmodulated video and audio from the CoCo. These signals being suitable for feeding a monitor. The problem then arises that one has to have a monitor (preferably colour) to feed into. Unfortunately colour monitors in Australia are not cheap, at around \$500 or so, but being constantly annoyed by poor colours along with radio and electrical interference it seemed the easiest remedy was to convert the existing T.V. So armed with a circuit and very basic knowledge of T.V. fundamentals I proceeded to do just that. Assuming one has already bought out the two 75 ohm co-ax leads from the CoCo (one video, one audio) the next step is to determine where in the T.V. to feed them.

WARNING!!!

Stay away from Live Chassis T. V's.
They can KILL!! If you don't know
the difference then leave alone!

Firstly I would like to mention that the T.V. I used was a General (not to be confused with a General Electric) Model No. 181 and for once everything fell into place. (SEE Figure 1.)

It seemed the obvious place to start was at the input to the 1st Video amplifier. By desoldering the base leg of this transistor, pulling it out of the P.C.B. to disconnect it from the previous circuitry, I connected the video co-ax from the computer to this transistor's base via a ten microfarad capacitor and wonder of wonders!!! I had a picture on the screen and a pretty good one at that. However it was not as perfect as I felt it could be and decided to investigate further. At this stage I had a slightly washed out picture and no amount of

(Peter Blackman is a member of our Gold Coast group. He is our resident technical advisor and recently made a bit of himself with our Premier Joh, when he hooked up a CoCo to Joh's big toe. The day will long be remembered as the day the Premier (who doesn't normally drink) was (c)loaded!)

adjusting contrast, brightness controls etc. resulted in any considerable improvement. After reading as much on the subject as I could I realised I was missing one important ingredient. Impedance Matching! The impedance matching of a T.V. video amplifier stage is between 75 and 300 ohms and I found that by connecting a 500 ohm potentiometer across the cable feeding into the video amp and adjusting it for the best results I was rewarded with a sharp, crystal clear picture. One last problem was left to solve. This was the very slight lack of contrast. This was overcome very simply by incorporating a Variable Bias supply using the T.V.'s own D.C. supply. This gave excellent control over the gain of the video amp stage and could be adjusted to give a well contrasted picture. Lastly the audio had to be connected and this was done by trying various points around the audio amplifier and with this model T.V. I could not find a point with enough gain so a simple one transistor pre-amp was required, and then audio fed in via the volume control. All in all the improvement to the picture quality was so dramatic that I find it impossible to go back to the original system. The colours are superior (orange is orange, green is green etc). The resolution is sharp and clear and there is not a trace of any electrical or radio interference. Even the sound is improved. To complete the conversion I mounted a change-over switch on the back of the T.V. so I could select either normal T.V. or monitor operation. No doubt I was fortunate that this model T.V. adapted so well without having to invert and/or amplify the video output from the CoCo. Other T.V.'s may be as easy or harder to convert but it is certainly worth the effort.

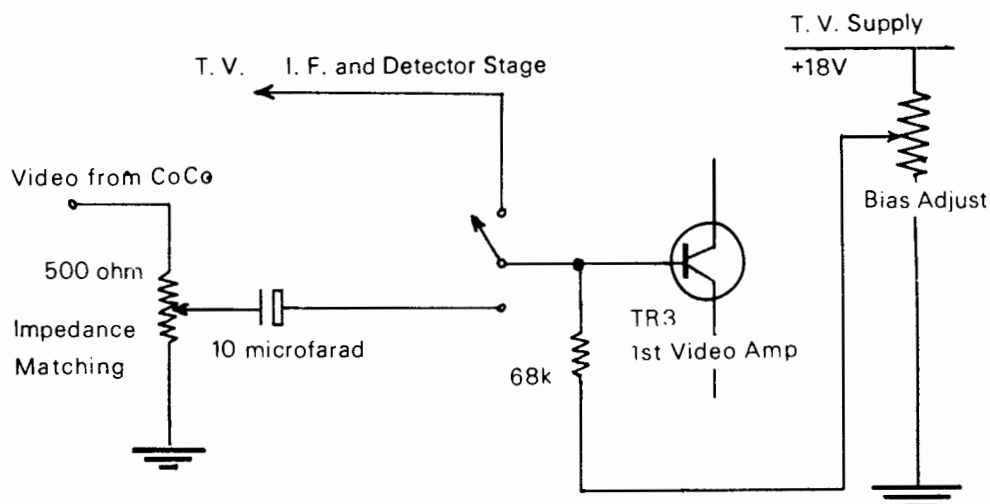


Figure 1.

LITTLE GEM

Kevin Michewski

Another program we have included in CoCoOz this month is this beaut new "LITTLE g", (Little Gem?) So if you've ever wanted more from your editor (Be nice to him G.M.) then this is the one Tandy didn't provide.

One area where our beloved CoCo often comes in for criticism is its lack of a decent screen orientated editor. Recently MiCo readers were treated to a marvelous little editor named (naturally enough) "LITTLE e". Now we are able to give all of you CoCo users our version of the same editor.

I will call this version "LITTLE g" for reasons that may appear obvious later on. I have modified this version from the original which appeared along with MiCo's version in April U.S. Rainbow. Now for a description of the program. First of all CLEAR 200,15616 before CLOADM"LITTLEg". The program resides in protected memory above 15616. EXEC the program and when OK reappears you are ready to start your masterpiece with LITTLE g sitting there waiting for you to call it.

To enter the editor all you need to do is press "shift 0" to get into lower case and type g followed by the number of the line you want to edit and <ENTER>. The program will reset the shift lock back to uppercase automatically. Those of you who have

installed a function key modification can produce a lowercase "g" simply by pressing only function key No. 3. The editor will clear the screen and the line you wish to edit will appear at the top of the screen. You can now control the cursor using the three arrow keys. Right arrow moves the cursor right, left arrow moves the cursor left, down arrow moves the cursor down one line, and <SHIFT> left arrow moves the cursor up one line. Any character in the line including the line number can be changed simply by typing over it. Be wary of changing line numbers. Firstly the old line will remain in the program unchanged, and secondly the new line will overwrite any line with the same number. Typing <SHIFT> @ will insert spaces under the cursor. The <clear> key will delete characters or spaces. All keys repeat if you continue to hold them down.

Once you are happy with the appearance of the line press <ENTER>. The cursor will move to the end of the line. Pressing <ENTER> again will enter the new line as a line in your program. At any time you can exit the edit mode simply by pressing the <BREAK> key. The code for "LITTLE e" is position independent and may be loaded anywhere in memory using an offset. Those of you with a 32K or 64K machine can CLOADM"LITTLEg",16384 after CLEAR 200,32000

Play with this program until you get the

hang of it. Those of you without Extended Color Basic should find "LITTLE g" very useful and if you already have ECB you may find this an easier editor to use than that of ECB.

ORBQUEST

by TONY PARFITT.

Tony Parfitt is one of these obviously very clever 16 year old guys who has a big future ahead of him. We are very pleased to include his work in this issue of Australian CoCo, however the program he herewith describes runs to two or three million pages so ... if you like the sound of it ... buy the tape!



Orbquest, as the name implies, is an adventure. It is a very (very! G.M.) long adventure (only 1K left from 32K when running) and is my first successful attempt at this type of adventure. I think that it will temporarily satisfy the hungry greed for new adventures that most adventure lovers possess (at least I hope so).

I used to be a dedicated arcade game fan and have already put a program called 'RALLY' into CoCoOz (#9). I had tried a couple of times to make adventures but never seemed able to get the right knack. So what does a frustrated adventure author do when sinking into the depths of a programming depression? Why, he turns to the RAINBOW tutorial on adventures by Eric W. Tilenius (bless him) and finds out he's been doing it wrong all the time. But with the help of his article (June, 1984, Rainbow), ORBQUEST is born. (I would have said regurgitated. G.M.). But I digress!

To me, a very important aspect of an adventure game that is often lacking is descriptive language and grammar. Now I'm the first person to admit that I'm far from perfect but I can smell, and I do try to put a little descriptive language in to brighten your day. You know what I really hate? This - > 'I am in a cave'. I would much prefer 'I am in a dark cave. The air is stale and a faint "plip plop" of dripping water can heard in

the background." This is probably one of the main reasons my game takes up 32K and not 16K. Well, I'd prefer to have descriptive language and encourage the 16Kers to go 64K!

The setting and background is outlined in the program. I'm a little (! G.M.) bloodthirsty so be prepared. The adventure runs along the lines of the traditional or classical theme. That is, you are a brave warrior sent out to destroy evil and set your country free. It may be a tired old story, but it's tried and true and a safe!

There are various monsters in the game. There is always one monster that even tracks you down. All of them are aggressive and no quarter is given. They will fight to the death. You can stay and fight (swing, thrust, throw) but you may also run away (chicken!) although the monster may follow you. You weaken each time you fight so try not to belt hell out of everything you meet. There is a warning shown when you can only take on one or two blows more - that's when you run away! The fights are also pretty quick!

The game movement is easy (go cave, go north). Dying (as always) is easy so be very careful. Plenty of traps are there for the unwary - look before you leap! A saving feature is also included.



NOUGHTS AND CROSSES



Graham Morphett

I resurrect this old game because people still have fun playing it and because it illustrates CoCo's ability to adapt fairly plain programming to it's more colorful environment.

The original program can be found in Tandy's book "Games for your TRS 80 Computer". I took this program and adapted it so that it will hopefully run on both CoCo and MiCo.

The logic that is given to the computer is not complete so it is possible to beat the computer, and unfortunately the computer does not get the chance to learn from it's mistakes. May be YOU can rewrite that bit!

Never the less the game gives hours of fun to those inclined to this type of pursuit!

We don't have the printer listing programs quite the way we want them yet, so you may need to approach the list which follows with a little caution.

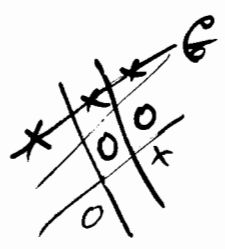
```

10 ****TIC TAC
TOE***11/82***ADJ 25/5/84
20
FORX=1TO9:S(X)=0:NEXTX:P#=CHR$(1
75):CLS:PRINT@11,"TIC TAC TOE"
30
FORT=1TO13:PRINTTAB(12)P#:TAB(20
)P#:NEXTT:P1#=STRING$(24,175):PR
INT@132,P1#:PRINT@324,P1#
40PRINT@71,"1";:PRINT@80,"2";:PR
INT@88,"3";:PRINT@231,"4";:PRINT
@240,"5";:PRINT@248,"6";:PRINT@3
91,"7";:PRINT@400,"8";:PRINT@408
,"9";
60R=RND(2):IFR=1THENC#="0":PRINT
@452,"I'LL GO FIRST THIS
TIME!":SOUND100,10:SOUND150,5:GO
TO100
70 C#=INKEY#:PRINT@452,"DO YOU
WANT 'X' OR
'0'?";:IFC#=""THEN70
80 IF C#="X" THEN 730
90 IFC#<>"0"THEN10
100 Q#="X"
110 G=-1:H=1:IFS(5)<>0THEN130
120 S(5)=-1:GOTO650
130 IF S(5)<>1THEN160
    
```



```

140 IF S(5)<>0THEN200
150 S(1)=-1:GOTO650
160 IF S(2)=1 AND S(1)=0 THEN
600
170 IF S(4)=1 AND S(1)=0 THEN
600
180 IF S(6)=1 AND S(9)=0 THEN
640
190 IF S(8)=1 AND S(9)=0 THEN
640
200 IFG=1THEN 220
210 GOTO270
220 J=3*INT((M-1)/3)+1
230 IF 3*INT((M-1)/3)+1=M THEN
K=1
240 IF 3*INT((M-1)/3)+2=M THEN
K=2
250 IF3*INT((M-1)/3)+3=M THEN
K=3
260 GOTO280
270 FORJ=1TO7STEP3:FOR K=1TO3
280 IF S(J)<>G THEN320
290 IF S(J+2)<>G THEN360
300 IF S(J+1)<>0 THEN390
310 S(J+1)=-1:GOTO650
320 IF S(J)=H THEN390
330 IF S(J+2)<>G THEN390
340 IF S(J+1)<>G THEN390
350 S(J)=-1:GOTO650
360 IF S(J+2)<>0 THEN390
370 IF S(J+1)<>G THEN390
380 S(J+2)=-1:GOTO650
390 IF S(K)<>G THEN430
400 IF S(K+6)<>G THEN470
410 IF S(K+3)<>0 THEN500
420 S(K+3)=-1:GOTO650
430 IFS(K)=H THEN500
440 IF S(K+6)<>G THEN500
450 IF S(K+3)<>G THEN500
460 S(K)=-1:GOTO650
470 IF S(K+6)<>0 THEN500
480 IF S(K+3)<>G THEN500
490 S(K+6)=-1:GOTO650
500 GOTO680
510 IF S(3)=G AND S(7)=0
THEN630
520 IF S(9)=G AND S(1)=0
THEN600
530 IF S(7)=G AND S(3)=0
THEN620
540 IF S(9)=0 AND S(1)=G
THEN640
550 IF G=-1 THEN G=1:H=-
1:GOTO200
560 IF S(9)= 1 AND S(3)=0
THEN610
570 FORI=2TO9:IFS(I)<>0THEN590
580 S(I)=-1:GOTO650
590 NEXTI
600 S(1)=-1:GOTO650
    
```



```

510 IFS(1)-1THEN570
520 S(3)=-1:GOTO650
530 S(7)=-1:GOTO650
540 S(9)=-1
550 PRINT@452," I'LL MOVE
TO... ";SOUND195,4
560 GOSUB820
570 GOTO740
580 IFG=1THEN710
590 IFJ=7ANDK=3THEN710
700 NEXTK,J
710 IF S(5)=G THEN510
720 GOTO550
730 Q#="0"
740 D#=INKEY#:PRINT@452,"
WHERE DO YOU MOVE
";:IFD#=""THEN740ELSEM=VAL(D#):S
OUND75,3
750 IFM=0THENPRINT@452,"
THANKS FOR THE
GAME";:PRINT@488,,:INPUT"RUN
AGAIN
Y/N";SS#:IFS#="Y"THEN10ELSE1280
760 IFM>9THEN780
770 IFS(M)=0THEN790
780 PRINT@452,"THAT SQUARE IS
OCCUPIED";:FORT=1TO8:SOUND230,2:
NEXTT:GOTO740
790 G=1:S(M)=1
800 GOSUB820
810 GOTO 110
820 FORI=1TO9:IFS(I)<>-1THEN850
830 GOSUB1130:IFQ#="0"THEN
GOSUB1110 ELSE GOSUB1120
840 GOTO880
850 IFS(I)<>0THEN870
860 GOTO880
870 GOSUB1130:IFC#="X"THEN
GOSUB1120 ELSE GOSUB1110
880 NEXTI
890 FORI=1TO7STEP3
900 IFS(I)<>S(I+1)THEN950
910 IFS(I)<>S(I+2)THEN950
920 A=I:B=I+1:C=I+2
930 IFS(I)=-1THEN1080
940 IFS(I)=1THEN1070
950NEXTI:FORI=1TO3:IFS(I)<>S(I+3
)THEN1000
960 IFS(I)<>S(I+6)THEN1000
970 A=I:B=I+3:C=I+6
980 IFS(I)=-1THEN1080
990 IFS(I)=1THEN1070
1000NEXTI:FORI=1TO9:IFS(I)=0THEN
1020
1010 NEXTI:GOTO1090
1020 IF S(5)<>G THEN1050
1030 IFS(1)=G AND S(9)=G
THENA=1:B=5:C=9:GOTO1060
1040 IFS(3)=G ANDS(7)=G
THENA=3:B=5:C=7:GOTO1060

```



```

1050 RETURN
1060 IFG=-1THEN1080
1070F#=C#:GOSUB1230:PRINT@452,"Y
OU BEAT ME!! GOOD
GAME";U=U+1:GOTO1100
1080F#=Q#:GOSUB1230:PRINT@452,"I
WIN
AGAIN,TURKEY!!!";V=V+1:GOTO1100
1090 PRINT@452," IT'S A
DRAW!!
";W=W+1:SOUND220,6:SOUND150,10
1100 SS#=INKEY#:PRINT@490,"RUN
AGAIN Y/N";:IFSS#=""THEN1100
ELSEIFSS#="Y"THEN10ELSE1280
1110PRINT@K,P#:P#:P#:PRINT@K+32
,P#:CHR$(143);P#:PRINT@K+64,P#:
P#:P#::RETURN
1120PRINT@K,P#:CHR$(143);P#::PRI
NT@K+32,CHR$(143);P#:CHR$(143)::
PRINT@K+64,P#:CHR$(143);P#::RETU
RN
1130 IFI=1THENK=39
1140 IFI=2THENK=47
1150 IFI=3THENK=55
1160 IFI=4THENK=199
1170 IFI=5THENK=207
1180 IFI=6THENK=215
1190 IFI=7THENK=359
1200 IFI=8THENK=367
1210 IFI=9THENK=375
1220 RETURN
1230FORT=1TO3:P#=CHR$(191):IFA=0
THEN1240 ELSE I=A:A=0:GOTO1260
1240 IFB=0THEN1250ELSE
I=B:B=0:GOTO1260
1250 I=C:C=0
1260 GOSUB1130:IFF#="X"THEN
GOSUB 1120 ELSE GOSUB1110
1270 NEXTT:RETURN
1280 CLS0:PRINT@43,"TIC TAC
TOE";:PRINT@139,"YOU
WON";U;:PRINT@204,"I
WON";V;:PRINT@267,"WE DREW";W;
1290 D#=INKEY#:PRINT@485,"PRESS
ENTER TO
RETURN";:IFD#=""THEN1290ELSERUN

```

Basically, No Address

• Please explain the use of addresses E000-FFFF. Are they used now for anything? Can SAM address these addresses directly or do some chips have to be added for decoding?

Addresses \$D800 to \$FEFF are not used by BASIC. You can use them for anything you like as long as you are in the 64K mode and have BASIC in RAM. As long as you are in Map Type 1, the SAM chip recognizes 64K of RAM. As soon as you hit RESET, though, you go back to 32K of RAM and 32K of ROM, or Map Type 0. Assuming you have already copied BASIC to RAM, all you have to do to re-enter the 64K RAM Map is POKE&HFFDE,0.

TV SCRIBBLER

FOURTH

Graham Morphett

John Redmond

At a recent meeting I was explaining to Mandy and Tracey some of the graphics capabilities of CoCo.

The little program herewith demonstrates what can be achieved with a few powerful CoCo commands!

The program is a drawing slate for the TV. It needs a joystick in the right port to work.

When in operation you press '1' to restart, '0' to switch graphics screens, '5' to go "invisible", '6', '7', & '8' to change colors.

See if you can figure out why we use the 'line' command in line 8150. If you don't know, ask your meet leader.

This type of program and article is the type we want to encourage in Australian CoCo. If you can supply this type of article please do so. People learn from these, in fact you probably used something similar to learn yourself.

```

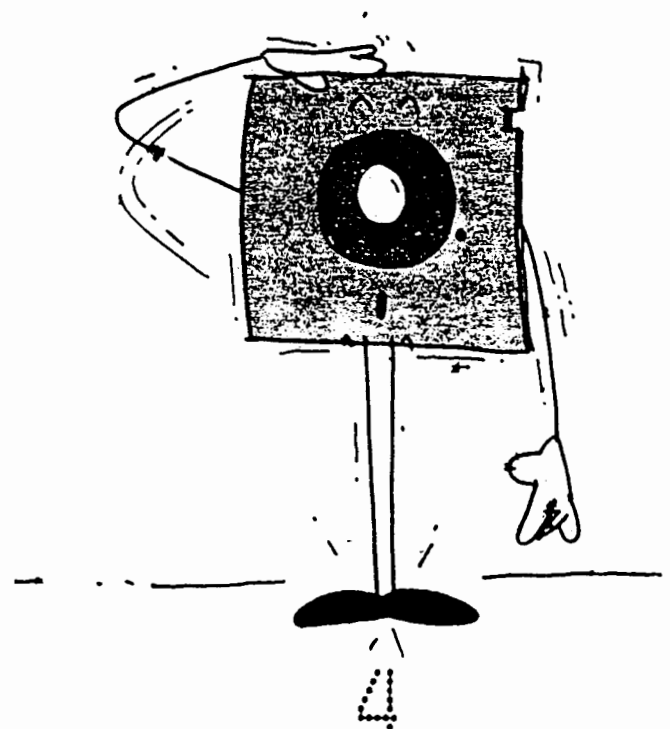
8000 '***DRAW PROGRAM***22/5/84***
8030 PMODE1,1
8040 C=8:X=1:S=0
8050 PCLS
8070 A=4*(JOYSTK(0));B=4*(JOYSTK(1))
8075 IFS=1THENS=0:SCREEN1,S:GOTO8080
8077 S=1:SCREEN1,S
8079 A=4*(JOYSTK(0));B=4*(JOYSTK(1))
8080 D#=INKEY#:IFD#=CHR$(32)THEN8030
8090 COLORC,5
8095 IFD#="1"THENRUN
8100 IFD#="5"THENC=5
8110 IFD#="8"THENC=8
8120 IFD#="7"THENC=7
8130 IFD#="6"THENC=6
8135 IFD#="0"THEN8075
8140 IFX=1THENPSET(A,B,C):X=2:GOTO8160
8150 LINE(E,F)-(A,B),PSET
8160 E=A:F=B
8170 GOTO8079
    
```



A number of you have been asking about alternative languages for your CoCo. The very nimble mind of John Redmond from Macquarie University has produced for us a FORTH compiler. I am very sorry to say that I am unable to give you a good overview of the program but we are including it on CoCo0z for you to try. CLOADM"FORTHOG" and EXEC. You are now ready to get into your new language. The first command to use is "HELP" (I didn't have too much trouble guessing that one K.M.). This will give you a screen full of instructions and the list of COMMAND WORDS used in the language. John recommends the text "STARTING FORTH" as your source of study to learn the language.

FORTH is a powerful language and I think we are all very privileged to have had John put such an effort into giving an added direction in which to apply our CoCo.

WITH DISK . . .



YOU'RE THE BOSS

PIPELINE

HOW ABOUT 128K? Yes, a line of 128K memory expanders has been introduced for the Color Computer by Dynamic Electronics Inc. These expanders mount inside the computer and are compatible with all existing software. The memories consist of two 64K memory banks which can be selected by either a miniature three-position switch or by software. Since each bank is totally independent, separate programs can be loaded and run in either bank. When banks are switched, the unselected bank is placed into the power-down mode with all variables and vectors being preserved. Control can be passed from one bank to the other by poking two values into a memory location.

The expanders consist of a control circuit mounted in modules that plug into a PIA socket and the SAM socket, two banks of 64K RAM, and a three position toggle switch for either hardware or software selection of the banks. Three models are available: ME-128D for upgrading "D" and "E" boards (\$269), ME-128F for upgrading "285" boards (\$259), and ME-128-64 for upgrading all 64K computers to 128K (\$199).

For more information, contact Dynamic Electronics Inc., P.O. Box 896, Hartselle, AL 35640; (205) 773-2758.

IF YOU DON'T already know, Radio Shack's *Microcomputer News* will cease to exist after its July issue. The "news-letter" which Radio Shack started in support of the Model I and expanded into a well-done piece will not longer be

available. Editor Bruce Elliott, who did an excellent job with *Microcomputer News*, has been reassigned to other areas. How will Radio Shack honor its subscriptions to *Microcomputer News* -- by offering readers an opportunity to receive subscriptions to eight other computer magazines for the duration of the subscription term. Those magazines, of course, include the RAINBOW and PCM -- our sister publication which covers both the Model 100 Portable Computer and the Tandy TRS-80 2000.

A NEW SERVICE is being offered by Newssoft -- a news service -- and they use the Color Computer exclusively throughout the operation. Newssoft News Service (NNS) is designed specifically to bring daily news and information to local bulletin board systems. It operates much like any wire service and is available to BBSs on a subscription basis for rates ranging from \$8.50 a month for a BBS with "network status" to \$24.95 for a one time, one month subscription.

Some of the regular columns being offered are a daily computer news column, a hardcore hackers' technical column, history, trivia, science, and a "women and computers" column.

For more information, contact Newssoft Inc. Computer Services, 2704 NE Everett St., Portland, OR 97232 or place a voice call to (503) 238-0741. Also, NNS has a free sample download available at 300 Baud on (503) 235-5114.

Corrections

MiCo:

D. Allen Curtis tells us that a change should be made to the LITTLE E program in Austin MiCo May 1984. Here are the corrected lines:

MC-10

10 CLS: X=256*PEEK(16976)-13
 30 X=256*PEEK(16976)-12
 40 FORZ=X TOX+267

Graham Pollock has found an error in his program in July 1984 MiCo on page 30. Point 6. should read:

6. TYPE "EXEC 17163" THEN
 ENTER

TAMING CANYON CLIMBER

For those of you that have this game and aren't too successful at it, I have a hint. Personally, I think five lives aren't enough when you reach the more difficult levels, so simply type *EXEC 49222*. The score at the bottom will display 800,600. If the screen is blurred, hit the Reset button. Now you have an almost unlimited amount of lives. Also, for an unusual sight, type *EXEC 49232* to set the cartridge in *PMODE 1*.



SCOREBOARD

For all you enthusiastic Ghost Gobbling, Dragon Slaying, Trigger Happy games players we want to introduce our own all Australian Scoreboard. This will preserve for at least one month your triumph in achieving the current highest score for any game you played on MiCo or CoCo.

We are relying only on your sense of honesty and fair play to determine who has the highest score. What you tell us is what we will print, and please tell us at what difficulty level that score was achieved.

So here we are waiting with the presses hot and ready to print those first unbeatable high scores. Just to whet your appetite here are a few scores to give you an idea of what competition you are up against.

CAIXTO (Mark Data) J Gans Bris 162	MEGABUG (Tandy) G Morphett Sthport 5700	SHENANIGANS (Mark Data) J Gans Bris 112
DONKEY KONG (Tom Mix) J Parkes Bris 31450	PYRAMID (Tandy) J Gans Bris 200	TIME BANDIT (Miehton) J Dougan Bris 35000
KATAPILLAR ATTACK (Tom Mix) G Morphett Sthport 5300	RAAKATU (Tandy) J Gans Bris 40	TUT (Ardvark) J S Gans Bris 53000
LANCER (Spectral) J Gans Bris 84200	SEA QUEST (Mark Data) J Dougan & J Gans Bris 165	

CLUB NEWS



A number of User Groups are already sending their News Letters to me. Thank you very much to these Groups. I am only mentioning three this time because at the time of writing, there were others amongst the mail but then there's a lot of mail and I'm hoping to take only 2 or 3 days to finish this column!

The Perth Users Group Newsletter "CoCoPug" was one of the first to come. They have been discussing modems and Bulletin Boards too! John Christou (Editor of CoCoPug) has been using the HJL Keyboard and loves it. It's dearer, but he says it's worth it.

There is a most useful article about memory maps in CoCoPug for both CoCo and MiCo.

Well done Perth - a valuable contribution to the CoCo/MiCo World. Anyone wanting a copy of the magazine may send \$20 to John Christou, 53 Raymond St. YOKINE, WA, 6060. It's good value!

The OS-9 Users Group's newsletter also came. They have listed 16 programs available, the dearest of which is \$295.00 (Pay Roll), most averaging \$100.00. These programs are definitely business programs and represent very good value for money when compared to

similar programs for other computers.

Kevin Holmes is the contact and his name and address is on page 2.

Geoff Spowart sent 'Valley CoCo-Nuts', the Newsletter of Latrobe Valley Tandy Color Computer Users Group.

It is a very nice job and includes programs and articles on a wide variety of topics including FORTH, Eprom Burners, a numbers teaching program for children, a very handy article on comparative CPUs, and an article on the new computer course at Gippsland Institute of Advanced Education.

Geoff can be contacted on 051-22-1389. This is an excellent newsletter, well worth acquiring.

Generally, the clubs have been fairly active this month and it appears from phone discussions with various club leaders that we can expect a plethora (like that word) of new programs and articles from the clubs next month. Well done clubs, you do a vital job! Did you know that IBM has only just realized the value of support groups and is just now in the States setting out to 'start' user groups. They should have read the Tandy book first and waited for the users to start them!

I look forward to seeing your Newsletters this month.

SUSPENDED CEILINGS

LUXALON CEILINGS

PLASTER WALLS

ALL CARPENTRY



**PROJECT
INTERNAL
LININGS** PTY. LTD.

CARPET LAYING

VINYL LAYING

RUBBISH REMOVAL

PAINTING

FOR THE COMPLETE INTERNAL FITOUT

BRISBANE (07) 208 1044 A/H (07) 290 1496, 208 5131

GOLD COAST (075) 32 4255 A/H (075) 58 1797, 56 1360, 58 1812

The RAINBOW[®]

THE COLOR COMPUTER MONTHLY MAGAZINE

PRINTER UTILITY

16K



Zero The Zero Slasher

By David Bailey

The short program which accompanies this article will make your CoCo slash the zeros when outputting to the printer. The program will work with *any* printer, because the routine is contained entirely *within the computer*. It is coded in machine language and is entirely user transparent — to use it, just load and *EXEC*, and all program listings, program outputs, etc., will have the zeros slashed.

The advantage of having a slashed zero is that you can more easily distinguish it from the letter 'O'. This is especially important in program listings where the variable 'O' is used. Typing an 'O' instead of a '0', or vice versa, can crash an entire program, and is very difficult to debug. Slashed zeros are also useful for spreadsheets and other printouts of computations. The reason that many printers do not have a slashed zero built into their character sets is because the slash is not very formal, and is not desired on reports, documents, or other word processing tasks. If a printer was designed to be used with a word processor to create such text, it probably will not have the slash. For this reason, I have made my program flexible — typing *EXEC* toggles the slash "on" and "off," so a BASIC program can use it only at certain times by having *EXECs* within the program.

To use the utility program, you must type in one of the following programs. If you have Color BASIC, very carefully type in Listing 1 (the BASIC program) and save it. When you want to use the program, *CLOAD*, type *RUN* and when it is done, type *NEW* and you are ready.

If you have Extended BASIC, but do not have an assembler, you also must type in the BASIC program and save it. However, to make it simpler to use, you can *RUN* it, then type:

16K: CSAVEM "SLASH", 16000,16063,16000
32K: CSAVEM "SLASH", 32000,32063,32000

If you have disk, change *CSAVEM* to just *SAVEM*. Now, whenever you want the program, just *CLEAR 200,16000 : (C)LOADM "SLASH" : EXEC*. (If you have 32K, change the *CLEAR* statement to read *CLEAR 200,32000*.) If you had a BASIC program already in memory, it would not be erased by loading "Slash."

If you have an assembler, you can follow the preceding directions, or type in the source code directly. I used *EDTASM+* to create it. If you have this assembler, save the source code by typing "*WSLASH*", then assemble it with *ASLASH/AO/WE*. If you have a different assembler, use the equivalent commands to save source code and object code to tape or disk. Now, to load it, follow the instructions for Extended BASIC after the *CSAVEM* instructions.

Regardless of your system and method of loading, all printouts you make at this point will have slashed zeros. If you want to shut it off, type *EXEC*. It can be re-initialized by another *EXEC*, and so forth as many times as you wish.

The BASIC program in Listing 1 was created translating the machine code produced by Listing 2 into decimal, and making a few other adjustments needed because of the lack of an assembler. Therefore, I will explain the machine language program, Listing 2.

The routine to make the slash is really very short. If you delete the remarks, it shouldn't take you more than 10 minutes to type it in, and I suggest you do so if you have an assembler for the learning experience. First of all, we locate the program in high memory (at 16000 for 16K, or 32000 for 32K). The positions I chose waste some memory above the program, but I wanted the even-starting locations for the ease of loading and saving.

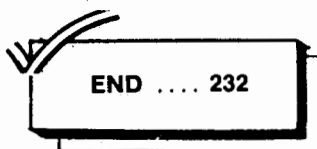
Lines 240 to 380 have nothing to do with the slashed zero — their only function is to allow the slash to be toggled on and off by typing *EXEC*. First the routine checks what is in address 360-361, which is the "hook" for BASIC's printing routine. If the contents have already been changed (so that when printing occurs, it will check with the slash routine first), then execution jumps to *INIT1*, where the toggling effect occurs. To toggle, we check the contents of address 359 (Lines 300-310). If it is a 126, then the diversion we put in addresses 360-361 is working, and we want to shut it off. To do this, we put a 57 in location 359 (Lines 330-340). Conversely, if address 359 contains a 57, then the routine has already been toggled off, and we want to turn it on by putting a 126 in that location (Lines 360-370). If addresses 360-361 have not been altered yet (only when the routine is executed the first time), then (Lines 270-280) it is changed to match the starting address of the slash routine. At the end of

all three of these possible routines, the program branches to *INIT3*, which returns to BASIC.

The real routine starts at line 430. When the slash is toggled on, the BASIC interpreter automatically jumps to this routine before printing *any* character, to *any* device. First, in Lines 430-450, it checks to see if the output device to be used is the printer. This information is contained in location \$6F (a -2 represents the printer, 0 is the screen, etc.). If the device is not the printer, then we branch to *RETURN*, which lets BASIC print whatever character it was going to, and continue on its way. If the device was the printer, then we check the character to be printed (it is held in the A register) in Lines 460-470. If it is not a zero, then we also branch to *RETURN*.

Now, if the device was the printer, and the character was a zero, then the routine must be performed. This happens in Lines 510-540. First of all, understand how the slashed zero is constructed: a slash is printed (the character next to the right shift key), the printer backspaces-one, then prints the regular zero. Line 510 loads the A register with the slash (remember the A register holds the character to be printed) then jumps to the ROM subroutine to print a character (the address of this routine is held in another address, \$A002 —this is called "indirect addressing"). We then repeat that procedure, only the character we load A with is going to be the backspace — the #S08 in Line 530. The printer backspaces, then flows to the *RETURN* routine. There, Line 610 automatically returns the zero into the A register, and this zero will be printed over the slash when we tell BASIC to continue on its way in Line 620 with an *RTS*.

This program was written with flexibility in mind. You can create any other character you like if it is formed by overlapping two already existing characters. Just put the character you want to change after the apostrophe in Line 460, then put the character you want to overlap it with after the apostrophe in Line 510. For example, to change the minus sign into the standard division symbol (the bar with a dot above and beneath it) you could put the dash (minus) character in Line 460, and put the colon in Line 510. Please note that when you do this, all minus signs will be printed as division signs when the routine is toggled on. Since you cannot change the toggle in the middle of a *LLIST*, for example, you would not want to list a program that had minuses *and* divisions in it because the minuses would come out like divisions even if you didn't want them to. The routine was originally intended only for redefining characters, and that is the way that it is most useful.



Listing 1:

```

10 *****
20 '          SLASHED ZERO          '
30 '    SLASHES THE ZEROES ON      '
40 'PRINTERS WHICH DO NOT HAVE    '
50 '    THEM BUILT IN              '
60 '
70 ' (C) 1983 BY DAVID BAILEY    '
80 *****
90 '
100 CLS:PRINT"DO YOU HAVE:":PRIN
T"  1) 16K":INPUT"  2) 32K";
    
```

```

A
110 IFA=1THENM=16000 ELSEIFA=2TH
ENM=32000 ELSE100
120 POKE1000,M/1000
130 CLEAR200,M:M=PEEK(1000)*1000
140 FOR X=M TO M+63:READ Y:POKE
X,Y:NEXT X
150 DATA190,1,104,140,125,36,39,
8,142,125,36,191,1,104,32,19,182
,1,103,129
160 DATA126,38,7,134,57,183,1,10
3,32,5,134,126,183,1,103,57,52,1
19,246,0,111
170 DATA193,254,38,16,129,48,38,
12,134,47,173,159,160,2,134,8,17
3,159,160,2,53,119,57
200 S1=M+36:S2=INT(M/256):S3=S1-
(S2*256)
210 POKE M+4,S2:POKE M+5,S3:POKE
M+9,S2:POKE M+10,S3
220 EXEC M
    
```

Listing 2:

```

00130 *LOCATE PROGRAM IN HIGH RAM
00140 *"CLEAR200,16000" OR "CLEAR200,32000"
00150 * BEFORE LOADING.
00160 *****CHOOSE ME OF THE FOLLOWING LINES
00170 *****ACCORDING TO YOUR MEMORY SIZE
3E80 00180 ORB 16000
00190 * ORB 32000
00200 *
00210 *INITIALIZE THE PROGRAM (CHANGE
00220 *BASIC HOOKS TO USE ROUTINE)
00230 *AN "EXEC" TURNS THE SLASH ON AND OFF
3E80 BE 0160 00240 INIT LDX >360
3E83 BC 3EA4 00250 CMPI #START
3E86 27 00 00260 BEQ INIT1
3E8B BE 3EA4 00270 LDX #START
3E8B BF 0160 00280 STX >360
3E8E 20 13 00290 BRA INIT3
3E90 B6 0167 00300 INIT1 LDA >359
3E93 01 7E 00310 CMPA #126
3E95 26 07 00320 BNE INIT2
3E97 B6 39 00330 LDA #57
3E99 B7 0167 00340 STA >359
3E9C 20 05 00350 BRA INIT3
3E9E B6 7E 00360 INIT2 LDA #126
3EA0 B7 0167 00370 STA >359
3EA3 39 00380 INIT3 RTS
00390 *
00400 *MAIN BODY OF PROGRAM
3EA4 34 77 00410 START PSHS A,B,X,Y,CC,U SAVE ALL REG'S
00420 *CHECK FOR A "0" GOING TO PRINTER
3EA6 F6 006F 00430 LDB >96F DEVICE #
3EA9 C1 FE 00440 CMPB #2 PRINTER?
3EAB 26 10 00450 BNE RETURN NO, SO BACK TO BASIC
3EAD 01 30 00460 CMPA #0 IS CHAR A ZERO?
3EAF 26 0C 00470 BNE RETURN NO, SO BACK TO BASIC
00480 *
00490 *THE CHARACTER IS A ZERO GOING TO
00500 *THE PRINTER, SO PERFORM ROUTINE
3EB1 B6 2F 00510 LDA #/ READY FOR '/'
3EB3 AD 9F A002 00520 JSR (#A002) PRINT IT TO PRINTER
3EB7 B6 08 00530 LDA #00 BACKSPACE PRINTER
3EB9 AD 9F A002 00540 JSR (#A002) FOR THE ZERO TO
00550 * OVERLAP THE SLASH
00560 *
00570 *RETURN TO BASIC: 1)RESTORE REGISTERS
00580 * 2)PRINT CHARACTER THAT WAS
00590 * INTENDED, ZERO OR NOT 3) CONTINUE
00600 * EXECUTION OF PROGRAM, LIST, ETC.
3EBD 35 77 00610 RETURN PULS A,B,X,Y,CC,U GET REG'S BACK
3EBF 39 00620 RTS PRINT CHAR & CONT.
3E80 00630 END INIT
    
```

TUTORIAL

THE COLOR COMPUTER

INTERFACE YOUR OWN CIRCUITS

By T. Whit Athey

While the majority of Color Computer owners are probably making their peace with at least some aspects of programming, not too many are all that comfortable with the guts of the gadget — the hardware, the digital circuitry. However, for anyone who has been secretly wishing that he/she knew a lot more about digital circuits and the operation of the Color Computer, I want to convince you that now is the time to learn. While it isn't exactly easy to understand digital circuits, it isn't any more difficult than programming, and in fact, is very similar to programming in many ways. Besides, it's great fun.

In this article I would like to entice you into building an I/O board which can interface between the Color Computer and your own projects. By taking the plunge and "getting your feet wet" with a real hardware project, you can learn much more than by just reading about it. Also, this is a very practical way to begin because the project is straightforward and leads naturally to further work on your own. I will also discuss some of the possible applications of the board.

I should begin by giving a large measure of credit for my interest in circuits to William Barden. His article, "A General-Purpose I/O Board for the Color Computer," appeared in the June 1982 issue of *Byte Magazine*. He has an excellent discussion on the way the Color Computer does I/O, both internally (to and from memory) and externally (to and from peripherals), and I would recommend that you look it up. The only problem is that Mr. Barden's design for an I/O board doesn't work on all Color Computers.

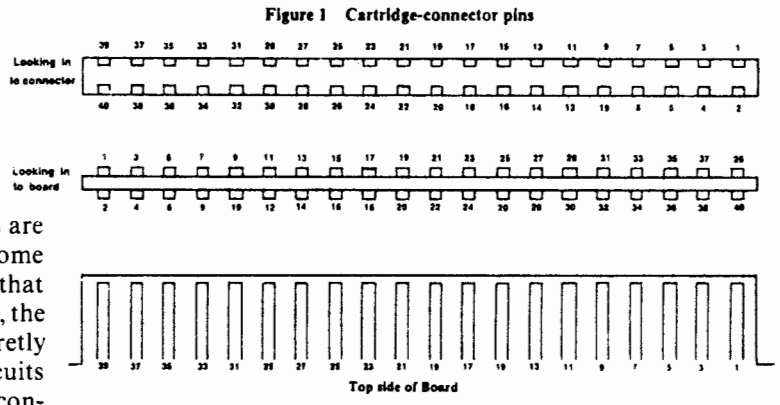
The Cartridge Connector

First of all, I am sure that everyone knows that the Color Computer has a slot on the right side where the game cartridges plug in. If you have a game cartridge lying around, turn it over and slide back the spring-loaded cover from the business end of it. Inside you can see the end of a printed circuit (PC) board and an edge connector with 40 pins (20 on top, 20 on bottom). So, there's nothing more inside that little black box than a PC board with assorted components (components not visible without taking the cartridge completely apart).

Figure 1 shows the computer's cartridge connector and the mating PC board connector. Those 40 lines give us access to nearly every signal of importance which is generated inside the Color Computer. Anyone who has a little soldering experience can put together his/her own PC board (with or without a cover) which plugs into the cartridge slot and interacts with the computer. The board design that I will discuss can provide the first stage, the interface, for your own designs, or for some examples I will present.

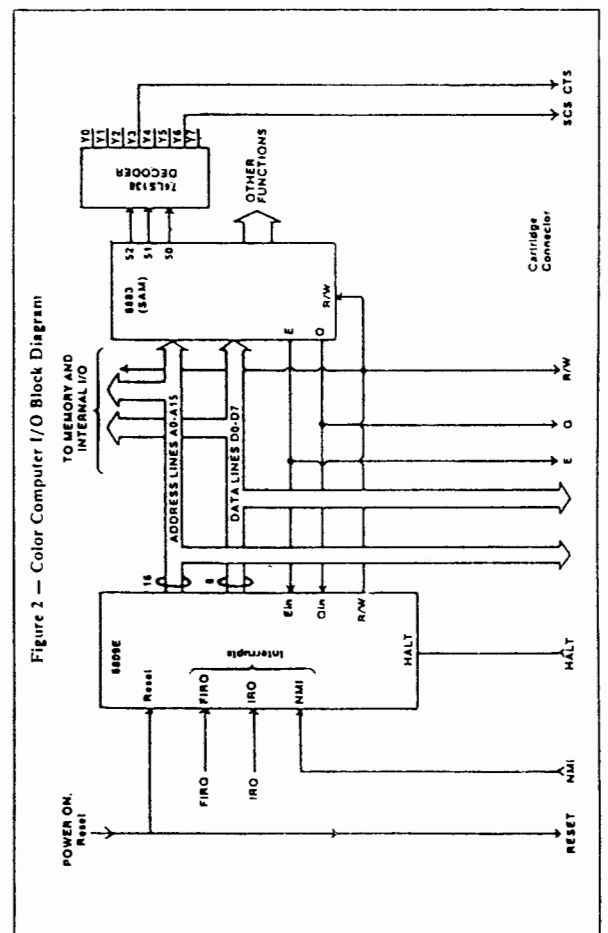
The Color Computer's I/O Structure

Figure 2 shows the block diagram of the Color Computer I/O and the lines which come out to the cartridge slot. Table 1 lists these lines with their names and functions. Many of the lines are connected directly to the heart of the computer,



the Motorola 6809E microprocessor. Also of fundamental importance is the Motorola 6883 synchronous address multiplexer (SAM). In fact, the Color Computer is made up almost entirely of Motorola integrated circuits ("chips").

The 6809E is the real brains of the outfit, controlling the whole operation, but it farms out many important tasks to other large scale chips like the SAM. The 6809E is mostly an 8-bit microprocessor, but with some 16-bit capability, and it is probably the most powerful 8-bit microprocessor around. There are 16 address lines designated A0 (least significant bit) to A15 (most significant bit) which allow unique addressing of up to $2^{16} = 65536$ ("64K") memory locations. The address lines are used whenever the 6809E fetches a byte (8 bits) of data or an operation code from memory, or writes



a byte to memory or to other internal devices. The data is transmitted over eight data lines designated D0-D7.

The SAM chip handles several routine functions for the 6809E. It provides two clock signals (just an oscillating square-wave signal), called E and Q to the microprocessor to permit all operations to have the proper timing. The SAM also controls and decodes the memory mapping of the system. The computer must know not only the exact address in an operation, but also what *area* of memory is being addressed. Since some memory areas are dedicated to specific tasks, the SAM feeds three signals to a 74LS134 decoder chip which, in turn, provides an output which depends on the area of memory being addressed. Only one of the eight output lines of the 74LS134 are active (low, or zero voltage) at any one time. When addresses in the range 0-7FFF are being addressed, Y0 will be active, indicating RAM (random access memory) addresses. Y1 and Y2 indicate that the ROM (read only memory) areas at 8000-9FFF or A000-BFFF are being addressed, and Y3 points to cartridge ROM at C000-DFFF. When Y4 is active the PIA (peripheral interface adaptor) addresses at FF00-FF1F are being addressed, and Y5 similarly selects the second PIA at FF20-FF3F (actually each PIA uses only four addresses in these ranges).

If Y6 is low, locations FF40-FF5F are being addressed. There is nothing in the Color Computer at these addresses, but Y6 could be used to select a third PIA, for example. Or, since Y6 is available at the cartridge slot (as the line labeled SCS), it can select a device plugged into the cartridge slot. We will make use of that fact in the interface circuit to be outlined here.

Note that when the microprocessor calls for a memory location, it can only put out the address on the address lines (to which the SAM/74LS134 adds the map signal, Yn) and "listen" for a response. It does not "know" what device is actually responding. It is only important that the device recognize that it is being addressed, and become active only when it is being addressed.

A more detailed discussion of the workings of the Color Computer is given in the *Color Computer Technical Reference Manual*,² available at Radio Shack.

I/O Operations

Input/output operations in the Color Computer are said to be "memory mapped," which means that the microprocessor is tricked into thinking that I/O devices or peripheral controllers are just another part of memory. All that is required to carry out I/O operations is the execution of an instruction like LDA (6809 operation to load the A register) or STA (store contents of A register) to the address of the device. This can even be done from BASIC with *PEEK* or *POKE* commands.

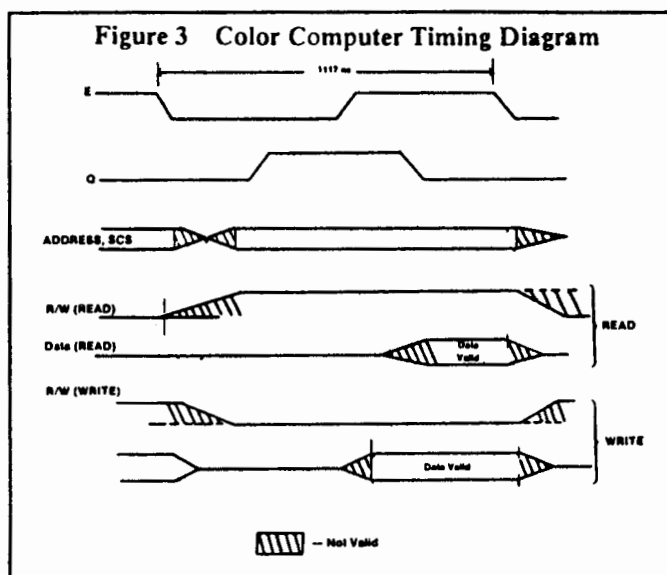
In the Color Computer, locations FF00-FF3F are used for I/O through the peripheral interface adaptors (PIAs). For example, FF00-FF03 are used to read the keyboard and joysticks through PIA U8. Locations FF20-FF23 are used for controlling several functions through PIA U4, including cassette I/O, serial I/O, and graphics modes. One must know the proper byte to write to these locations in order to obtain the desired effect, but the bottom line is that the byte can get to the proper place from a simple STA or *POKE*.

The PIA can determine when it's being addressed by the state of the memory map signals, Y0-Y6, which were discussed above. Recall that only one of these is active (low) at any time, and that addresses in the range FF00-FF1F result

Pin No.	Signal Name	Description
1	-12 V	-12 volts (100 mA)
2	+12 V	+12 volts (300 mA)
3	HALT	Halt input to the CPU
4	NMI	Non-maskable interrupt
5	RESET	Reset and power-up signal
6	E	Main CPU clock signal
7	Q	Clock Signal which leads E
8	CART	Interrupt for cartridge detect
9	+5 V	+5 volts (300 mA)
10	D0	CPU bit 0
11	D1	CPU bit 1
12	D2	CPU bit 2
13	D3	CPU bit 3
14	D4	CPU bit 4
15	D5	CPU bit 5
16	D6	CPU bit 6
17	D7	CPU bit 7
18	R/W	Read/write signal from CPU
19	A0	CPU Address bit 0
20	A1	CPU Address bit 1
21	A2	CPU Address bit 2
22	A3	CPU Address bit 3
23	A4	CPU Address bit 4
24	A5	CPU Address bit 5
25	A6	CPU Address bit 6
26	A7	CPU Address bit 7
27	A8	CPU Address bit 8
28	A9	CPU Address bit 9
29	A10	CPU Address bit 10
30	A11	CPU Address bit 11
31	A12	CPU Address bit 12
32	CTS	Cartridge select signal
33	GND	Ground
34	GND	Ground
35	SND	Sound input
36	SCS	Spare select signal
37	A13	CPU Address bit 13
38	A14	CPU Address bit 14
39	A15	CPU Address bit 15
40	SLNB	Disable device selection

in Y4 going low. Thus Y4 can be used as a "chip select" signal for the PIA U8, and similarly for Y5 for PIA U4. By using only two of the 16 address lines, namely A0 and A1, along with the chip select signal, the PIA U8 can distinguish its four addresses, FF00-FF03, and will only respond to addresses in this range. The fact that Y4 is low means that the 16 address lines carry the values 1111 1111 0000 00-- (FF0- in Hex), and only the last two lines, A0 and A1, need to be checked, and that is all that the PIA does check.

Figure 3 shows the timing for the read and write cycle of the 6809E. For example the LDA read cycle begins with the clock signal E going low. Within 100-200 ns (1 ns = 10⁻⁹ seconds) the R/W line has gone high (indicating read) and



the address lines and Y0-Y6 have assumed their appropriate values. After E returns high the data lines will contain the byte being read and the 6809E "strokes" in the data.

The write cycle, for example during the execution of a STA instruction, proceeds in a similar fashion. In this case the R/W signal goes low to indicate a write. The data from the 6809E is put out on the data bus as E goes high and remains valid until the end of the E cycle. During this "data valid" period it may be "picked off" or "strobed in" by another device.

The I/O Interface Board

An interface board could be designed around another Motorola PIA chip which would insure compatibility with the rest of the Motorola system. However, the PIA is rather cumbersome to control (program), and it has only two 8-bit I/O ports. On the other hand, the analogous chip made by Intel, the 8255A PPI (programmable peripheral interface) chip is very easy to control, has three I/O ports and has more than enough flexibility for most applications. The only potential problem is that the Color Computer timing signals don't quite meet the specifications for the 8255A.

The Intel 8255A is a 40-pin large-scale integrated-circuit (LSI) chip. It has four 8-bit registers, three of which are bidirectional I/O ports, designated A, B, and C and the fourth is a control register which is used to set the operating mode of the chip's three ports under program control.

There are three modes under which the 8255A can be operated. The simplest mode, and the mode which will be discussed here, is mode 0, basic input and output. Mode 1 is for strobed input and strobed output, and mode 2 is for strobed bidirectional I/O. Modes 1 and 2 use lines from the C port as control lines for the other two I/O ports. With these last two modes you can get about as fancy as you like, but here we will concentrate on the mode 0 I/O for which programming and interfacing is very easy. Later, after building your interface and gaining experience with it, you can always use modes 1 and 2 with only software changes. These modes are discussed in detail in Paul Goldsbrough's book in the Blacksburg Continuing Education Series, *Microcomputer Interfacing with the 8255A PPI Chip*³.

Under any of the modes the chip functions can be configured under program control by *POKEing* the proper byte into the control register (location FF43 in this design). Ports A, B and C can be either input or output ports, or any combination thereof. Port C can even be split into two 4-bit ports so that four lines are for input and four are for output. Table 2 shows the values for control words which select the various combinations.

Table 2 Control Words for 8255A Mode 0 Input/Output				
Control Word (hexadecimal)	Port Function (1=input, 0=output)			
	Port A	Port B	C0-C3	C4-C7
80	0	0	0	0
81	0	0	1	0
82	0	1	0	0
83	0	1	1	0
88	0	0	0	1
89	0	0	1	1
8A	0	1	0	1
8B	0	1	1	1
90	1	0	0	0
91	1	0	1	0
92	1	1	0	0
93	1	1	1	0
98	1	0	0	1
99	1	0	1	1
9A	1	1	0	1
9B	1	1	1	1

The three I/O ports each consist of an internal 8-bit data register and eight I/O lines coming out to the pins of the chip. Whenever a port is programmed as an output, the contents of the internal data register will appear continuously on the I/O port pins (5 volts for ones and 0 volts for zeros) until the contents of the register are overwritten.

As an example, I'll show how an alternating pattern of ones and zeros can be written to the A register of the 8255A. The hexadecimal number AA has the bit pattern 10101010. From Table II we can set all the registers for output with the control word 80 (also in Hex). Assuming that we have completed the interface and have it plugged into the computer we would first set the control register with *POKE &HFF43 &H80* and then *POKE &HFF40, &HAA*. Then if we test the pins for port A (more on how to do that later), we should find the alternating pattern we wanted. The control register would only have to be set once at the beginning of a program.

There is only enough current capacity on these output pins to drive other integrated circuits. However, by feeding the lines through a line driver/buffer chip, small relays can be controlled. This will be discussed further in the section on applications.

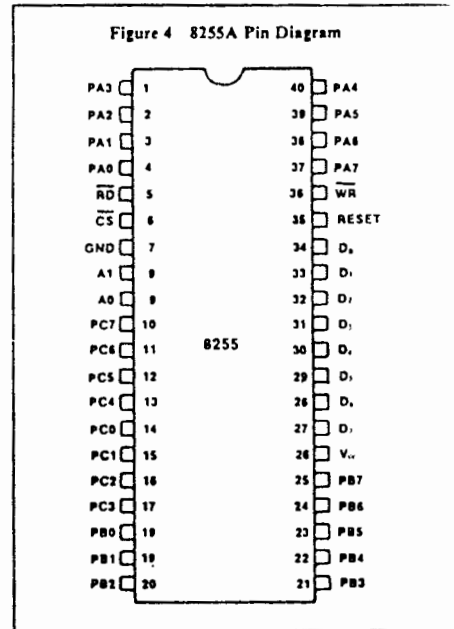


Figure 4 shows the pin diagram of the 8255A. Most of the pins are I/O lines and have been discussed already. The function of the others is listed below:

- CS (Chip Select)** A low on this input pin enables the chip. When the input is high the chip will not respond to any other signals.
- RD (Read)** A low on this input pin enables the 8255A to put data on the data lines for the microprocessor to read.
- WR (Write)** A low on this input pin enables the microprocessor to write data or a control word to the 8255A.
- A0 and A1 (Address lines)** These input signals control the selection of one of the four registers of the 8255A (00 selects port A, 01 selects port B, 10 selects port C, and 11 selects the control register).
- RESET** A high on this input clears all internal registers.

These input signal requirements are mostly, but not com-

pletely, compatible with the Color Computer signals available at the cartridge connector. The two address lines can be connected directly to the two lowest order bits of the Color Computer address lines. We can use Y6 (SCS) directly for the chip select (CS) input. However, the Color Computer's reset signal is low when active instead of high as required by the 8255A. This signal will have to be inverted. Also, the Color Computer has only one line for both read and write, while the 8255A requires separate signals with both being active when low.

The modification of these latter signals requires a slight detour into the field of logic gates. Logic gates have two inputs and one output. For example, an OR gate will have a high output when either of the inputs is high. The AND gate has a high output only when both of the inputs is high. The NOR and NAND gates are just OR and AND gates with an added inverter on the output (compliment of the OR and AND operations). For example, the NAND gate has a low output when both inputs are high, and has a high output otherwise. A good (and cheap!) reference for logic gates and their applications (and which covers many other common integrated circuits) is the Radio Shack *Engineer's Notebook II*. It is available at under \$3 at Radio Shack.

A logic signal can be inverted by feeding it into both inputs of a NAND gate. The output will be high if the inputs are low, and low if the inputs are high. Nearly all digital circuits have several logic gates, which usually come as four gates on a 14-pin chip, and we will make use of NAND gates on our I/O board.

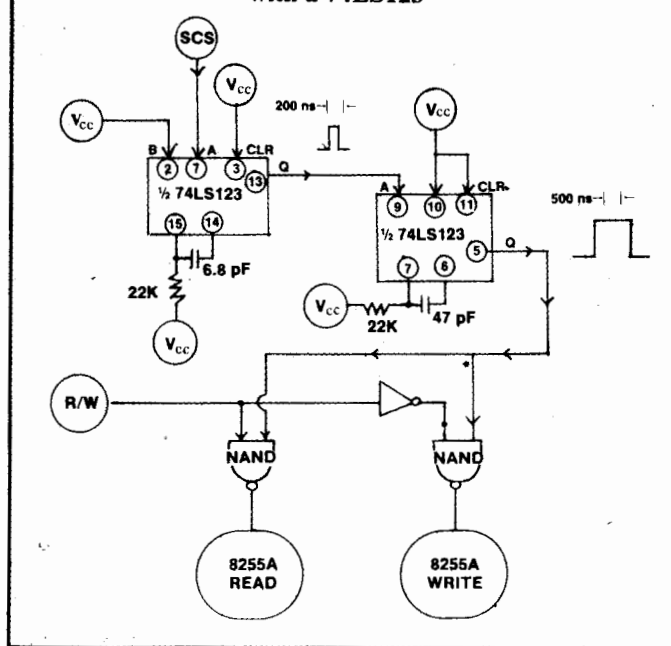
Therefore, the Color Computer's RESET signal will be first fed to both inputs of a NAND gate on a 74LS00 chip, and the gate output will be connected to the 8255A RESET pin. The R/W signal requires a little more work to get acceptable 8255A READ and WRITE signals. On some Color Computers you can use the R/W signal directly for the 8255A WRITE signal (and the inverted R/W signal for READ), but mine wouldn't, and neither would half of those I tested. I recommend that the READ and WRITE signals be generated as described next.

The 8255A READ and WRITE signals must go high again during their operation *before* the CS, A0 or A1 lines change. In fact, the WRITE must return high at least 20 nanoseconds before the lines change. So, what is needed is WRITE pulses and READ pulses which only go low 100-200 nanoseconds after SCS (chip select), and return high 100-200 nanoseconds before SCS does.

The solution is a 74LS123 "one-shot" chip, and a couple more NAND gates (which you already have on the 74LS00 chip). The schematic diagram of this part of the circuit is shown in Figure 5. The 74LS123 is described on Page 52 of the Radio Shack *Engineer's Notebook II*, but note that the pin diagram on Page 52 has the labels for pins 9 and 10 reversed. This chip has two independent sections, each of which allow you to trigger on the state of two inputs, and the pulse length is controlled by the value of an external capacitor. I used the first section to trigger a short pulse 200 nanoseconds (ns) when SCS goes low. The trailing edge of this short pulse is then used to trigger the second section of the 74LS123 for a final output pulse of about 500 ns.

This resulting pulse is shaped and timed perfectly relative to the SCS (chip select) signal to be a READ or WRITE pulse. Note also that we only need this special READ/WRITE signal on our board when, in fact, the chip is selected. Therefore, we can use the R/W and inverted R/W signals to gate on (with a NAND gate) this new specially designed pulse to produce perfect READ and WRITE

Figure 5 Creation of READ and WRITE signals with a 74LS123



pulses, just when we need them.

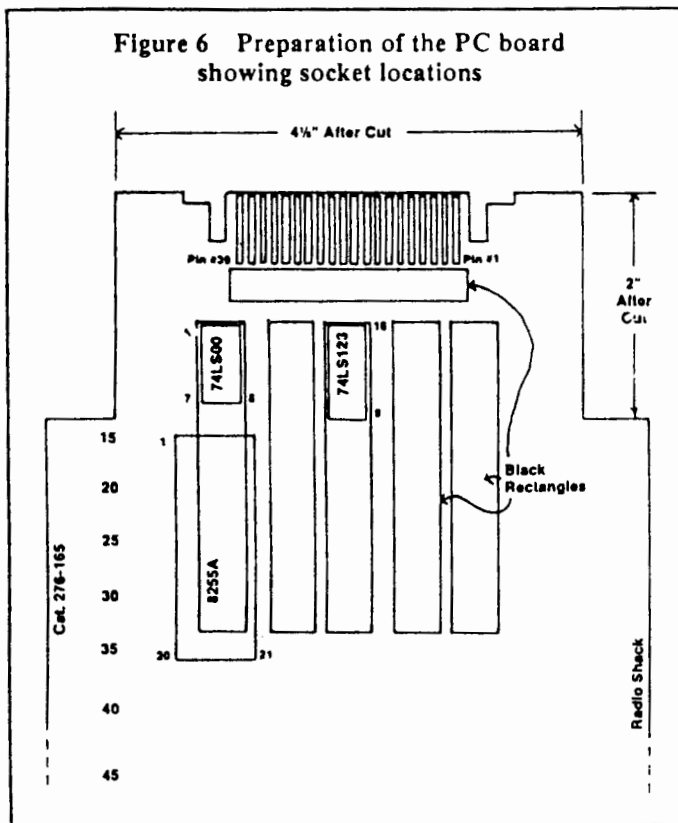
Since the two address lines of the 8255A are connected to A0 and A1 of the Color Computer address bus, and the chip select is connected to SCS, we can use the addresses FF40-FF43 for the four registers of the 8255A. These locations can be treated just as any other memory locations. Note that the four registers do not have unique addresses since FF50-FF53 (or even FF44-FF47 — only the FF and the last two bits matter) will also address the registers. With further address decoding (using address lines besides A0 and A1) you could even add more PPIs to the board, each separately addressable.

Building The Interface

Assuming that I have you sufficiently hooked on the idea, the next step is to build the I/O board. I must confess that I had a little help in building the board — my 12-year-old daughter did most of the work.

I am aware of no widely available, reasonably priced PC board which is specifically designed for the Color Computer, but there are several which will work with a little modification. The main requirement is that the board have an edge connector with at least 40 pins (20 on each side) with 0.1 inch spacing. Radio Shack sells a board, catalog number 276-165 which is my first choice. It is large enough to accommodate future additions, already has the right number of pins, and has edge connectors at both ends (the second one might be useful to connect a cable for a future application). It also costs less than the others I considered. Radio Shack also sells a board about half the size of the recommended one. It has plenty of room for the I/O interface, but not much room for anything else that you might want to add later. You will be better off with the flexibility of the larger board.

Whatever board you use, it must be cut down to fit the cartridge slot. Figure 6 shows the finished dimensions for the part which is plugged into the cartridge slot (the rest of the board can be any size). If your board has more than 20 pins on each side of its connector, the others must be cut off in getting down to the required dimensions. Keep the middle



20 pins and cut down the center of the pins on either side of the middle 20. After making these cuts, strip off the remaining half of the pin conductor material of the pins which were cut (leaving intact the board underneath). This narrow board area left at the edges of the connector will serve to guide the pins to their proper mating pins in the cartridge slot.

If you are using the Radio Shack board, the connector is okay as it is. However, the edges of the board near the connector must be trimmed because it is too wide for the cartridge slot. Draw a line along the outermost row of holes on each edge of the board and extend it to the end of the board (the end with the low-numbered rows). Cut along the line up to the fourteenth row of holes using a nibbling tool, jigsaw, or small hacksaw. Repeat on the other side.

The board and other parts you will need are shown in Table 3, along with some suggestions for sources. Where more than one source is listed, my personal preference is listed first. In case you can't find some of the parts locally, I have arranged for HIB Associates, 3505 Hutch Place, Chevy Chase, Md., 20815, to handle mail orders at the prices listed in the table. Or, if you prefer, you can get all the parts listed from HIB for \$27 (include \$2 for postage and handling on all orders for parts).

Now take a good look at the board. The side with the copper pads is the wiring side and will be the bottom side as the board is inserted into the cartridge slot. The side with the black rectangles is the component (top) side where the chips will be mounted. Note that the two halves of the board are not exactly alike, and that the instructions here assume that the end with the low-numbered rows is used for the interface.

Place a 14-pin socket on the component side of the board with the pins sticking through the holes on rows 6-12 as shown in Figure 6. This socket will fit exactly on the black lines at the end of the black rectangle. Make sure that the pins are inside the rectangle, and then bend the four corner

pins outward and over onto the copper pad to keep the socket from falling off. Looking at the board from the bottom, there should be a two-hole-wide copper strip running down between the pins, but not touching the pins. Now solder the four bent pins to their copper pads. By first pressing the bent pins flat on the pads with the soldering iron for 2-3 seconds, soldering will be facilitated.

On the same black rectangle mount the 40-pin socket, leaving two rows of holes separating the two sockets. The 40-pin socket is wider, but its pins should come through onto the same kind of copper pads as those of the first socket. Again, bend and solder the corner pins, plus two more around the middle of the socket. This socket is placed here rather than closer to the connector so that all of its pins will be accessible for testing without removing the computer cover.

You will need a 16-pin socket for the 74LS123. Mount this socket on the middle black rectangle between rows 6 and 13, using the same procedure as for the other sockets (there will be one black rectangle between the two we are using). For this socket it will be more convenient to make many of the connections to the pads rather than the pins. Therefore, bend and solder to the pads the pins 2, 3, 5, 6, 8, 10, 11, 14, 15 and 16 (the pins do not contact the pads unless they are soldered). Then any wires to be connected to these pins (most of the connections will be jumpers from one of the buses) can be made to the corresponding pads.

Before starting to run the wires, I found it helpful to label the connector pins with numbers with a felt-tip pen for at least pins 1 and 39 on the top (component) side of the board, and 2 and 40 on the bottom (wiring) side of the board. Refer to Figure 1 to make sure you label the pins properly. When wires must be soldered to the top connector pins, the wire should pass through the holes in the small rectangle in front of the pins and be bent over to contact the proper connector pin. Then the wires can be soldered to the pins.

It is also helpful to label the four corners of each socket with the corresponding pin numbers. For each IC socket, pin 1 should be at the upper left corner when looking at the component side of the board as shown in Figure 6. Turn the board over and label the socket pins (pin 1 will now be at the

**Table 3
Parts List and Approximate Prices**

Item	Sources
1. PC Board, 276-165	Radio Shack (\$10)
2. 40-pin socket	Heathkit, Radio Shack (\$1)
3. 16-pin socket	Heathkit, Radio Shack (50¢)
4. 14-pin socket	Heathkit, Radio Shack (50¢)
5. Intel 8255A	Heathkit (\$11)
6. 74LS00	Heathkit, Radio Shack (\$1)
7. 74LS123	Heathkit (\$2)
8. Capacitors, .1µF (3)	Radio Shack (75¢)
9. Capacitor, 47 pF	Heathkit, Radio Shack (15¢)
10. Capacitor, 6.8 pF	Heathkit (15¢)
11. Resistors (2), 22K ohm	Radio Shack (20¢)
12. Resistor, 330 ohm	Radio Shack (30¢)
13. LED	Radio Shack, Heathkit (50¢)

Items 1-13 above available from HIB (see text). Prices listed are approximate for Radio Shack and Heathkit. (Heathkit parts are not listed in their catalog, but are carried by Heathkit Electronic Centers in some major cities.)

upper right) by writing the numbers on the board next to the pins.

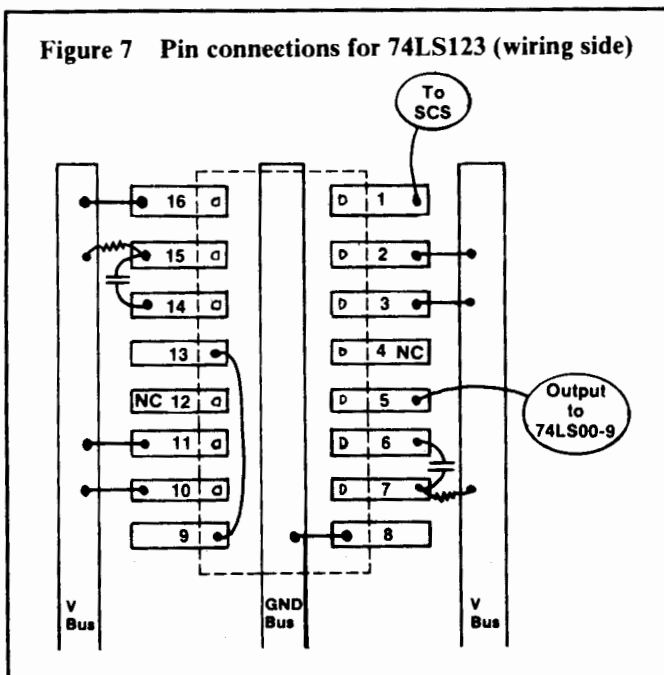
The two-hole-wide copper strips which run along each side of the sockets on the wiring side can be used to supply +5 volts (V_{cc}). These strips will be referred to as the V_{cc} "bus." Similarly, the strips which run directly under the sockets can be connected to ground and will be called the ground bus.

Follow the wiring list in Table 4. Use #30 wire for all logic signals (address and data lines, control signals, etc.) and regular (single-conductor) hookup wire for the power and ground connections. Do not insert the IC chips into the sockets until all wiring has been completed. Because it is easy to make a mistake on the connections on the 16-pin socket, Figure 6 shows these connections from a wiring side view.

The computer should be turned off when inserting or removing a cartridge or PC board from the cartridge slot. *Failure to do so can result in damage to the computer.* Radio Shack has built in a measure of protection for their cartridges by trimming about a millimeter off the leading edge of pin 9 (+5 volt pin) of their cartridge connectors. If a Radio Shack cartridge is accidentally removed or inserted with the power on, it probably won't be fatal (but don't press your luck). We can give ourselves that same measure of protection by trimming back pin 9 on our I/O board. Use a sharp knife or razor to cut through the metal strip about one mm back from the ends of the other pins. Then peel the cut-off strip from the board, leaving pin 9 a little shorter than the others. Since it is very easy to have a board come out of the cartridge slot by accident when you are testing or using it, be sure to give yourself this little safety factor.

After completing the wiring (and before the ICs are inserted) use a continuity checker if you have one to test all connections. If you don't have one, carefully examine all solder points, and then check the board against the wiring list one more time. Check especially the wiring of the 74LS123 against Figure 7, and for any stray bits of solder between the pads. Warning: You can destroy the microprocessor and SAM chips with improper connections.

When you have satisfied yourself that all is well, insert the chips into the sockets. If you are doing this for the first time,



**Table 4
Wiring List**

From	To	Signal
CC-9	V_{cc} Bus	+5 V (V_{cc})
CC-33	GND Bus	Ground
74LS00-7	GND Bus	Ground
8255A-7	GND Bus	Ground
74LS123-8	GND Bus	Ground
74LS00-14	V_{cc} Bus	V_{cc}
8255A-26	V_{cc} Bus	V_{cc}
74LS123-2	V_{cc} Bus	V_{cc}
74LS123-3	V_{cc} Bus	V_{cc}
74LS123-10	V_{cc} Bus	V_{cc}
74LS123-11	V_{cc} Bus	V_{cc}
74LS123-16	V_{cc} Bus	V_{cc}

22K resistor between V_{cc} and 74LS123-7

22K resistor between V_{cc} and 74LS123-15

6.7 pF capacitor between 74LS123-14 and 74LS123-15

47 pF capacitor between 74LS123-6 and 74LS123-7

CC-18	74LS00-1	R/W
74LS00-1	74LS00-2	R/W
74LS00-2	74LS00-13	R/W
74LS00-3	74LS00-10	R/W compliment
CC-5	74LS00-4	RESET
74LS00-4	74LS00-5	RESET
74LS00-6	8255A-35	RESET
74LS123-5	74LS00-9	Output of 74LS123
74LS00-9	74LS00-12	Output of 74LS123
74LS00-11	8255A-5	READ
74LS00-8	8255A-36	WRITE
74LS123-9	74LS123-13	200 ns delay pulse
CC-36	74LS123-1	SCS/CS
CC-36	8255A-6	SCS/CS
CC-19	8255A-9	A0
CC-20	8255A-8	A1
CC-17	8255A-27	D7
CC-16	8255A-28	D6
CC-15	8255A-29	D5
CC-14	8255A-30	D4
CC-13	8255A-31	D3
CC-12	8255A-32	D2
CC-11	8255A-33	D1
CC-10	8255A-34	D0

1 μ F capacitors between V_{cc} and ground near each chip's V_{cc} pin.

be extremely careful. You will probably have to bend the chip leads slightly in toward the chip to get them lined up with the socket holes. Once everything is lined up, apply pressure to start the insertion. Once it starts in, stop and check all leads to make sure none are being bent. Press down until the chip is almost seated. You might want to leave it out slightly the first time in case you have to remove it later (use a tiny screwdriver to carefully pry it out if that becomes necessary).

For testing the board you need a logic probe. For about \$1 you can make yourself a perfectly good one: Cut off a pencil-sized piece of small wooden dowel (or just use a pencil) and sharpen it in a pencil sharpener. Solder a 270-350 ohm resistor to the cathode lead of an LED (light emitting diode). Then solder a two-foot piece of flexible (insulated)

wire to the other side of the resistor and attach an alligator clip to the free end of the wire. Or, buy a wire with alligator clips at each end, cut one clip off, and solder the wire to the resistor. Stretch out the leads of the LED/resistor/wire to make one straight line as shown in Figure 8, and tape them

to the dowel with about 1/4-1/2 inch of the anode lead of the LED extending beyond the point. Tape beside the LED, but don't cover it — we have to see it when it lights up.

When the alligator clip is connected to ground, the LED will light up if the probe tip is touched against something at +5 volts. You may want to pass a half-inch piece of stiff wire from the component side of the board through to the ground bus and solder it so you will have a convenient point to connect the alligator clip of the logic probe while testing.

Now we're ready for the big moment! Turn the computer off, insert the board (you may need to support the end of it), and then turn the computer on again. Connect the alligator clip of the probe to ground, and test the probe by touching pin 26 (V_{cc}) of the 8255A (do this from the component side of the board). If you have wired the pin correctly, the LED will light up brightly indicating the presence of +5 volts. Now try the next pin, number 27. This time the probe should light up, but only dimly. Pin 27 is a data line and its state (+5 volts or ground) is changing at almost a million times per second. The LED just indicates an average reading. Test the READ, WRITE, and chip select input pins. They should glow almost as brightly as with V_{cc} since they are normally high. Check pin 35 (RESET) — it should always be low (LED not illuminated) except when you press the Color Computer RESET button (try it). Test all the data lines (pins 27-34) and the address lines (pins 8 and 9) to make sure that the probe gives at least a dim glow. If any of the above tests indicate a problem, turn off the computer, remove the board (while holding up the cartridge slot with your fingers) and check your wiring.

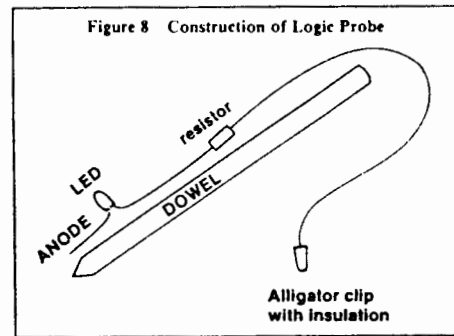
If everything seems to be in order, let's see if we can communicate with the 8255A. Set all ports for output with `POKE &HFF43,&H80`. Test pins 1-4 and 37-40 (the eight lines of port A) and you should now find all zeros (no illumination). Now `POKE &HFF40,&HFF` (Hex number FF is 11111111 in binary) and if it's working, you should now find that all of the port A pins have ones and will light the logic probe. `POKE &HFF40,0` to set port A back to all zeros and test again. You might also try an alternating bit pattern like `&HAA` or `&H55`. Test the other ports in similar manner (refer to the pin diagram to find ports B and C). Port B is at address FF41 and port C is at FF42.

Try writing something to the A register and then reading it back with a `PRINT HEX$(PEEK(&HFF40))`. You won't have to change the control word for this kind of "internal read." Reading from the outside will have to wait until you have something connected to the I/O ports. If you try it now you will just get whatever noise is around.

Applications

I hope that you already have some ideas for some ways to use the I/O board besides lighting up a logic probe. Clearly it can be used to check on the status of switches (the switches might be the detectors of a security system, for example) or to control read relays (these applications are discussed in the article by William Barden). However, one of the objects of this article was to get you interested in digital circuits, so in the next article I will give an example of how the I/O interface can be used to control another large scale integrated circuit. The chip I will use is a General Instruments Sound Generator Chip (AY-3-8910). It has three independ-

ent tone channels, a noise channel, envelope control and even two more I/O ports (so you don't really "lose" an 8255A I/O port by connecting it to the AY-3-8910).



References

- ¹Barden, William. "A General-Purpose I/O Board for the Color Computer." *Byte*, June 1982, p. 261.
- ²*Color Computer Technical Reference Manual*, Radio Shack Cat. No. 26-3193.
- ³Goldsbrough, Paul F. *Microcomputer Interfacing With The 8255 PPI Chip*. Howard W. Sams & Co., Indianapolis, Indiana, 1979.
- ⁴Mims, Forest M. *Engineer's Notebook II: A Handbook Of Integrated Circuit Applications*. Radio Shack Cat. No. 276-5002, 1982.

Program Quickie . . .

Finding Those Bad Sectors

By Paul Gani

I have seen dozens of programs to find bad sectors and then isolate them from BASIC. Yet, all use `DSKIS` and thus, you always get I/O Errors and have to manually continue the program to find other bad sectors. Below is a short program to find all bad sectors with no interruptions. Just enter it and type `RUN`. The program will look for bad sectors (if any) and if it finds one, the program will say so and then continue. Then you can use one of the dozens of programs already published to isolate that area.

The listing:

```

10 DEFUSR0=PEEK(&HC004)*256+PEEK
   (&HC005)
20 FOR T=0 TO 34:FOR S=1 TO 18
30 POKE 234,2: SET TO READ
40 POKE 235,0: DRIVE NUMBER
50 POKE 236,T: TRACK
60 POKE 237,S: SECTOR
70 POKE 238,14: DUMP TO THE
80 POKE 239,00: GRAPHICS AREA
90 Y=USR0(0): P=PEEK(240)
100 IF P<>0 THEN PRINT "ERROR IN
   TRACK";T;"- SECTOR";S
110 NEXT S:NEXT T

```



Computer Simulations For Fun And Profit

By Robert K. Tyson, Ph.D.

Prophecy by computer is an art form and it is a science. Computer Simulations are used for examining events which can or will be duplicated in the real world. So far, computers have been used to simulate traffic patterns, human population changes, molecular chemistry, the weather, and countless other things. Computers have even been used to simulate other computers to determine data rates, I/O throughput, computational speed, and debugging techniques. Since many phenomena that we can observe are governed by a mathematical formula, a computer Simulation can be used to expand our window into the world. When random occurrences determine the outcome of a series of events, a computer Simulation is particularly useful since it can simulate literally thousands or millions of events. An investigator can then determine probable outcomes.

Of course, many situations that we wish to simulate are not determined by formulae or probability but are controlled by a logical human thought process. These "heuristic" Simulations are often the most useful and the most fun. For instance *Strategy Football* (THE RAINBOW, August 1983) is a heuristic Simulation with formulae and random occurrences taking a back seat. The NASA computer Simulations which determine the best time to launch, to fire boosters, etc. are almost entirely formula driven with little or no human tampering. A Simulation of roulette (Gerry Schechter, THE RAINBOW, April 1984) is based on random motions of the ball and wheel while the betting payoffs are strictly formula derived. The human interaction is used only for changing the initial conditions. These three methods of prophesy; formulae, random (probability), and human, all coupled through logic, form the basis for all computer Simulations.

What is the difference between a computer Simulation and a computer Model? Actually, very little. The difference is about the same as the difference between human anatomy and human physiology. One is the structure of the object while the other is the function of it. A Model is nothing more than a scaled-down replica of an object so it can be studied more easily, cheaply, or safely than studying the real object. A computer Model is a computer-scale replica of an object or a process. A computer Simulation is the function of the computer Model. The Model is the "program;" the Simulation is "running the program." To have a successful Simulation one must begin by building a reasonable Model of it. You must determine what you want it to do, then, limited by your resources, you write a program to do it.

This article is the first of a series discussing the makeup of a computer Simulation, how to implement the idea into a usable computer program, and how to use its results. I will discuss the fundamentals showing you how they can be used in a scientific Simulation of orbital motion. The next article in the series will emphasize human thought by the "investor" in a realistic simulation of the stock market. I will also discuss some special hints for simulating war (strategic con-

flict and tactical conflict), simulating sports events and, a brief word about human thought Simulations (artificial intelligence).

Once the idea or problem is formulated (in this case, orbital motion) I must define a "universe." This term sounds more alarming than it really is. The universe simply provides me with the boundaries in which to work. For instance, shall I simulate the entire solar system (a problem with 10 or more independent objects), or the entire Milky Way galaxy (billions of variables)? No, for purposes of illustration, I will choose a simple planet/satellite system and allow myself to vary the laws which govern the force between the two bodies. For fun, I want to be able to alter the motion of the satellite during the course of the Simulation. I will also add some random processes later to simulate "random" meteorites, etc.

Defining the universe is just the first step of placing constraints on the Simulation. Thorough knowledge of your computer is required to really form the basis for the Simulation. Remember, the CoCo uses five bytes for each variable so storing the position, the velocity, and the acceleration (all in three dimensions), requires 45 bytes for *each* body in motion. This may not be a problem for a simple solar system Simulation where we only worry about the major planets and their motion, but it becomes formidable when we start to include the dozens of moons and hundreds of asteroids, not to mention keeping track of the rotations, magnetic fields, etc., of each. Pretty soon the biggest constraint to the Simulation universe becomes the computer itself.

For sake of simplicity and illustration my universe contains one planet, with a mass much larger than the satellite (so it doesn't move), and I will restrict motion to two dimensions rather than three. This is actually pretty reasonable since two bodies in space will only move in a two-dimensional plane anyway. This also allows me to watch the motion in graphics form rather than just stare at a stream of numbers.

Now that I know my universe, I must give it a start. I must define my "initial boundary conditions." The "final conditions" are not defined for this Simulation, but in many cases we may want to know them, e.g. the state of the satellite after two hours. In that case the Simulation will halt after the final conditions are met. Computer Simulations are equally useful and valid in either case. A spreadsheet calculation from *Elite*Calc* is nothing more than a Simulation with open final boundary conditions.

The set-up clearly defines the universe while the number entry inputs the initial boundary conditions. For my Simulation, I will put the planet in the center of the solar system

(on the graphics screen) and place the satellite close to it. The computer can then prompt me for an initial velocity and direction of the satellite. I will then be able to observe its trajectory (orbit). I may want to see what a satellite with random initial direction will do. The Simulation will let me select random initial conditions. I also want to be able to alter the velocity of the satellite by "human interaction." That is accomplished by scanning the keyboard during the course of the Simulation to search for an arrow-key press. The right arrow speeds up the satellite while the left-arrow slows it down. This could be used to simulate an OMS (orbital maneuvering system) burn of the space shuttle. With this set of conditions and a universe you're ready to key in the program listed and begin the Simulation.

Begin by choosing "deterministic" starting conditions. When the Simulation asks for a velocity, enter '1'. (Don't type the quotes). For the angle, try '90'. The orbit should be a nice ellipse, just like Kepler predicted. Hit BREAK and restart. Try velocity=2, angle=90. To get a circular orbit, try velocity=1.3, angle=90. Now go ahead and play with it. You will see all three of Kepler's laws demonstrated. Some of the orbits will not be closed, that is the satellite will go screaming off the screen. These are parabolas and hyperbolas. If your satellite goes near the planet, you will see the famous "slingshot" effect. The dots plotted are equal timesteps so notice how the satellite speeds up near the planet. This acceleration has been used successfully to send probes to the outer planets as well as men to the moon. If you measure the area of the triangle formed by any other two adjacent dots and the planet, it will be the same area as the triangle formed by any other two adjacent dots and the planet. Kepler thought of this one, too. (Note to science students: Don't be too picky; I know that the equal area law is not made up of triangles, but it uses areas of sectors of the ellipses. If you can figure an easy way to measure the areas on the non-square video display, you'll be accurate enough.)

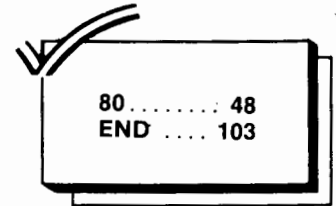
While you're at it, try "driving" the satellite around using the left- and right-arrow keys. It's an art to get used to exploiting the laws of orbital mechanics, but it's like riding a bike; once you have it, you have it forever. If you don't want to plot the entire trajectory but just want to see the satellite and the planet, change the MODE in Line 10 to MODE=0.

There is one other neat change you can make. Remember, I said that I wanted to be able to vary the force between the two bodies? Well, now's your chance to be Isaac Newton. The law of universal gravitation states that the force between two bodies is proportional to the *square* of the inverse of the distance between them. If that got by you, don't worry. It just means that the exponent in the denominator of the equation that calculates the force is 2. If the exponent is bigger the force would be less; if smaller than 2, the force would be greater. Kepler (him again) showed that only the exponent 2 would give you closed orbits — ones that come back to where they started and repeat. I wanted to see if he was right.

To change the exponent, just retype Line 20 with N equal to anything you like. Try 20 N=1.5. Now RUN the Simulation and enter the initial conditions that gave you an ellipse, velocity=1, angle=90. Let it go. Watch the orbit of the satellite now that the force law is changed. Boy, am I glad we don't live in a universe like that; the moon would be full for a few days, then it would come ripping by, creating tides you wouldn't believe. Then it would go away and take longer to come back. Every month would be longer until the moon just went away.

This simple, short, but powerful computer Simulation

allowed us to prophesy the end of the world as we know it just by altering the laws of motion (no simple task). Computer Simulations don't have to be long, complicated, number-crunching beasts. Just create your universe and give it a push.

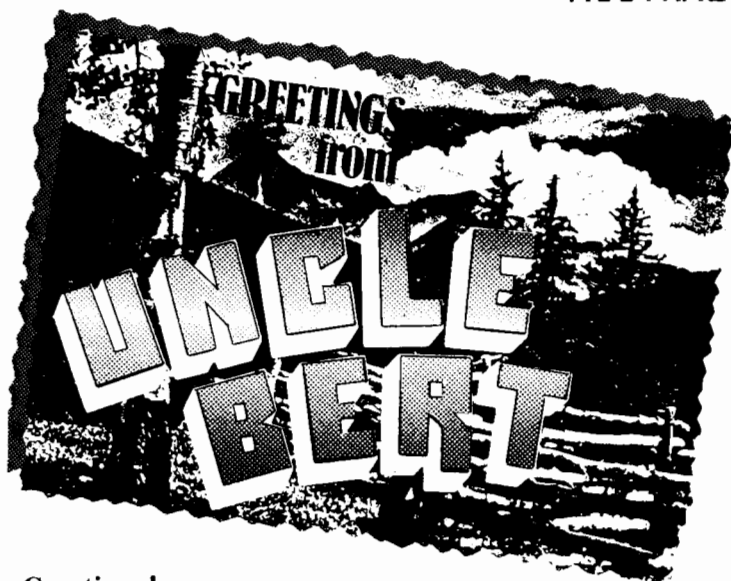


The listing:

```

5 'SIMULATION OF ORBITAL MOTION
  BY DR. BOB TYSON - 1984
10 PCLEAR 4:ZX=128:ZY=96:PX=128:
  PY=32:MODE=1:CLS
20 N=2.'N=EXPONENT OF THE FORC
  E LAW
30 V=RND(TIMER)
40 CLS:PRINT"ORBIT INVESTIGATION
  S":PRINT:PRINT"SELECT STARTING C
  ONDITIONS:      R=RANDOM
                  D=DETERMINISTIC"
50 K$=INKEY$:IF K$=""THEN 50 ELS
  E IF,K$="R" THEN 60 ELSE IF K$="
  D" THEN 70 ELSE 50
60 V=RND(5):A=RND(360):CLS:PRINT
  "VELOCITY=";V:PRINT"ANGLE OF EN
  TRY=";A;" DEG.":FOR I=1 TO 1500:
  NEXT:GOTO 90
70 PRINT:INPUT"SATELLITE VELOCIT
  Y 0-8";V:IF V<0 OR V>8 THEN 70
80 PRINT:INPUT"ANGLE OF VELOCITY
  0-360          CLOCKWISE FROM 12
  O'CLOCK POSIT. ";A
90 VX=V*SIN(A*.0174532):VY=-V*CO
  S(A*.0174532) 'VELOCITY COMPONE
  NTS
100 M=100:PMODE 4,1:SCREEN 1,0:P
  CLS
110 GOSUB 160
120 R=SQR((PX-ZX)^2+(PY-ZY)^2):A
  X=(M/R^N)*(ZX-PX)/R:AY=(M/R^N)*(
  ZY-PY)/R:VX=AX+VX:VY=AY+VY:PX=PX
  +VX:PY=PY+VY 'VELOCITY AND POSI
  TION CALCULATIONS
130 K$=INKEY$:IF K$="" THEN 110
  ELSE IF K$=CHR$(9) THEN U=1 ELSE
  IF K$=CHR$(8) THEN U=-1 ELSE EN
  D
140 V=SQR(VX*VX+VY*VY):VX=VX+U*V
  X/V:VY=VY+U*VY/V
150 GOTO 110
160 IF MODE<>1 THEN PRESET(QX,QY
  )
170 CIRCLE(ZX,ZY),2:IF PX<0 OR P
  X>255 OR PY<0 OR PY>191 THEN 190
180 PSET(PX,PY,1):QX=PX:QY=PY
190 RETURN

```



Greetings!

Summer has begun, as always, with the annual religious pilgrimage of the fleas. Apparently my farm is a "Holy Site" for fleas of a certain persuasion, and it seems that the pig barn remains their holiest shrine. Strange as it may seem, the earliest indications that the pilgrimage has begun appear not in the pig barn itself, but on the very personage of the primary guardian of that shrine, namely Ben.

It begins innocently enough with a few gloomy, hang-dog expressions and gestures. Ben begins finding excuses to be alone, to slink off into corners, to curl up under beds and tables, to lower himself with a groan to his favorite corner of a room. Then, when he thinks no one is looking or listening, the scratching and chewing begin. The scratching becomes violent — if Ben is next to a wall or piece of furniture, one hears a thump-thump-thump of hock against block. The chewing becomes obsessive — quiet at first, but soon accompanied by agonized snuffles and snorts. Within a few days, Ben has managed to produce bald patches on his back and haunches. One quick look at the bald spots will confirm the annual flea pilgrimage has begun.

Fleas love other animals — dogs and pigs and parrots and such — but they don't seem to like each other very much, I've noticed. Generally, here is what happens when one flea meets another:

```
TO FLEA1
  IF NEAR 2<50 (RT 90 FD 8)
  END
TO FLEA2
  IF NEAR 1<50 (RT 90 FD 8)
  END
```

In spite of their typical avoidance of one another, however, somehow they manage to crawl all over the place and replicate rapidly.

```
TO FLEAS
  HATCH 1 MOVE 4 96 90
  HATCH 2 MOVE 128 0 0
  VANISH
  END
TO MOVE :X :Y :H
  PU
  SX :X SY :Y SH :H
  REPEAT 150 (FD 8 IF ME=1 (FLEA1))
```

```
ELSE (FLEA2))
FLEAS
END
```

Well, that's more or less what the fleas look like when they finally appear, either on Ben's bald spots or on the pigs themselves. And you may have noticed that in the TO MOVE procedure I introduced yet another control statement — ELSE. The ELSE statement actually works only in conjunction with an IF statement. As I have demonstrated in some of my past letters, IF can be used by itself, and generally it says to the computer, "If such-and-such is true, do a certain action." The ELSE statement expands that instruction so that it reads: "If such-and-such is true, do a certain action, otherwise do another certain action." Since there are only two basic turtles (fleas) at work, I might have simply used two IF statements, like so:

```
IF ME=1 (FLEA1) IF ME=2 (FLEA2)
```

But the nice thing about ELSE is that it refers to everything else not carried out by the IF. Thus, if I had had five hatched turtles in the TO FLEAS procedure, turtle one would be instructed to carry out FLEA1, but turtles two through five would then be instructed to carry out FLEA2. Try it and see.

Now the fleas don't know this, and if they did they wouldn't care — but what is a religious pilgrimage to them is an invasion and an annual big nuisance to everyone else. Ben becomes so busy scratching and chewing himself he's not good for much else. Similarly, the pigs become so involved with the fleas crawling all over them that they stop playing, stop eating, and stop just about everything else that's useful. Clearly, we must terminally discourage those fleas. One possible way is to use flea poison. I don't like that because I don't like to use poison in the vicinity of farm animals. Also, the poison seems merely to slow down the fleas a little, but it doesn't really kill them. Let me show you:

```
TO POISON
  PU
  MAKE :X 0 SY 90 SH 90
  WHILE :X<64
```

Editor:
 How would you like a disk file to appear on the directory, but nobody except you can load it? To do this, save your file as follows: **SAVE "FILE"+CHR\$(143)**. The file will appear normally on the directory, but attempts to **LOAD "FILE"** will give you a ?NE ERROR. To load the file, use **LOAD "FILE"+CHR\$(143)**. I'm sure you can find variations on this process.

*Craig M. Arnold
 Dallas, TX*

```
(SLOW 4 FLEA)
WHILE :X<128
(SLOW 8 FLEA)
WHILE :X<192
(SLOW 16 FLEA)
WHILE :X<244
(SLOW 32 FLEA)
END
```

```
TO FLEA
MAKE :X :X+5
SX :X
END
```

By the way, the WHILE statement I just used is very much like the IF statement. IF tests to see if something is true; if it is true, then a certain action (in parentheses) is carried out. Likewise, WHILE tests to see if something is true; if it is true a certain action (in parentheses) is carried out. The difference is that IF tests once, and WHILE tests continually. As long as the condition is true, WHILE will continue to carry out again and again the specified action. For example, in the POISON procedure above, WHILE tests continually for the location of the turtle's X coordinate. While the value of X continues to be less than 64, the computer will continue to carry out the action of SLOW 4 FLEA.

Since WHILE tests continually for a certain condition and causes a specified action to happen continually as long as the condition is true, we can use WHILE to make an action continue forever, merely by specifying a condition that will always be true.

```
TO FLEABITE
WHILE ME=0
(SX RANDOM 230
SY RANDOM 170
REPEAT 1000 ()
PRINT ".")
END
```

In the above FLEABITE procedure, ME is always 0, since there are no hatched turtles. Thus, the WHILE statement forces the procedure to repeat itself indefinitely. Of course, we could have the same effect by turning FLEABITE into a simple recursion, like so:

```
TO FLEABITE2
SX RANDOM 230
SY RANDOM 170
REPEAT 1000 ()
PRINT "."
FLEABITE
END
```

But the WHILE statement potentially can give us at least one advantage in this kind of use.* Normally, in an indefinitely repeating procedure such as FLEABITE2, there is no way to stop the procedure except by hitting the BREAK key, at which time the procedure stops — but at the same time we go into the BREAK corridor, and lose our picture. What if we are creating constantly changing pictures with an indefinitely repeating procedure, but we want to be able to stop and look at any of the pictures? Is there any way of stopping without hitting BREAK? Yes, we can combine the WHILE statement with the KEY function.

The KEY function asks the computer to tell us the secret

computer code (called the ASCII code) number for whatever key has been pressed on the keyboard. If no key has been pressed, the KEY function yields the value of 0. Thus, with a WHILE KEY=0 we can make a procedure repeat itself indefinitely until we press any standard key on the keyboard. Pressing a standard key on the keyboard means that KEY is no longer 0; the procedure stops; but we can still remain in the RUN room, and thus can still see the RUN screen. Why don't you try it with the FLEABITE procedure?

```
TO FLEABITE3
WHILE KEY=0
(SX RANDOM 230
SY RANDOM 170
REPEAT 1000 ()
```

* A second advantage is that WHILE does not use up as much memory as a recursion does.

```
PRINT "."
END
```

Let me be the first to admit that some people may not be very interested in stopping the FLEABITE procedure to examine a pattern. But what if you're working with real art? For instance, remember the KLEE procedure I described a while back? Wouldn't it be nice to have a KLEE permanently on your screen? You could hang the TV on your living room wall.

Anyhow, fleabites are terrible. They itch, and modern science so far has not come up with the perfect cure for them. That's why poor Ben and the pigs are forced to waste so much time and energy scratching and chewing. I have an idea, however. I propose that we combine the latest in computer technology and lasers to locate and surgically remove the little bites. Locating them is easy since Color LOGO includes the XLOC and YLOC functions for calling up X and Y locations of any turtle (in this case, turtle number 0, the mother turtle).

```
TO LOCATE
PU SX :X-24 SY :Y-20 SH 90
PRINT XLOC 0 FD 32 PRINT YLOC 0
END
```

And using a laser to remove the bites shouldn't be so difficult either.

```
TO LASER
CLEAR
END
```

In short, applying modern technology to the age old problem of fleabites may be a perfect solution.

```
TO BITECUREMACHINE
WHILE KEY=0
(HT MAKE :X RANDOM 210+20
MAKE :Y RANDOM 150+20
SX :X SY :Y
PRINT ".")
REPEAT 1000 ()
HATCH 1 LOCATE
REPEAT 1000 ()
```

LASER)
END

(REPEAT 12 (RT 45)
FD 12 REPEAT 48 (LT 45)
FD 10)
END

I've used WHILE KEY=0 to make the BITECUREMACHINE repeat indefinitely, until I press any standard key on the keyboard. I might also use WHILE KEY=0 to build a human-operated pause device, by using WHILE to make a meaningless action continue until I press a key. Like so:

```
TO MACHINE
HT MAKE :X RANDOM 120+20
MAKE :Y RANDOM 150+20
SX :X SY :Y
PRINT"."
WHILE KEY=0
(PU)
HATCH 1 LOCATE
REPEAT 1000 ()
LASER
MACHINE
END
```

Of course, some destructive personage might try to steal the laser from my bite-cure machine and use it as a weapon against fleas. What would happen? We can only speculate, but knowing how tough fleas are, I would guess that a minor microsurgery laser might do nothing but disorient them for a while.

```
TO FLEAINJURE
PU
MAKE :N 2
SX 40 SY 90 SH 90
WHILE XLOC 0<64
(SPIRAL1)
WHILE XLOC 0<128
(SPIRAL2)
WHILE XLOC 0<192
(SPIRAL3)
WHILE XLOC 0<244
(SPIRAL4)
FLEAINJURE
END
```

```
TO SPIRAL1
FD :N RT 60
MAKE :N :N+1
END
```

```
TO SPIRAL2
FD :N LT 60
MAKE :N :N+1
END
```

```
TO SPIRAL3
SLOW 10
FD :N RT 360 LT 360
RT 60
MAKE :N :N+1
END
```

```
TO SPIRAL4
SLOW 20
REPEAT 4
```

I think that is a reasonable demonstration of a flea in deep trouble. However, I had hoped (with the extra FLEAINJURE at the bottom of the TO FLEAINJURE procedure) that the flea would recycle through the entire sequence. It didn't. I tried to figure out why, and then realized that my last WHILE statement -- WHILE XLOC 0<244 -- remained permanently true, thus keeping the moving turtle locked into that part of the overall procedure. So, I changed the last WHILE statement to an IF statement, assuring that the procedure would recycle itself. Another thing I found: when the turtle (flea?) did finally recycle it was still carrying the SLOW 20 command from SPIRAL4. So I put a SLOW 0 command at the beginning of SPIRAL1 to cancel out the SLOW 20. Make sense?

All I can say is we better do something about these fleas. Otherwise:

```
TO INFEST
MAKE :X 20
SX :X SY 20
HATCH 1 CRAWL
SET
HATCH 2 CRAWL
SET
HATCH 3 CRAWL
SET
HATCH 4 CRAWL
SET
HATCH 5 CRAWL
SET
HATCH 6 CRAWL
SET
HATCH 7 CRAWL
SET
HATCH 8 CRAWL
SET
CRAWL
END
```

```
TO CRAWL
PU
SLOW 5
RT 30 FD 10
REPEAT 6 (RT 60)
REPEAT 6 (LT 60)
LT 60 FD 10
RT 30
CRAWL
END
```

```
TO SET
SX XLOC ME + 25
END
```

Do you have the feeling you've seen more of fleas than you ever wanted to? So do I! So does Ben! So does Bertha! Ditto the pigs! So does everyone down here! I remain,

EXPANDING BASIC

64K
Disk

COOKING WITH COCO

PART I

In which we gather together the ingredients and utensils and explore the possibilities of CoCo's Disk Operating System.

By Colin J. Stearman

I know I do not need to tell you that CoCo is a powerful computer. You have probably spent as much time as I arguing its merits over those "fruity" and "big blue" machines. So while we are in agreement thus far, you'll surely also agree that even the "best" can be improved.

In this series of articles over the next few months we will explore how to incorporate many improvements, some of which are often only found on systems costing 10 times as much. I hasten to add that these improvements will be completely incorporated into the operating system and will be there when you want them. There have been other articles detailing enhancements, but they always involve loading programs into memory and they never seem to be there when you need them. Not so with the enhancements we are going to cook up here!

What exactly are we going to enhance and what is it going to take to do it? These articles are aimed at the standard 32K Disk CoCo system running with version 1.1 of Color BASIC, 1.0 of Extended Color BASIC and 1.0 of Disk Extended Color BASIC. The earlier 1.0 version of Color BASIC will probably work also, but the 1.1 version of Disk BASIC will not without modifying the programs presented here.

Although I will give you every assistance, it is going to take some skill on your part. Even the best written recipe requires the cook to add his skill. Some of the enhancements require hardware construction and some electronic construction skills. Others will involve the assembly of machine language programs. But none of it is really difficult and if you have a 64K system you can have almost all of the enhancements without even lifting a screwdriver.

Required Utensils

If you're going to attempt the hardware projects, you will need the normal set of screwdrivers, pliers, cutters and a soldering iron. If you are about to embark on a "hardware hacking" career, your local Radio Shack can accommodate you.

The assembly language programs will require an assembler. *EDTASM+* will do the job, but I much prefer *MAC* from Computerware. This is what I use and I will attempt to

point out the differences when necessary. For typing in the source code, a good editor is a must.

The Glossy Photo

Every good cookery book has glossy photos of the finished dish to whet your appetite. Our photo is by way of a list of the more tasty features:

- *FAST and SLOW to control CoCo's clock speed
- *XEQM to load and execute a machine code program
- *DATES to return a string containing the date
- *Directory pause
- *File creation date in the directory
- *Confirmation of the Kill request
- *WPEEK/WPOKE 16-bit word *PEEK* and *POKE*
- *Error trapping in BASIC programs
- *Error code, error line and error name functions
- *Auto execution of a BASIC file on start-up
- **AUTO* line numbering, with start line and increment
- *Flexible keyboard entry (*FLEXIKEY*)
- *Fully spelled-out error messages
- *SCANS, "INKEYS" with built-in wait for key press
- *40-track versions of *DSKINI*, *BACKUP* and *DSKIS/DSKOS*
- *Fixes to the *FILES* and *PCLEAR* bugs
- *Up to *PCLEAR 16* allowed
- **BAUD* command to set Baud rate
- *Parallel printer port
- **LDIR* to send the directory to the printer
- *And more . . .

By now your mouth should be thoroughly watering so let's start cooking!

Underlying Principles

When Microsoft wrote the BASIC operating system for Radio Shack they planned ahead and left numerous "hooks" in the code to allow modifications and changes. These hooks take the form of jump instructions located in the lower RAM (random access memory) area of the map. Many of the useful subroutines in BASIC first jump to these hooks, making it very easy to intercept their function and modify or completely change.

Fortunately for us, when Microsoft was contracted to write Disk Extended Color BASIC (DECB), something odd happened. Color BASIC (CB) and Extended Color BASIC (ECB) fully occupied their 8K ROMs (read only memory). But DECB did not come close to filling the 8K. In fact, some 2000 or so bytes were unused. Maybe money or time ran out, but this available space can be put to good use for all those added functions mentioned earlier. The only requirement is to come up with a way to permanently insert the new instructions in this available place.

There are two ways to do this. We can either replace the ROM with a similar EPROM (Eraseable Programmable ROM) containing our additional code, or we can make use of the 64K RAM capability of our CoCo (if we have it). The EPROM approach requires the design of an EPROM programmer and that will be the subject for next month. But the 64K method requires no hardware and does nearly as good a job, so for the remainder of this installment I'll detail what I mean.

Disk Resident DOS

If you have installed 64K memory chips and the now

famous "Frank Hogg Modification," you know that CoCo can run in an "all RAM" mode in which the three BASIC ROMs play no part. Using this technique it is possible to store the entire BASIC operating system on a specially prepared disk and then boot it into the all RAM system and start it up. In fact, for many computers (the IBM PC, for example) this is the only way of loading the DOS (disk operating system) and is the normal procedure for getting things started at turn on.

If we give CoCo the ability to boot or load its DOS from disk, there is nothing to say that we cannot modify the contents as we desire. As a result we can have the original DOS in the internal ROMs and our enhanced DOS on a special "system disk."

I said this approach is nearly as good as "burning" EPROMs with the modified code. There are some limitations. If you press the Reset button while running the disk-resident DOS, you will be summarily returned to the ROM version. Also, if you run some application programs which make use of CoCo's 64K capabilities, you will probably be returned to the ROM DOS when you exit them. But disk-resident DOS (let's call it DRDOS) can be booted and running in about 10 seconds, so this is not much of a penalty. Further, there are not just 2,000 or so bytes available for enhancements, but using all the address space from \$D7DD to \$FFEF, there are some 10,000 bytes. This is because we are not limited to the 8K increments and socket space of the ROMs.

Two machine code programs are needed here — one to get DRDOS saved onto disk and the other to retrieve it and start it up. The first I called *SYSSAVE* and the second *SYSTEM*. As a result, the currently running DOS, modified as desired, can be saved to disk by *SYSSAVE* and recovered by *SYSTEM*.

Running BASIC In RAM

BASIC cannot run in a 64K RAM environment without two slight modifications. When it goes through its start-up procedure it switches back to the regular RAM/ROM configuration and we would rather it did not. Second, it goes through a sizing procedure to find out exactly how much RAM is available (remember the days of 4K and 16K?). This testing plays havoc in the 64K RAM mode and must be removed. We already know that BASIC has 32K to work with, so we can skip the testing and report this number immediately. This savings in bytes provides just the room we need to fix the first problem.

So, the first thing we must do is copy an image of BASIC from the three ROMs into the RAM, slightly modify it and then start it up. This is done by a program called *BASLOAD*, shown in Listing 1. This is entirely a BASIC program, but it does load a simple machine code routine and the source for this I have included as REM statements at the end. The program is singularly anticlimatic! After a few seconds a tone sounds and the start-up credits are issued. Nothing seems to have changed. But, in fact, you are in a 64K RAM environment. Don't believe me? Try *POKE&HE000,55* and then *PRINT PEEK (&HE000)*. You'll get the 55 back because RAM is at \$E000. In the ROM system you will POKE to no avail.

By the way, I'll be using the assembler convention, throughout these articles which says that a "\$" in front of a number says it's hexadecimal; a "%" means binary and nothing in front means it's decimal. But in BASIC statements I will use "\$H" to signify hexadecimal.

Type in the program in Listing 1, save it to a convenient

disk and then run it. If you get the tone and new credits everything ran fine and we're ready to save the slightly modified system to a disk. To make absolutely sure your RAM version is okay, type *POKE113,0:EXEC\$HA027*. This will do a cold start of the BASIC in RAM and should clear the screen and display the credits. After you're sure it works, get back to the ROM version by typing *POKE113,0* and then pressing Reset. I'll hang around here till you get back!

Saving DOS To Disk

The currently running operating system is saved to disk with a program called *SYSSAVE.BIN*. Once the assembler has created the binary file it is just run by the *LOADM* and *EXEC* commands.

SYSSAVE will request which drive (0 or 1) you wish to save the system to. This drive should contain a blank, formatted disk. The program will then save the contents of memory from \$8000 where ECB starts, up to \$FF00. This is one more than the highest useable memory. From here to \$FFFF are system addresses and vectors. It does not matter whether you have anything in the saved range, it just stores what is there on the disk. As DECB starts at \$C000 we could extend it up to \$FFEF and be able to run the system in RAM. (That would be a lot of capability, as all the enhancements I listed earlier will fit into the 8K space allotted to the DECB ROM from \$C000 to \$DFFF.

The bytes are stored on disk on tracks 0 through 6, plus the first sector of track 7. This means that granules 0 through 14 are used and unavailable to BASIC. The granule map on track 17 sector 2 is modified to reflect this. Therefore, once a system has been saved to a new disk, the *FREE* function will return a value of 53, even though the directory shows no files.

Sector 1 on track 17 is not used by BASIC, so the first byte is set to \$55 to indicate that this is a system disk. When *SYSSAVE* is run it first checks that this \$55 is there. If it is, then a system can safely be stored on the disk. If not, then this disk has never had a system saved on it before. In this case, *SYSSAVE* checks that the first 15 granules are free. If so, the system can be saved. If not, a "DISK NOT FREE FOR SYSTEM STORAGE" message is returned and *SYSSAVE* gives up. As a result it should not be possible for *SYSSAVE* to overwrite valuable data on a disk.

To run *SYSSAVE* it must first be entered as shown in Listing 2 and then assembled. If you're using *EDTASM+*, leave out the lines with mnemonics "NAM" and "OPT" in them; these are just directives to my *MAC* assembler. This will be true for all future assembly language programs. Another mnemonic *MAC* has which must be converted for

EDTASM+ is the FCS instruction. This forms a constant string and allows embedded hexadecimal control codes and automatically adds a terminating zero byte. So the line in *SYSSAVE* which I have as:

```
FCS / <0D>DRIVE NUMBER (0 OR 1) ? /
```

would become:

```
FCB $0D A CARRIAGE RETURN
FCC /DRIVE NUMBER (0 OR 1) ? /
FCB 0 TERMINATING ZERO
```

You can convert all other FCS instructions into these groupings and *EDTASM+* will like them just fine.

When the code assembles correctly and you have checked it carefully, the only thing left is to try and run it! The best technique is to first load and run *BASLOAD*. This gets the system running in RAM and suitably modified for this environment. Now *LOADM"SYSSAVE"* but don't execute

it yet. Then remove all important disks from all your drives as chaos may be about to reign. Load a blank, formatted disk in drive 0 and type in EXEC.

The screen will clear and a request will appear asking which drive to save to. Enter a zero. Drive 0 should whirl for a few moments and the OK prompt appear. If not, it's back to the editor and look for that inevitable typo!

The system has now been saved to disk. A couple of checks will help confirm this. Type in PRINTFREE(0) and a value of 53 should be returned. Another check would be to type the following commands:

```
CLEAR 500
DSKIS 0,17,1,AS,BS
PRINT HEX$(ASC(AS))
```

This last line should print the value 55. But the ultimate test is to try to retrieve and run the saved system.

Booting From Disk

If you study the code of SYSTEM you will find it very similar to SYSSAVE, and it is hardly surprising. Type in and assemble the program in Listing 3. When you've thoroughly checked it for typing errors and are certain it is right, put a write-protect tab on your system disk anyway. Then when the impossible happens, your saved system won't be erased.

Now LOADM the binary file called SYSTEM, remove the disk and place the system disk in drive 0. SYSTEM always boots from drive 0. Then EXEC the program. Once again the screen will clear and a message will announce what is happening. Drive 0 will run and you will hear the head moving. When it is finished you will be requested to input which "flavor" of BASIC you want, CB, ECB or DECB. Pressing the appropriate letter will cold start that version. This feature is a convenient way of disabling DECB should you want to run one of the other configurations.

If everything worked as expected, you can copy the SYSSAVE and SYSTEM source and machine code files to your system disk. Then they will all be in the right place. I also wrote a simple BASIC program to start up SYSTEM which you might want to include. Then, if you call it BASIC you can just type RUN BASIC. It is:

```
10 DISK OPERATING SYSTEM LOADER
20 LOADM"SYSTEM"
30 EXEC
```

To remove the system from a system disk and make the full 68 granules available, the simplest way is to reformat with the DSKINI command. Don't have anything else valuable on the disk though, as it will be erased.

Wrapping It Up

You now have the first tool to be used later in the DOS enhancements. When these have been installed and saved to a system disk, they can be booted at power-up and all the features will be there without absorbing any RAM space. Even if you intend taking the EPROM route, it's still a good idea to have these programs as it makes testing quicker and easier.

Which brings me to next month. Putting the enhanced version of the DOS in an EPROM is certainly a nice way to go. Then everything is there just as soon as the power is turned on. So, next month we will start the EPROM programmer. This is a very simple hardware project using only three chips! Most of the work is done by the software. So, if you've ever had a soldering iron in your hand, give it a try.

Throughout this series I will be happy to try to answer

related questions which might arise. Please address them to me at 143 Ash Street, Hopkinton, Mass. 01748 and enclose a S.A.S.E. Be as precise as you can and give me a few weeks to get back to you. You can also send me EMAIL on CompuServe to 71036.256.

See you next month!

LISTING 1. (BASLOAD) @ end of listing 3.
was missing from original article. (km)

```
SYSSAVE COMPUTERWARE MACRO ASSEMBLER
COOKING WITH COCO PART 1/LISTING 2 (C)1984 COLIN J. STEARMAN

0004 OPT NOG,LIS
0005 *
0006 * THIS LOADS BASIC FROM $8000
0007 * UP TO $FFF0 ONTO A BLANK
0008 * FORMATTED DISK. IT USES
0009 *THE FIRST 15 GRANULES.
0010 * 14 gran + 9 sectors + 256 bytes = 32256
0011 * plus
0012 * 1 sector = 32512 byte which cover from
0013 * $8000 to $FFF0. All of accessible upper
0014 * memory.
0015 *
0016 *****
0017 *SOME EQUATES
C002 0018 RETURN EQU $C002
B000 0019 BASIC EQU $8000
C004 0020 DSKCOM EQU $C004
C006 0021 DCOPC EQU $C006
A002 0022 CHROUT EQU $A002
A000 0023 POLCAT EQU $A000
A928 0024 CLEAR EQU $A928 DIRECT JUMP TO CLEAR ROUTINE
0025 *
0E00 0026 ORG $E00
0027 *
0E00 7F0F17 0028 SYSSAV CLR TRACK RESET TRACK POINTER
0E03 7F0F18 0029 CLR SECTOR CLEAR SECTOR POINTER
0E06 7C0F18 0030 INC SECTOR SET TO 1
0031 *
0E09 0DA928 0032 JSR CLEAR CLEAR SCREEN
0E0C 3080209 0033 LEAX TITLE.PCR LOAD TITLE MESSAGE POINTER INTO X
0E10 1700E3 0034 LBSR DISPLY DISPLAY IT
0035 *
0E13 308D0224 0036 ASKDMO LEAX DRIVNO.PCR ASK FOR DRIVE NUMBRP
0E17 1700DC 0037 LBSR DISPLY
0E1A AD9FA000 0038 REPET JSR [POLCAT] GET DRIVE NUMBER
0E1E 27FA 0039 BEQ REPET
0E20 AD9FA002 0040 JSR [CHROUT] ECHO ENTRY
0E24 B130 0041 CMPA #0 IS IT LOWER THAN ASCII ZERO?
0E26 25E8 0042 BLO ASKDMO YES
0E28 B131 0043 CMPA #1 IS IT HIGHER THAN ASCII 1?
0E2A 22E7 0044 BHI ASKDMO YES
0E2C 108EC006 0045 LDY DCOPC POINT Y AT PARAMETERS
0E30 8030 0046 SUBA #0 REDUCE TO A NUMBER
0E32 A721 0047 STA 1,Y SELECT DRIVE
0048 *
0049 *
0050 *GET SECTOR: TRACK17 TO SEE IF
0051 *THIS WAS A SYSTEM DISK.
0E34 8611 0052 LDA #17 TRACK
0E36 A722 0053 STA 2,Y
0E38 8601 0054 LDA #1 SECTOR
0E3A A723 0055 STA 3,Y
0E3C CC0F19 0056 LDD #BUFFER
0E3F ED24 0057 STD 4,Y
0E41 8602 0058 LDA #2 READ CODE
0E43 A7A4 0059 STA ,Y
0E45 AD9FC004 0060 JSR [DSKCOM]
0E49 6D26 0061 TST 6,Y ERRORS?
0E4B 10260091 0062 LBNE ERRORS
0E4F F60F19 0063 LDB BUFFER TEST FOR #55
0064 *BET EXISTING DISK MAP INTO BUFFER
0065 *
0E52 1700A6 0066 LBSR BETMAP
0E55 6D26 0067 TST 6,Y ANY ERRORS
0E57 10260085 0068 LBNE ERRORS
0E5B C155 0069 CNPB #55
0E5D 2605 0070 BNE NEWSYS
0E5F 8E0F20 0071 LDX #BUFFER+15
```

```

0E62 200E      0072      BRA      OUTMAP
                0073 *
                0074 *CHECK FOR 255 IN FIRST 15 BYTES
                0075 *IF NOT ALL 255 THEN DISK NOT AVAILABLE.
                0076 *
0E64 0E0F19   0077 NEWSYS LDA  #BUFFER      POINT X TO BUFFER
0E67 A680     0078 NXTBYT LDA  ,X+         GET BYTE
0E69 81FF     0079 CMPA  #0FF           IS IT 255?
0E6D 267A     0080 BNE  NOTAV          OUTPUT NOT AVAILABLE MESSAGE
0E6E 8C0F20   0081 CHPI  #BUFFER+15     DONE ALL 15?
0E70 25F5     0082 BLD  NXTBYT
                0083 *
                0084 *SET UP MAP AND WRITE OUT
                0085 *
0E72 86C6     0086 OUTMAP LDA  #9C6       LAST GRANULE POINTER
0E74 A782     0087 STA  ,X
0E76 060F     0088 LDA  #15              15 AT 14 ETC.
0E78 4A       0089 DONEXT DECA
0E79 A782     0090 STA  ,X
0E7B 8C0F19   0091 CHPI  #BUFFER      DONE ALL 15?
0E7E 22F8     0092 BHI  DONEXT
                0093 *
                0094 *PUT IT ONTO DISK
0E80 17007E   0095 LBSR  PUTMAP
0E83 6D26     0096 TST  #Y              ANY ERRORS?
0E85 10260057 0097 LBNB  ERRORS
                0098 *****
                0099 *MARK DISK AS A SYSTEM DISK BY
                0100 *SETTING BYTE1 IN SECTOR 1 TO #55 IN TRACK 17
0E89 8601     0101 LDA  #1              SECTOR
0E8B A723     0102 STA  #Y
                0103 *SET UP DRIVE OP CODE
0E8D 8655     0104 LDA  #55             MARKER
0E8F 870F19   0105 STA  BUFFER
0E92 AD9FC004 0106 JSR  [DSKCON]
0E96 6D26     0107 TST  #Y
0E98 2646     0108 BNE  ERRORS
                0109 *****
0E9A 8603     0110 LDA  #3              WRITE CODE
0E9C A7A4     0111 STA  ,Y
                0112 *POINT X AT START OF BASIC
0E9E 8E8000   0113 LDX  #BASIC
                0114 *
                0115 * START TRANSFER
                0116 *
0EA1 860F17   0117 NXTSCT LDA  TRACK      GET TRACK NUMBER
0EA4 A722     0118 STA  #Y
0EA6 860F10   0119 LDA  SECTOR      GET SECTOR NUMBER
0EA9 A723     0120 STA  #Y
0EAB AF24     0121 STI  #Y           BUFFER ADDRESS
                0122 *
0EAD AD9FC004 0123 JSR  [DSKCON]      WRITE BLOCK
0EB1 6D26     0124 TST  #Y           CHECK FOR ERRORS
0EB3 2620     0125 BNE  ERRORS      REPORT THEM
                0126 *
                0127 *INCREMENT VALUES
0EB5 3080100 0128 LEAX 256,X         MOVE BUFFER POINTER
0EB9 860F17   0129 LDA  TRACK         IS IT LAST TRACK?
0EBC 0106     0130 CMPA  #6
0EBE 2509     0131 BLD  NOTLST
                0132 *WE GOT HERE BECAUSE THIS IS THE LAST TRACK(7)
0EC0 860F18   0133 LDA  SECTOR
0EC3 8102     0134 CMPA  #2           LAST SECTOR IN TRACK 6
0EC5 2727     0135 BEQ  CLOSE
0EC7 2007     0136 BRA  INCMT       GO TO INCREMENT
                0137 *
0EC9 860F18   0138 NOTLST LDA  SECTOR
0ECC 8112     0139 CMPA  #18
0ECE 2705     0140 BEQ  NXTTRK
0ED0 7C0F18   0141 *BET HERE BECAUSE NOT ALL SECTORS DONE YET
0ED3 20CC     0142 INCMT INC  SECTOR
                0143 BRA  NXTSCT       DO NEXT SECTOR
                0144 *
                0145 *BOT HERE BECAUSE LAST SECTOR
0ED5 7F0F18   0146 NXTTRK CLR  SECTOR
0ED8 7C0F18   0147 INC  SECTOR
0EDB 7C0F17   0148 INC  TRACK
0EDE 20C1     0149 BRA  NXTSCT
                0150 *****
0EE0 3080170 0151 ERRORS LEAX ERR,PCR
0EE4 8D10     0152 BSR  DISPLY
0EE6 39       0153 RTS

```

```

0154 *****
0EE7 308017D 0155 NOTAV LEAX NTAV,PCR
0EEB 8D09     0156 BSR  DISPLY
0EED 39       0157 RTS
                0158 *****
0EEE 7FFF40   0159 CLOSE CLR  #FF40     TURN OFF MOTOR
0EF1 39       0160 RTS
                0161 *****
                0162 *
0EF2 AD9FA002 0163 DISPL1 JSR  [CHROUT]   DISPLAY ON SCREEN
0EF6 A680     0164 DISPLY LDA  ,X+     GET CHARACTER
0EF8 26F8     0165 BNE  DISPL1
0EFA 39       0166 RTS
                0167 *
0EFB 8602     0168 GETMAP LDA  #2       READ OP CODE
0EFD A7A4     0169 STORE STA  ,Y
0EFF 2004     0170 BRA  CONT
0F01 8603     0171 PUTMAP LDA  #3       WRITE OP CODE
0F03 20F8     0172 BRA  STORE
0F05 8611     0173 CONT LDA  #17      SELECT TRACK
0F07 A722     0174 STA  #Y
0F09 8602     0175 LDA  #2              SELECT SECTOR
0F0B A723     0176 STA  #Y
0F0D 8E0F19   0177 LDX  #BUFFER      BUFFER ADDRESS
0F10 AF24     0178 STX  #Y
0F12 AD9FC004 0179 JSR  [DSKCON]
0F16 39       0180 RTS
                0181 *****
                0182 *
                0183 * VARIABLES AND STRINGS
0F17          0184 TRACK RMB 1
0F18          0185 SECTOR RMB 1
0F19          0186 BUFFER RMB 256
1019 20       0187 TITLE FCS  / BASIC TO DISK<#D> STORAGE PROGRAM<#D><#D>/
1038 0D       0188 DRIVNO FCS  /<#D>DRIVE NUMBER (0 OR 1)? /
1054 0D       0189 ERR  FCS  *<#D><#D>READ/WRITE ERROR<#D>*
1068 0D       0190 NTAV FCS  /<#D>DISK NOT FREE FOR SYSTEM STORAGE<#D>/
                0191 *
0E00          0192 END  SYSSAV
                NO ERROR(S) DETECTED

```

SYMBOL TABLE:

ASKDNO 0E13	BASIC 0000	BUFFER 0F19	CHROUT A002
CLEAR A928	CLOSE 0EEE	CONT 0F05	DCOPC C006
DISPL1 0EF2	DISPLY 0EF6	DONEXT 0E78	DRIVNO 1038
DSKCON C004	ERR 1054	ERRORS 0EE0	GETMAP 0EF8
INCMT 0ED0	MARG 0000	NEWSYS 0E64	NOTAV 0EE7
NOTLST 0EC9	NTAV 1068	NXTBYT 0E67	NXTSCT 0EA1
NXTTRK 0ED5	OUTMAP 0E72	POLCAT A000	PUTMAP 0F01
REPET 0E1A	RETURN C002	SECTOR 0F18	STORE 0EFD
SYSSAV 0E00	TITLE 1019	TRACK 0F17	

CMD=SYSSAVE /P

SYSTEM COMPUTERWARE MACRO ASSEMBLER PAGE 1
COOKING WITH COCO PART 1/LISTING 3 (C)1984 COLIN J. STEARMAN

```

0004 OPT NOB.LIS
0005 *
0006 *THIS WILL LOAD A SYSTEM DISK
0007 *IN DRIVE 0 INTO 64K RAM AND
0008 *START IT UP
0009 *THE SYSTEM SHOULD HAVE BEEN SAVED
0010 *BY "SYSSAVE" AND OCCUPY THE FIRST 15
0011 *GRANULES ON THE DISK. A FLAG IN THE
0012 *FIRST BYTE OF TRACK 17 SECTOR 1 TELLS
0013 *IF THE DISK CONTAINS A SYSTEM
0014 *THIS WILL RESTORE FROM 0000 TO #FF00
0015 *****
0016 *
0017 *
0E00 0018 ORG 0E00
0019 *
0020 *SOME EQUATES
A002 0021 CHROUT EQU 0A002
A000 0022 POLCAT EQU 0A000
0000 0023 BASIC EQU 00000

```

AUSTRALIAN RAINBOW

```

C004      0024 DSKCON EQU 0C004
C006      0025 DCOPC EQU 0C006
A928      0026 CLEAR EQU 9A928      DIRECT JUMP TO CLEAR ROUTINE.
FFDE      0027 ROM EQU 0FFDE
FFDF      0028 RAM EQU 0FFDF
A027      0029 COLD EQU 1A027
          0030 *
          0031 *SET UP FOR DRIVE 0
          0032 SYSTEM LDY DCOPC
          0033 CLR 1,Y      DRIVE NUMBER
          0034 *
          0035 *CLEAR SCREEN
          0036 JSR CLEAR
          0037 *
          0038 *PUT UP TITLE
          0039 LDX @TITLE
          0040 LBSR DISPLY
          0041 *
          0042 *
          0043 *CHECK FOR SYSTEM DISK
          0044 BSR SYSCHK
          0045 *RETURN "A" AS 055 IF SYSTEM DISK
          0046 CMPA 055
          0047 BEQ DISKOK
          0048 LDX @MOSYS      POINT X TO NO SYSTEM DISK MESSAGE
          0049 LBRA DISPLY      OUTPUT IT
          0050 RTS
          0051 *
          0052 DISKOK BSR GETSYS
          0053 *
          0054 CLR 0FF40      TURN OFF DRIVE
          0055 STA RAM      SWITCH TO RAM
          0056 CLR 071      CLEAR TO COLD START
          0057 *ASK FOR WHICH SYSTEM TO BOOT
          0058 LDX @BOOT
          0059 LBSR DISPLY
          0060 POLAGN JSR (POLCAT)      GET RESPONSE
          0061 BEQ POLAGN      NONE YET?
          0062 CMPA 07B      IS IT BASIC?
          0063 BNE EORD
          0064 CLR 00000      SET UP COLOR BASIC
          0065 JMP COLD      GO TO IT
          0066 EORD CMPA 07E      IS IT EXTENDED BASIC
          0067 BNE ISITD
          0068 CLR 0C000      SET UP FOR EXTENDED BASIC
          0069 JMP COLD      GO TO IT
          0070 ISITD CMPA 07D
          0071 LBEO COLD      GO TO DISK BASIC
          0072 BRA POLAGN
          0073 *****
          0074 *SYSTEM DISK CHECK
          0075 SYSCHK LDA 017      TRACY
          0076 STA 2,Y
          0077 LDA 01      SECTOR
          0078 STA 3,Y
          0079 LDD 0BUFFER
          0080 STD 4,Y
          0081 LDA 02      READ OPCODE
          0082 STA ,Y
          0083 JSR (DSKCON)
          0084 TST 6,Y
          0085 BNE ERRORS
          0086 *SEE IF FIRST BYTE IS 055
          0087 LDA BUFFER
          0088 RTS
          0089 *****
          0090 *DISPLAY ROUTINE
          0091 DISPL1 JSR (CHRQUT)
          0092 DISPLY LDA ,I+
          0093 BNE DISPL1
          0094 RTS
          0095 *****
          0096 * SYSTEM LOAD ROUTINE
          0097 GETSYS LDI 0BASIC      POINT X AS START OF BASIC
          0098 *SET UP DRIVE
          0099 CLR TRACK
          0100 CLR SECTOR
          0101 INC SECTOR      TO SECTOR 1
          0102
          0103 *
          0104 DOTFR LDA TRACY      SET UP TRACK
          0105 STA 2,Y
          0106 LDA SECTOR      SET TO SECTOR
    
```

```

0E88 A723      0107 STA 3,Y
0108 *READ SECTOR
0109 JSR (DSKCON)
0E8E 6D26      0110 TST 6,Y
0E90 2628      0111 BNE ERRORS
          0112 *
          0113 *MOVE BUFFER INTO RAM AREA
          0114 BSR BUFMOV
          0115 *
          0116 *INCREMENT VALUES
          0117 LDA TRACK      IS IT LAST TRACK?
          0118 CMPA 06      HIGHEST FULL TRACK
          0119 BLO NOTLST
          0120 *WE GOT HERE BECAUSE THIS IS ON TRACK 7
          0121 LDA SECTOR      LAST SECTOR
          0122 CMPA 02      ONLY NEED ONE SECTOR ON TRACK 7
          0123 BNE INCMT      GO TO INCREMENT
          0124 RTS
          0125 *
          0126 NOTLST LDA SECTOR      LAST SECTOR IN OTHER TRACKS?
          0127 CMPA 01B
          0128 BEQ HITTRK
          0129 *
          0130 *GOT HERE BECAUSE NOT ALL SECTORS READ YET
          0131 INCMT INC SECTOR
          0132 BRA DOTFR      CONTINUE TRANSFER
          0133 *
          0134 *GOT HERE BECAUSE LAST SECTOR
          0135 HITTRK CLR SECTOR
          0136 INC SECTOR
          0137 INC TRACK
          0138 BRA DOTFR      CONTINUE TRANSFER
          0139 *****
          0140 ERRORS LDX 0ERR
          0141 BSR DISPLY
          0142 RTS
          0143 *****
          0144 * THIS MOVES 256 BYTES FROM BUFFER
          0145 *TO LOCATION POINTED TO BY REG X
          0146 BUFMOV LDU 0BUFFER      POINT U TO BUFFER
          0147 ORCC 050      DISABLE INTERRUPTS
          0148 STA RAM      SWITCH TO RAM
          0149 STORE LDA ,U+      GET BYTE AND INCR U
          0150 STA ,X+      STORE & INCR X
          0151 CMPU 0BUFFER+256      ALL DONE
          0152 BNE STORE      CONTINUE MOVING
          0153 STA ROM      SWITCH BACK TO ROM
          0154 ANDCC 00AF      ENABLE INTERRUPTS
          0155 RTS
          0156 *****
          0157 *STORAGE
          0158 BUFFER RMB 256
          0159 TRACK RMB 1
          0160 SECTOR RMB 1
          0161 ERR FCS *(0)READ/WRITE ERROR(0D)<(0D)*
          0162 NOSYS FCS /(0)NO SYSTEM ON DISK IN DRIVE 0<(0D)/
          0163 TITLE FCS / DISK BASIC LOADER(0D)<(0D)/
          0164 BOOT FCS /BASIC, EXTENDED OR DISK(B,E,D)?/
          0165 *
          0E00 0166 END SYSTEM
          NO ERROR(S) DETECTED
    
```

SYMBOL TABLE:

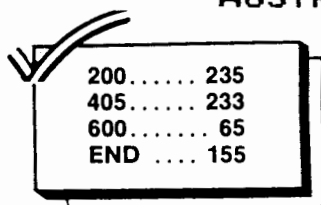
```

BASIC 0000      BOOT 1022      BUFFER 0E08      BUFMOV 0EC0
CHRQUT A002      CLEAR A928      COLD A027      DCOPC C006
DISKOK 0E1C      DISPL1 0E68      DISPLY 0E6F      DOTFR 0E80
DSKCON C004      EORD 0E3C      ERR 0FDA      ERRORS 0E8A
GETSYS 0E74      INCMT 0EAA      ISITD 0E46      NARG 0000
NOSYS 0FEE      NOTLST 0EA3      HITTRK 0EAF      POLAGN 0E2C
POLCAT A000      RAM FFDF      ROM FFDE      SECTOR 0FD9

STORE 0ECB      SYSCHK 0E4E      SYSTEM 0E00      TITLE 100D

CMD=SYSTEM /P
    
```

Martha Gripwhistle says: "Having problems finding those FC errors. Put a hammer through your monitor!"



Listing 1 (BASLOAD):

```

10 'THIS WILL TRANSFER BASIC
20 'EXTENDED BASIC AND DISK
30 ' BASIC TO ROM
40 ' CORRECT IT, THEN
50 ' COLD START IT.
60 ' IT WILL WORK WITH OR WITHOU
T
70 ' EXTENDED BASIC OR DISK BASI
C
80 ' IN ROM
90 'NOTE: For Color Basic 1.1 on
ly.
100 'Revs of Ext. and Disk not i
mportant
110 CLEAR 200,32511
120 DATA 32512,41044,41092
130 'RELOCATION PROGRAM
140 DATA 26,80,142,128,0,166,132
,183,255,223,167,128,140,224,0,3
9,5,183,255,222,32,239,28,175,57
150 ' PATCH #1
160 DATA 198,13,189,160,137,18,1
8
170 ' PATCH #2
180 DATA 142,127,254,32,10,167,1
93,90,38,251,206,255,224,57
190 READ S1,S2,S3
195 TT=S1+S2+S3
200 ' LOAD RELOCATION PROGRAM
210 FOR A=S1 TO S1+24
220 READ CODE
225 TT=TT+CODE
230 POKE A, CODE
240 NEXT A
245 IF TT<>117877 THEN PRINT"PRO
GRAM ERROR, PLEASE CHECK":STOP
250 '*SUBROUTINE IS NOW IN
260 'GO EXECUTE IT
270 EXEC 32512
280 SOUND 120,1' ANNOUNCE COMPLE
TION
290 ' OVERLAY PATCH #1 PREVENTS
MEMORY TYPE
300 ' FROM BEING SWITCHED BACK T
O ROM/RAM
310 FOR A=S2 TO S2+6
320 READ CODE
325 TT=TT+CODE
330 POKEA, CODE
340 NEXT A
345 IF TT<>118610 THEN PRINT "ER

```

```

ROR IN PATCH #1, PRESS RESET, RE
LOAD 'BASLOAD' AND CHECK":POKE11
3,0:STOP
350 ' PATCH #2
360 ' INITIALIZE PARALLEL PIA
370 FOR A=S3 TO S3+13
380 READ CODE
390 POKE A, CODE
395 TT=TT+CODE
400 NEXT A
405 IF TT<>120656 THEN PRINT "ER
ROR IN PATCH #2, PRESS RESET, RE
LOAD 'BASLOAD' AND CHECK":POKE11
3,0:STOP
410 ' CLEAR COLD START FLAG
420 POKE 113,0
430 'START UP BASIC
440 EXEC40999
450 ' THIS IS THE ASSMEBLY SOURC
E FOR THE
460 ' ABOVE CODE SEGMENTS
470 '*****
*
480 '* BASIC RELOCATOR
490 ' ORCC #$50 DISABLE
INTERRUPTS
500 ' LDX #$8000 BASIC
START ADDRESS
510 'LOOP LDA ,X GET A BYTE
520 ' STA $FFDF SWITCH
TO RAM MAP
530 ' STA ,X+ PUT BYTE
IN RAM
540 ' CMPX #$E000 END O
F BASIC
550 ' BEQ DONE ALL MOVE
D LEAVE IN RAM MAP
560 ' STA $FFDE SWITCH
BACK TO ROM MAP
570 ' BRA LOOP CONTINUE
MOVING
580 'DONE ANDCC #$AF ENABLE
INTERRUPTS
590 ' RTS RUNNING IN AL
L RAM SYSTEM
600 '*****
*****
610 '*PATCH 1 PREVENTS SAM FROM
BEING SWITCHED
620 '*BACK TO ROM MAP TYPE DURIN
G BASIC STARTUP
630 ' ORG $A054
640 ' LDB #0D ADDRESSES
TO SET IN SAM
650 ' JSR $A089 JUMP TO
NEW SETUP CODE
660 '* SPACE FOR THIS NEW ROUTIN
E IS MADE
670 '* AVAILABLE BY THE REMOVAL
OF THE MEMORY

```



```

680 '* SIZING ROUTINE IN PATCH #
2. MEMORY MUST
690 '* BY 32K TO EVEN BE DOING T
HIS.
700 '*****
*****
710 '*REMOVE MEMORY SIZE ROUTINE
AND INSTALL
720 '*SAM SETUP ROUTINE FOR PATC
H #1
730 '          ORG $A084
740 '          LDX #$7FFE MEMORY
SIZE
750 '          BRA CONT DO REST
OF ORIGINAL CODE
760 '*****
770 '* INITIALIZE SAM
780 'INIT      STA ,U++ WRITE TO
SAM
790 '          DECB      COUNTER
DOWN
800 '          BNE INIT DONE ALL
ADDRESSES?
810 '          LDU #FFE0 RESET U
FOR REST OF CODE
820 '          RTS TO CODE AFTER
PATCH #1
830 '          NOP FILLER BYTE
840 'CONT     EQU * FIRST BYTE
OF OLD CODE
850 '          END
860 '*****
*****

```

Software Review

Here's The 'Beef' Where's The Pork, Lamb?

More Beef is a program with a functional approach to aid many farmers, feed lots, feed mills and other such persons with interest in the beef or farming industries. It will provide a cost per pound value based on analysis of available feed rations. The program is provided on tape complete with instructions for loading to disk. It does require 16K Extended BASIC and is advertised to work on the TDP-100 and the Dragon computers.

First of all, let's establish a scenario. A feed lot operation has 200 head of 375-pound steers with a limited amount of their usual feed source available. The operator of the feed lot, being the aggressive, up-to-date person that he is, has his very own CoCo for multiple uses around the business. *More Beef* is one of the "CoCo jewels" available to our man. Geographic location is no problem as both metric and standard American measures are supported, by different versions of the program, both on the same tape.

The dilemma our man has to confront is that he has planned poorly and somehow has allowed his supply of feed to be less than required to support the operation. His ability to support the herd to full market weight is greatly impaired unless he can make the right decision.

He ponders frantically at what can be done to save face before he loses money or, worse yet, his herd. Several thoughts cross his mind. "I can sell the herd at feeder prices, if I'm lucky I can at least break even, and start over again next year." He begins to see his world crumble, and another idea develops. "I should have bought more hay, but the price per bale was so high this year. That drought last summer is what really ruined me." His mind is in such a turmoil that he can't think straight. "I would find another source for feed but what should I purchase?"

Tah-Daa! *More Beef* to the rescue! He turns on his CoCo and loads the *More Beef* program. Knowing the size of his herd and their feeding requirements, he uses the program and soon determines not only the most fitting feed source, but additional supplemental requirements and cost per head (excluding supplements) as well. With a great sigh of relief he gently slumps down into his chair as he praises the development of such business aids as *More Beef*.

The scenario could have happened anywhere in the world. At least anywhere that cattle are fed and anywhere that the CoCo is available. (It would be tough to raise a herd in either of the pole regions and a few other out-of-the-way places.) The important thing is that *More Beef* is a quality piece of software that provides a maximum level of flexibility and with a medium amount of effort will provide the desired results.

The program can't do it all. Using *More Beef* does require some knowledge about the environment you're working in. You would be required to supply or verify the following:

- What kinds of feed sources are available.
- What the herd requirements are thought to be.
- Units of measure (bales, pounds, grams, etc.).
- Approximate cost per defined unit (should be really close if not exact).

More Beef allows the user to control and edit all the data contained within the file which you develop using the program. The documentation was plentiful and used frequently in the beginning. As I became more proficient at using the program, the documentation was still helpful as a reference. Using the program on a tape system can be cumbersome, as the file needs to be reloaded each time you desire to process a request for different functions. On a disk-based system this would not be noticeable at all.

All in all, I really struggled trying to find something about the program I didn't like. Having been a part-time vocational school instructor I'm convinced that if I look hard and long enough, I can find something that could be better. Well, I finally saw something that could cause confusion, but by no means disrupts the function of the program. On the menu screen, when choices for action are listed, the first line shows the function followed by the selection number. On the second line these are reversed, showing the selection number then the function supported. (Or was it the other way around?)

I have one question for the source of this program and I haven't asked them yet but maybe they'll see this review and get the hint: Where's the "More Pork," the "More Lamb," the "More Chicken," etc.?

(Moreton Bay Software, 316 Castillo St., Santa Barbara, CA 93101, \$49.95, provided on tape with instructions for loading to disk)

— A.R. Compton



If you're gonna play the game . . .

YOU GOTTA PAY THE RENT!

By Gene Meador

Landlord is a 16K non-Extended BASIC game for two or more players. Its distant cousin is Monopoly, but I think you'll find that with so many changes, it is now a unique game.

Each player begins *Landlord* with \$10,000. You will invest this money by buying properties and building apartment complexes. By collecting rent from other players unfortunate enough to land on those properties, you try to gather enough wealth to win the game. There are actually two ways to win this game; by forcing all the other players to go bankrupt, or by reaching a money limit in cash and assets. The money limit is set by the players before the game begins.

If you'll look at Figure 1, you'll see the "board" used by the computer during the game. The computer will keep track of everything and will keep each player informed, so the board is not actually used or even needed. It's shown here to give you a mental picture of the game for a few turns until you get the hang of the game. The lot prices shown are the beginning prices only and are subject to change during the game.

Before the game begins, you will need to tell the computer the players names. Next, you will be asked to put in the money limit that each player is playing for. Since each player starts the game with \$10,000, the limit should be higher than that; a \$35,000 limit makes about a two-hour game. \$50,000 is about four to five hours of wheeling and dealing!

Everyone begins the game on payday and movement is clock wise around the board. At the beginning of each player's turn, his die roll is rapidly changing at the bottom of the screen.

The computer stops by hitting a number key, and if the number the player hits matches the number the computer was on at the time, the player receives an extra \$1,000 paycheck. Next, the computer will tell the player where he has landed, cash level, etc.

Good News and Bad News squares are just what they say they are. Beware of the Income Tax square; landing there will cost you 10 percent of your cash on hand!

Your CoCo will be the Bank and it will also keep track of each player's position on the board, his properties and hold-

ings, cash, rolling the dice, issuing paychecks to the players (for passing payday), and, of course, making sure everyone plays by the rules.

As the Bank, CoCo can do two important things. The Bank can loan money to the players. A player can borrow money whenever he wants. Of course, there are some catches! A player may only borrow up to his credit limit, which is a percentage of his assets (less any existing loans he already has). The more property you own, the more credit you have. As you might have guessed, the Bank charges interest on its loans. That interest rate depends on the Economic Index at the time. Every so often there will be a "news flash" announcing the new Economic Index and the new interest rate on loans. (Interest rates will never go below five percent, but there is no upper limit!) A player taking out a loan has his loan balance spread out over 10 equal payments. A payment will be taken out of the player's cash each time he passes on or lands on the Loan Payment Due square. A player has the option to make additional payments whenever he wants to, but they only reduce the number of payments, not the payment amount. It's a good idea to keep enough cash on hand to make your loan payments. Otherwise, you might have to go to the In-The-Hole square.

The In-The-Hole square is something like Monopoly's In Jail square, however it really doesn't come into play until a player tries to end his turn with a negative cash balance. If that happens, the player is given the following options: 1) get a loan from the Bank; 2) sell some property to another player; 3) go to the In-The-Hole square; 4) repossession of some of his properties by the Bank; and 5) quit the game.

If a player decides to go to the In-The-Hole, he may stay there for no more than three turns. If he still has a negative cash balance on his third turn, the option to stay there is deleted and the player has the other four options. In other words, once you go to the In-The-Hole square, the only way off of it is to get a positive cash balance within three turns or quit the game. (Paying \$50 won't save you in this game!)

Payday is, as mentioned earlier, the square all the players start the game on. Each time a player lands on or passes payday, he receives a paycheck from the Bank for 10 percent of the value of his holdings or \$2,000, whichever is higher.

All the other squares on the board are the Lots of the Landlord. They are spaced evenly around the board in eight groups of three lots each. A player must land on a lot in order to buy it from the Bank. If he wishes to do so, he need

Landlord
Figure 1

LOAN PAYMENT DUE	May Ave. \$2500	Lans-Brook \$2500	Parling \$2500	GOOD NEWS	Robinson \$3000	Mac-Arthur \$3000	Agency Blvd \$3000	INCOME TAX
Rockwell \$1800								Bella-Vista \$2500
Ann Arbor \$1800								Lakeview \$2500
Patterson \$1800								Park Manor \$2500
BAD NEWS								BAD NEWS
Hazelwood \$1500								Country Club Dr \$4000
Wash-Quater \$1500								Wishire \$4000
Man-Quater \$1500								Broadway \$4800
IN THE HOLE	Claasen \$800	Eastern \$800	Walnut \$800	GOOD NEWS	Oxreal \$500	Black-welder \$500	Agnew \$500	PAYDAY

only use the "Buy" option and the computer will handle the transaction for him. The "Recap" option is very handy. It gives you a complete rundown of your cash, position on the board, loan balances and payments left, credit limit, a complete rundown of all the properties, who owns them, and the number of apartments on each lot.

As you might have guessed, you must own all three lots of a group before you may build any apartment complexes on them. You may put up to 50 apartments on each lot. Each apartment will cost you 10 percent of the current lot price to build, which is quite a sizable investment for 50! Except for paychecks, and an occasional Good News once in a while, the rent collected from the other players who land on these improved lots will be your only income! Bare lots don't collect any rent. (The actual rent collecting is taken care of for you by the computer as its first order of business each turn.) Tenants are moving in and out of these apartments constantly, so the exact amount of rent that a player will receive depends on how many apartments on that lot are occupied at the time. (Don't worry, at least 60 percent will be.) In other words, just because a player has, say, 10 apartments on a lot doesn't mean that he will collect the rent for all 10 apartments each time someone lands on them. (Is nothing sacred in this game?)

When a player is In-The-Hole, the Bank has the ability to repossess a player's properties. (There is no mortgaging lot or apartments in this game. The Bank just takes possession.) Here's what happens when a player chooses that option: the Bank will begin at Payday and go clockwise around the board repossessing the player's holdings, lot by lot, while giving him one-half the current value for them until he either has a positive cash balance or he has no more property! Those repossessed properties may then be bought from the Bank by any player who lands on them.

Apartments, once built, can never be taken off that lot for the rest of the game. The lot and its apartments are sold or traded as a package deal, so don't forget to value them as such. Buyers should remember that they don't have to own all the lots of a group in order to collect rent from that lot, but they do if they wish to build any additional apartments on that lot.

Let me mention some fine points of the game and you should be ready to play:

- 1) If a player quits, the Bank will take over his holdings.
- 2) The Bank will collect rent from a player if you land on one of its improved lots. You may then buy it from the Bank if you'd like (and can still afford it!).
- 3) As protection to the players, you can't buy another player's property during your turn, but he can sell it to you during his turn.
- 4) Remember that even though you can win the game by bankrupting the other players, someone will usually win by reaching the money limit first. The key to winning this game is to make as much money as fast as possible.

The Program

I had several objectives when I wrote this program: to fit it into a 16K non-Extended Color Computer, write it as simply as I could so that beginning programmers could go through it and understand it, and finally, write some kind of game besides a space shoot-'em-up that the whole family could enjoy.

The program is simple, don't let the size intimidate you. By using the program outline and the variables list you can go through the listing and see that it's just a lot of *IF-THEN-*

GOTO programming. I've compressed the program lines to save memory, but I've used a lot of lines to spread it out so you could understand it easier.

I didn't use any special programming tricks, but there is one tip I'd like to pass on: if there is more than one of something, put them in an array. You can save yourself many program lines if you do. For instance, if P is the number of players and P1\$, P2\$, P3\$, etc. are the player's names, then to print out the player's names for turn identification you'd have to add something like this:

```
10 ON P GOTO 20,30,40,50
20 PRINT P1$;"'s turn":GOTO__
30 PRINT P2$;"'s turn":GOTO__
40 PRINT P3$;"'s turn":GOTO__
```

Now if you put that into an array, P\$(), you'll only need one line to take care of any number of players: 10 PRINT P\$(X); "'s turn". Now that's a pretty simplistic example but the point is that if you find yourself typing in several almost identical lines in your next program, take a close look at it. You might be able to use an array and shorten it.

After typing in the program and checking for mistakes, *CSAVE* it to tape. Then either *PCLEAR 0* or *POKE 25,6:NEW* and reload the game. Use the *POKE* or *PCLEAR* each time before you load it. As the game goes on, all those arrays get filled with information and you'll need all the memory you can get to keep from getting an OM? Error.

After hours of playtesting with my friends and family, it was decided not to display the board. It's just not needed to play the game and only served as a time-consuming distraction once you get into the game.

Program Lines:

10-160	Setup and credits
160-300	Players' names and continue setup
310-380	Get game limit and start game
410-420	Start of turn, get next player
450-480	Die roll
520-560	Check for passing payday
590-630	Income Tax
640-650	Loan Payment
660-810	Good News
820-960	Bad News
970-1080	Landed on lot; check out owner
1090-1130	Take rent out of player's cash
1140-1350	Main menu
1230-1350	Secondary menu
1360-1420	Buying property
1430-1600	Selling property
1610-1820	Trading property
1830-2090	Building apartments
2100-2360	Getting a loan
2370-2530	Recap and rundown of properties
2540	Check player's cash level
2550-2860	Deficient cash options
2860-2950	End of game
2980-3040	Payday
3050-3130	Economic Index
3140-3220	Rundown of properties

Variables List:

A\$()	Square names
PS()	Players' names
S	Square used during the turn
F()	Lot price

- LB() Loan balance
- LP() Loan payment
- A1() Used to check assets
- A2() Lot group number
- P Number of players
- H() Number of apartments on a lot
- P() Player's position on the board
- H1() Number of times In-The-Hole
- T Turn number
- L1 Game limit
- I Interest rate
- M() Player's cash
- Q() Used to identify players that have quit

For the more advanced programmers with 32K, here's a challenge: I've shown you the basics of how to write a program of this type, so why don't you create a program that plays Monopoly!

310	233	2100	113
550	33	2290	35
1000	32	2450	149
1340	113	2670	181
1540	232	2900	167
1710	155	3170	87
1920	241	END	111

The listing:

```

10 CLS: CLEAR200
20 PRINT: PRINT: PRINT
30 PRINT@105, "COLOR COMPUTER": PR
INT@140, "PRESENTS"
40 PRINT@201, "L A N D L O R D"
50 PRINT@261, "WRITTEN BY GENE ME
ADOR"
60 GOSUB2960: GOSUB2960: DIM A$(32
), P$(10), S(32), F(32)
70 DIM LB(10), LP(10), A1(32), A2(32
), P(10), H(32), Q(10), H1(10)
80 FORX=1 TO 32: READ A$(X): NEXT
90 DATA PAYDAY, AGNEW, BLACKWELDER
, DREXEL, GOOD NEWS!
100 DATA WALNUT, EASTERN, CLASSEN,
IN-THE-HOLE!, MANCHESTER
110 DATA WESTCHESTER, HAZELWOOD, B
AD NEWS!, PATTERSON
120 DATA ANN ARBOR, ROCKWELL, LOA
N PAYMENT DUE!, MAY AVE.
130 DATA LANSBROOK, PORTLAND, GOOD
NEWS!, ROBINSON
140 DATA MACARTHUR, REGENCY BLVD,
INCOME TAX, BELLA VISTA
150 DATA LAKEVIEW, PARK MANOR, BAD
NEWS!, COUNTRY CLUB
160 DATA WILSHIRE BLVD, BROADWAY
170 CLS(1): PRINT: INPUT "HOW MANY
ARE PLAYING"; P
180 FORX=1 TO P: PRINT "PLAYER #" X "
S NAME";
190 INPUT P$(X): M(X)=10000: P(X)=1
: Q(X)=0: NEXT
200 PRINT: PRINT " I'LL KEEP TRAC

```

```

K OF EVERYTHING", "FOR YOU, SO LE
T'S PLAY!"
210 GOSUB2960: M(0)=0
220 FORX=1 TO 32: READ F(X): NEXT
230 DATA 0, 500, 500, 500, 0, 800, 800
, 800, 0
240 DATA 1500, 1500, 1500, 0, 1800, 1
800, 1800, 0
250 DATA 2500, 2500, 2500, 0, 3000, 3
000, 3000, 0
260 DATA 3500, 3500, 3500, 0, 4000, 4
000, 4000
270 FORX=1 TO 32: READ A2(X)
280 NEXT: P$(0)="BANK": I=15
290 DATA 0, 1, 1, 1, 0, 2, 2, 2, 0, 3, 3, 3,
0, 4, 4, 4
300 DATA 0, 5, 5, 5, 0, 6, 6, 6, 0, 7, 7, 7,
0, 8, 8, 8
310 CLS(5): PRINT: PRINT "HOW MUCH
IN CASH AND ASSETS DO", "ALL OF Y
OU WISH TO PLAY TO?"
320 PRINT "($50,000 IS AVERAGE)."
: INPUT L1
350 CLS(5): PRINT@32, "OK! THE FIR
ST PLAYER TO REACH", "$" L1 "IN CAS
H AND ASSETS"
360 PRINT "WILL WIN THE GAME!!": P
RINT
370 PRINT "THE OBJECT OF THE GAME
IS TO ", "FIGURE OUT HOW TO WIN!
"
380 PRINT: PRINT "LOTZA LUCK! (YOU
'LL NEED IT!)"
390 GOSUB2960: GOSUB2960
400 M(0)=0
410 CLS(5): SOUND 180, 3: IF RND(20
)>18 THEN GOSUB 3050
420 T=T+1: IF T>P THEN T=1
430 IF Q(T)>0 THEN 420
440 PRINT@32, P$(T) "'S TURN": M(T)
=INT(M(T)): GOSUB2970
450 PRINT "PICK YOUR DIE ROLL: "
460 A=RND(11)+1: PRINT@98, A: R$=IN
KEY$: IFR$="" THEN 460
470 IF VAL(R$)<>A THEN PRINT "MIS
SED AGAIN!"
480 PRINT "YOUR CASH ON HAND IS $
" M(T)
490 GOSUB2970: PRINT
500 IF VAL(R$)=A THEN PRINT "EXTRA
PAYDAY!!!": XX=1000: GOSUB3030
510 IF M(T)<=0 THEN 570
520 IF (P(T)+A>17) AND (P(T)<17) THE
N 530 ELSE 540
530 PRINT "PASSED LOAN PAYMENT DU
E!": IFLB(T)>0 THEN GOSUB3240
540 P(T)=P(T)+A: IF P(T)>32 THEN P
(T)=P(T)-32: GOSUB2980
550 S=P(T): IF S<>9 THEN 590
560 IFS=1 THEN 2540

```

```

570 IFM(T)>0 THEN H1(T)=0:PRINT"
YOU'RE AT "A$(S):GOTO1150
580 H1(T)=H1(T)+1:SOUND1,30:GOTO
2540
590 IF S<>25 THEN 640
600 SOUND1,30:PRINT:PRINT"OH,OH!
INCOME TAX TIME!"
610 PRINT"YOU OWE 10% OF YOUR CA
SH ON", "HAND. YOUR CASH IS $"M(T
)"
620 PRINT"SO YOU OWE $"M(T)*.1"!
!"
630 M(T)=M(T)-(M(T)*.1):M(T)=INT
(M(T)):GOTO1140
640 IF S<>17 THEN 660
650 PRINT"YOU LANDED ON LOAN PAY
MENT!":GOSUB3240:GOTO1160
660 IF S<>5 AND S<>21 THEN820
670 FORZ=200TO225:SOUNDZ,1:NEXT
680 X=RND(5):PRINT"GOOD NEWS!!!"
:PRINT
690 ON X GOTO 700,720,760,780,80
0
700 XX=RND(35)*.1:I=I-XX:I=INT(I
):IFI<5 THEN I=5
710 PRINT"BANK INTREST RATES HAV
E DROPPED", "TO" I "% !":GOTO750
720 PRINT"PROPERTY VALUES JUST W
ENT UP", "10% ON ALL YOUR PROPERT
IES!"
730 FORX=1TO32:IFA1(X)=T THEN F(
X)=INT(F(X)+(F(X)*.1))
740 NEXT:GOTO750
750 GOSUB2970:S=P(T):PRINT:GOTO1
160
760 PRINT"INCOME TAX REFUND!"
770 X=RND(500)+300:PRINT"YOU GET
BACK $"X:M(T)=M(T)+X:GOTO750
780 PRINT"EVERYONE PAYS YOU $100
0!"
790 FORX=1TOP:M(X)=M(X)-1000:M(T
)=M(T)+1000:NEXT:GOTO750
800 PRINT"YOU GET GO TO PAYDAY!"
:GOSUB2970
810 P(T)=1:GOSUB2980:GOTO1160
820 IF S<>13 AND S<>29 THEN 970
830 SOUND1,40:PRINT:PRINT"BAD NE
WS.....":GOSUB2970:PRINT
840 X=RND(6):ON X GOTO 845,850,8
80,910,930,950
845 PRINT"EVERYONE VOTED YOU 'LE
AST ", "LIKELY TO WIN'!!":GOTO750
850 XX=RND(35)*.1:I=I+XX:I=INT(I
)
860 PRINT"BANK INTREST RATES HAV
E GONE UP", "TO" I "%!"
870 GOSUB2960:GOTO1160
880 PRINT"ALL YOUR PROPERTY VALU
ES HAVE", "DROPPED 10%!"
890 FORX=1TO32:IFA1(X)=T THEN F(

```

```

X)=F(X)-(F(X)*.1)
900 NEXT:GOTO870
910 PRINT"YOU PAY EVERYONE $1000
!"
920 FORX=1TOP:M(T)=M(T)-1000:M(X
)=M(X)+1000:NEXT:GOTO870
930 PRINT"IT'S INCOME TAX TIME A
GAIN!", "(AND NO PASSING PAYDAY!)
"
940 GOSUB2960:P(T)=25:GOTO550
950 PRINT"OH NO! LOAN PAYMENT DU
E!!":GOSUB2960
960 IFLB(T)>0 THEN:GOSUB3240:GOT
O1140
970 PRINT"YOU'RE NOW AT "A$(S):M
(T)=INT(M(T))
980 IFA2(S)=0 THEN 1000
990 PRINT"OWNER: "P$(A1(S)):PRIN
T
1000 IFA1(S)=T ORA2(S)=0 THEN 11
60
1010 IF H(S)>0 THEN 1090
1020 IFA1(S)>0 THEN 1160
1030 GOSUB2960
1040 CLS:PRINT:PRINT"LOTS OF THI
S GROUP:" :PRINT:FORX=1TO32
1050 IFA2(X)=A2(S) THEN PRINTA$(
X)"-"P$(A1(X))"'S-APTS.:"H(S)
1060 NEXT:PRINT:PRINTA$(S)" IS $
"F(S)+(H(S)*F(S)*.1)
1070 PRINT"(YOUR CASH IS $"M(T)"
)"
1080 PRINT"BUILDING COSTS: $"F(S
)*.1"PER UNIT":GOTO1160
1090 D=RND(40)+60:IFH(S)<=0 THEN
1160
1100 PRINT"OF" H(S) "APARTMENTS," I
NT(H(S)*D*.01)"ARE FILLED"
1110 RD=INT(H(S)*D*.01)*F(S)*.125
):PRINT"RENT DUE IS $"RD
1120 M(A1(S))=M(A1(S))+RD
1130 M(T)=M(T)-RD:RD=0:PRINT"YOU
R CASH IS NOW $"M(T)
1140 GOSUB2960
1150 GOSUB2960:CLS:S=P(T)
1160 PRINT"WOULD YOU LIKE TO:"
1170 PRINT" B)BUY, P)PASS, R)SEE
A RECAP,", " OR O)SEE OTHER OPTI
ONS?"
1180 R$=INKEY$:IFR$="" THEN 1180
1190 IFR$="B" THEN 1360
1200 IFR$="P" THEN 2540
1210 IFR$="R" THEN 2370
1220 IFR$="O" THEN 1230ELSE1180
1230 CLS:PRINT@32, "YOU HAVE THE
FOLLOWING OPTIONS:" :PRINT
1240 PRINT"S) SELL SOME PROPERTY
", "T) TRADE PROPERTIES"
1250 PRINT"A) BUILD MORE APARTME
NTS", "L) GET A LOAN"

```


to: **Graham Morphett**
PO Box 1742
SOUTHPORT
QLD. 4215

Cheque Charge my bankcard
 Money Order
 Cash

Send me:- \$ ----- Cardholder

AUSTRALIAN RAINBOW Magazine: Latest @ \$3.25 6 months \$18 one year \$29
 Back copies @ \$3 -----

AMERICAN RAINBOW on TAPE (programs listed) @ \$12 for month/s of -----

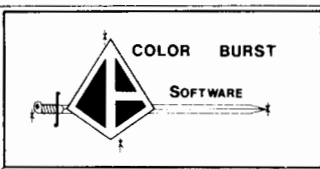
CoCoOz Tape Monthly- Latest @ \$6 6 months \$32 one year \$58
MiCoOz Tape Monthly- Latest @ \$6 6 months \$32 one year \$58

BLANK C30 TAPES 12 for \$18 or \$1.70 ea.

CASSETTE CASES 10 for \$5

BOOKS **Byte** Elementary \$5.95 **Help** Medium \$9.95 **Facts** Advanced \$11
MiCoHelp Medium \$9.95 **MiCo Exposed** V Advanced \$11.50

from ----- of -----



COLOR BURST SOFTWARE

Color Burst Software

QUALITY PROGRAMMES FOR THE TRS80 COLOR COMPUTER


P.O. BOX 256, ROSEVILLE, NSW, 2069 PHONE: (02) 467-1619

FOR A FREE CATALOG SEND NAME AND ADDRESS TO COLOR BURST SOFTWARE, P.O. BOX 256, ROSEVILLE, NSW, 2069. OR PHONE: (02) 467-1619

ARC = ARCADE	SIM = SIMULATION	ADV = ADVENTURE	GRA/ADV = GRAPHIC ADVENTURE	GME = GAME	EOC = EDUCATIONAL																																																																																													
ASTRO BLAST (16K ARC) 28.00	BLACK SANCTUM (16K ADV) 22.00	CALIXTO ISLAND (16K ARC) 25.00	DEFENSE (16K ARC) 28.00	DONKEY KING (32K ARC) 17.00	FROG TREK (16K ARC) 17.00																																																																																													
GALACTIC HANGMAN (16K GME) 17.00	GEOGRAPHY PAC (16K EDC) 33.00	GHOST GOBLERS (16K ARC) 22.00	HUD flight/simltn (16K SIM) 20.00	KATERPILLAR ATTK (16K ARC) 28.00	KEYS OF WIZARD (16K ADV) 22.00																																																																																													
LANCER/JOUST (32K ARC) 28.00	MATH DRILL (16K EDC) 22.00	MS. GOBLER (32K ARC) 28.00	PLANET INVASION (16K ARC) 25.00	SPACE COMMAND (16K ARC) 10.00	STORY PROBLEMS (16K EDC) 22.00																																																																																													
TEXT GUESS (16K GME) 10.00	TIMS DATA BASE (16K UTL) 28.00	TRAPFALL (16K GRA/ADV) 31.00	VIKING (16K SIM) 22.00	WHIRLYBIRD RUN (16K ARC) 28.00	ZAKSUND 3D (32K ARC) 30.00																																																																																													
BUMPERS (16K ARC) 29.00	ZONE 6 3D (16K ARC) 25.00	DEMON SEED (32K ARC) 32.00	CASHMAN (32K ARC) 23.00	QUEST (16K GRA/ADV) 31.00	WIZARDS TOWER (32K GRA/ADV) 23.00																																																																																													
DUNGEONS OF DEATH (32K GRA/ADV) 23.00	BAG-IT-MAN (32K ARC) 29.00	CATERPILLAR (16K ARC) 29.00	CAVE HUNTER (16K ARC) 29.00	JUNIORS REVENGE (32K ARC) 33.00	KOMET KAZE (16K ARC) 23.00																																																																																													
LABYRINTH (16K ARC) 23.00	MAD BOMBER (16K ARC) 22.00	MARS LANDER (16K ARC) 22.00	MEGAPEDE (16K ARC) 29.00	THE NIBBLER (16K ARC) 23.00	MS. NIBBLER (16K ARC) 23.00																																																																																													
MUOPIES (32K ARC) 32.00	OUTHOUSE (32K ARC) 32.00	PATTI-PAK (32K ARC) 32.00	PLANET RAIDERS (32K ARC) 29.00	RAIL RUNNER (16K ARC) 29.00	SEAWOLFE (32K ARC) 29.00																																																																																													
SHARK TREASURE (16K ARC) 29.00	SPACE RAIDERS (16K ARC) 29.00	TIME BANAIT (32K GRA/ADV) 32.00	TUT (32K GRA/ADV) 29.00	VENTURER (16K ARC) 29.00	WACKY FOOD (32K ARC) 26.00																																																																																													
XYGOTO (16K ARC) 23.00	ZEUS (16K ARC) 29.00	ADVENTURE TRIL (16K GRA/ADV) 29.00	CIMEEON MOON 3D (16K GRA/ADV) 29.00	CALIXTO ISLAND (32K GRA/ADV) 30.00	CIRCLE WORLD (16K ADV) 23.00																																																																																													
DEATHSHIP (16K ADV) 18.00	DERELICT (16K ADV) 23.00	EARTHQUAKE (16K ADV) 23.00	FEMBOTS REV. 3D (32K GRA/ADV) 29.00	GREYMOON (16K SIM) 18.00	HAUNTED HOUSE (16K ADV) 23.00																																																																																													
INCA TREASURE (16K ADV) 23.00	MARS (16K ADV) 23.00	COSMIC CLONES (16K ARC) 29.00	CESSNA LANDER (16K SIM) 22.00	CHOPPER STRIKE (16K ARC) 32.00	COLOR FURY (32K ARC) 32.00																																																																																													
DEMON ASSAULT (16K ARC) 29.00	DESERT PATROL (32K ARC) 29.00	DOODLE BUG (16K ARC) 31.00	EL BANAITO (16K ARC) 29.00	FOOD WAR (32K ARC) 29.00	FRYA ORACA (32K ARC) 29.00																																																																																													
GLAXONS (16K ARC) 29.00	HAYWIRE (16K ARC) 29.00	ICE MASTER (32K ARC) 29.00	INTERCEPTOR (32K ARC) 23.00	PARAMITOS ANOM. (16K ADV) 23.00	PYRAMID (16K ADV) 23.00																																																																																													
SEA QUEST (32K ADV) 29.00	SHENNANIGANS (32K GRA/ADV) 29.00	WIZARD 64 (64K ADV) 32.00	FLIPPER (16K ARC) 20.00	INTERGALACTIC FORCE (16K ARC) 30 25.00	EIGHT BALL (32K GME) 29.00																																																																																													
BLOC HEAD (16K ARC) 29.00	REDWOOD GOLF (32K GME) 29.00	TALKING FINAL ONTOWN (32K ARC) 29.00	STAGECOACH (32K GRA/ADV) 25.00	INSPECTOR CLUESEAU (32K GRA/ADV) 25.00	(Disk Only)																																																																																													
<table border="1" style="width: 100%; border-collapse: collapse; font-size: 8px;"> <thead> <tr> <th>APPLICATION PROGRAMS</th> <th>TAPE</th> <th>DISK</th> </tr> </thead> <tbody> <tr><td>AUTOTERM</td><td>46.00</td><td>-</td></tr> <tr><td>CC DATABASE/LETTERWR</td><td>52.00</td><td>57.00</td></tr> <tr><td>COLOR DFT</td><td>30.00</td><td>35.00</td></tr> <tr><td>ELITE CLAC</td><td>63.00</td><td>68.00</td></tr> <tr><td>ELITE WORD</td><td>68.00</td><td>-</td></tr> <tr><td>FILMASTER</td><td>35.00</td><td>-</td></tr> <tr><td>SCHEMATIC DRAFTING PR</td><td>-</td><td>62.00</td></tr> <tr><td>SMALL BUSINESS ACCTN.</td><td>-</td><td>150.00</td></tr> <tr><td>TELEWRITER-64</td><td>57.00</td><td>68.00</td></tr> <tr><td>TERMTALK</td><td>45.00</td><td>50.00</td></tr> <tr><td>VIP DATABASE</td><td>-</td><td>68.00</td></tr> <tr><td>VIP SPELLER</td><td>-</td><td>68.00</td></tr> <tr><td>VIP TERMINAL</td><td>52.00</td><td>57.00</td></tr> <tr><td>VIP WRITER</td><td>63.00</td><td>68.00</td></tr> <tr><td>TIMS (TAPE INFO. MAN. SYS)</td><td>\$28.00</td><td>-</td></tr> <tr> <th>UTILITIES</th> <th>TAPE</th> <th>DISK</th> </tr> <tr><td>64COL. MOD I/III EMULATOR</td><td>-</td><td>23.00</td></tr> <tr><td>64K BOOT/PAGER</td><td>17.00</td><td>-</td></tr> <tr><td>64 DISK UTIL. PACK</td><td>-</td><td>26.00</td></tr> <tr><td>DISK MANAGER</td><td>-</td><td>29.00</td></tr> <tr><td>DISK UTILITY</td><td>23.00</td><td>-</td></tr> <tr><td>HIDDEN BASIC</td><td>23.00</td><td>-</td></tr> <tr><td>MDISK</td><td>23.00</td><td>-</td></tr> <tr><td>PRITTY PRINTER</td><td>23.00</td><td>-</td></tr> <tr><td>QUICKSORT</td><td>12.00</td><td>-</td></tr> <tr><td>ROMBACK</td><td>20.00</td><td>-</td></tr> <tr><td>SUPER SCREEN</td><td>35.00</td><td>-</td></tr> <tr><td>SUPER ZAP</td><td>-</td><td>40.00</td></tr> <tr><td>TAPE UTILITY</td><td>35.00</td><td>40.00</td></tr> <tr><td>VIP DISK ZAP</td><td>-</td><td>68.00</td></tr> </tbody> </table>						APPLICATION PROGRAMS	TAPE	DISK	AUTOTERM	46.00	-	CC DATABASE/LETTERWR	52.00	57.00	COLOR DFT	30.00	35.00	ELITE CLAC	63.00	68.00	ELITE WORD	68.00	-	FILMASTER	35.00	-	SCHEMATIC DRAFTING PR	-	62.00	SMALL BUSINESS ACCTN.	-	150.00	TELEWRITER-64	57.00	68.00	TERMTALK	45.00	50.00	VIP DATABASE	-	68.00	VIP SPELLER	-	68.00	VIP TERMINAL	52.00	57.00	VIP WRITER	63.00	68.00	TIMS (TAPE INFO. MAN. SYS)	\$28.00	-	UTILITIES	TAPE	DISK	64COL. MOD I/III EMULATOR	-	23.00	64K BOOT/PAGER	17.00	-	64 DISK UTIL. PACK	-	26.00	DISK MANAGER	-	29.00	DISK UTILITY	23.00	-	HIDDEN BASIC	23.00	-	MDISK	23.00	-	PRITTY PRINTER	23.00	-	QUICKSORT	12.00	-	ROMBACK	20.00	-	SUPER SCREEN	35.00	-	SUPER ZAP	-	40.00	TAPE UTILITY	35.00	40.00	VIP DISK ZAP	-	68.00
APPLICATION PROGRAMS	TAPE	DISK																																																																																																
AUTOTERM	46.00	-																																																																																																
CC DATABASE/LETTERWR	52.00	57.00																																																																																																
COLOR DFT	30.00	35.00																																																																																																
ELITE CLAC	63.00	68.00																																																																																																
ELITE WORD	68.00	-																																																																																																
FILMASTER	35.00	-																																																																																																
SCHEMATIC DRAFTING PR	-	62.00																																																																																																
SMALL BUSINESS ACCTN.	-	150.00																																																																																																
TELEWRITER-64	57.00	68.00																																																																																																
TERMTALK	45.00	50.00																																																																																																
VIP DATABASE	-	68.00																																																																																																
VIP SPELLER	-	68.00																																																																																																
VIP TERMINAL	52.00	57.00																																																																																																
VIP WRITER	63.00	68.00																																																																																																
TIMS (TAPE INFO. MAN. SYS)	\$28.00	-																																																																																																
UTILITIES	TAPE	DISK																																																																																																
64COL. MOD I/III EMULATOR	-	23.00																																																																																																
64K BOOT/PAGER	17.00	-																																																																																																
64 DISK UTIL. PACK	-	26.00																																																																																																
DISK MANAGER	-	29.00																																																																																																
DISK UTILITY	23.00	-																																																																																																
HIDDEN BASIC	23.00	-																																																																																																
MDISK	23.00	-																																																																																																
PRITTY PRINTER	23.00	-																																																																																																
QUICKSORT	12.00	-																																																																																																
ROMBACK	20.00	-																																																																																																
SUPER SCREEN	35.00	-																																																																																																
SUPER ZAP	-	40.00																																																																																																
TAPE UTILITY	35.00	40.00																																																																																																
VIP DISK ZAP	-	68.00																																																																																																

MAIL ORDERS POSTED WITHIN 24 HOURS Make Cheque/Money Order out to COLOR BURST SOFTWARE
 Address Orders to: COLOR BURST SOFTWARE, P.O. Box 256, Roseville, NSW, 2069 OR Phone: (02) 467-1619

NORTH SHORE USERS GROUP:



Goodbye and many thanks to Eugene and Rochelle Staker. Many of us have been greatly helped over the past two years in which Eugene and Rochelle were in Australia. The call of the wild frontier of the Canadian Wild West (and cheaper hardware items for the CoCo) has seen our good friends sail homeward bound. Now for the GOOD news. The monthly USER GROUP MEETS is still on at the same time and date. First Sunday in each month. Starts at 1.00 pm. at 49 CARNARVON ROAD, ROSEVILLE. Ph: (02) 467-1619. Ken Uzzell. All Color Computer Users and interested persons are invited to attend. Feel free to bring along your COCO and TV or just come along and try out our software programmes before you buy. You are under no obligation to purchase.
 PROGRAMMING PROBLEMS - I/Output Errors on loading, Software Operation queries - there is sure to be someone there with years of experience with the COCO who would be happy to assist you and give advice.

DID YOU READ?

RAINBOW Jan '83

- ★ CREATING YOUR OWN ADVENTURE
- ★ INVITATIONS MADE EASY
- ★ MATH PAL
- ★ GET INTO THE HOBBIT OF PLAYING
- ★ LOAD BASIC PROGRAMS 'UP TOP'
- ★ CREATING 3 DIMENSIONAL GRAPHICS

IF NOT YOU ALMOST MISSED OUT!

RAINBOW Feb '83

- ★ ORGANISE YOUR TAPES
- ★ STAYIN ALIVE AT OUTPOST FIVE
- ★ HEX PAD CONNECTION
- ★ MANAGEMENT DECISION MODEL
- ★ HAUNTED HOUSE ADVENTURE
- ★ RANDOMIZE RND NUMBERS
- ★ HANG'N AROUND 'HANGMAN'
- ★ CHEMICAL BONDING SIMULATION
- ★ NON-STANDARD INTER-FACING

RAINBOW Mar '83

- ★ HALF LIFE OF NUCLEAR DECAY
- ★ TECHNIQUES FOR PLOTTING SCREEN GRAPHS
- ★ D.O.S. DETACH
- ★ DUMB TERMINAL ROUTINE
- ★ VARIABLES TUTORIAL
- ★ THE WORD - LEARN YOUR GRAMMAR
- ★ DEACTIVATE THE BOMB
- ★ TWO ILLUSIONS

Get 'em while they last cos there's still a few copies left

NORTH QLD. COLOUR SOFTWARE

9 DURHAM COURT, KIRWAN, TOWNSVILLE 4814.

WE'RE USER FRIENDLY (077)732064

MEMORY 64K UPGRADES

\$89 ONLY

(32K) EIGHT BALL EIGHTBALL EIGHT BALL \$28-95

* TOTALLY REALISTIC*CONTROL THE CUE*
 *HIT HARD,SOFT,MEDIUM*SCREW BACK,RUN ON*
 SPIN & SWERVE /click/CLICK/go the balls!!!

so realistic you'll be looking for the chalk! SORRY! NO SMOKING OVER THE TABLE!

"Your turn EDDIE!"

if not already on 64K, take a 5% discount on upgrade/eightball package.

whirlybird-5 screens-guide your chopper on a mission- if you have the nerve! \$2895

astroblast-defend your home base-try to refuel \$2895

ghost gobbler- how's your blood pressure?-colourful PACMAN type.\$2895

FILEMASTER-powerful database-speedy filing and sorting/run to prntr.\$3420

32K*32K*32K* VIP WRITER \$6295/VIP DATABASE

DONKEY-what a classic! \$2995

CASHMAN-40-yes,40screens.

Two players on screen at same time.Beat your opponent to the MONEY.

SPACE SHUTTLE-great! Realistic simulation. retrieve satellite, land on 3D runway.

 WRITE OR PHONE FOR MORE INFORMATION

64K UPGRADES
 ONLY \$89
 . /KIT

```

1260 PRINT"R) REDUCE YOUR LOAN B
ALANCE"
1270 PRINT"M) GO BACK TO MAIN ME
NU","Q) QUIT":PRINT:PRINT"YOUR C
HOICE?"
1280 R$=INKEY$:IFR$="" THEN1280
1290 IFR$="R" AND LB(T)>0 THEN 2
290
1300 IFR$="S" THEN1430
1310 IFR$="M" THEN 1160
1320 IFR$="T" THEN1610
1330 IFR$="A" THEN1830
1340 IFR$="L" THEN2100
1350 IFR$="Q" THEN2790ELSE1280
1360 IFA2(S)=0 THEN PRINT"YOU CA
N'T BUY "A$(S)!"":GOTO1150
1370 IF M(T)>F(S)+(F(S)*.1*H(S))
THEN 1390
1380 PRINT"SORRY, YOU DON'T HAVE
ENOUGH","CASH TO BUY IT!":GOTO1
150
1390 IFA1(S)=T THEN PRINT"YOU AL
READY OWN IT!":GOTO1150
1400 IFA1(S)<>0 THEN PRINTP$(A1(
S))" OWNS IT!":GOTO1150
1410 PRINT"TITLE DEED RECORDED"
1420 M(T)=M(T)-(F(S)+(F(S)*.1*H(
S))):A1(S)=T:GOTO1600
1430 CLS:GOSUB3140:INPUT"LOT NO.
YOU'RE SELLING";S
1440 IFS<>0 THEN 1470
1450 CLS:GOSUB3190:INPUT" LOT NO
. YOU'RE SELLING";S
1460 IFS=0 THEN 1150
1470 IFS<0 OR S>32 THEN1490
1480 IFA1(S)=T AND A2(S)<>0 THEN
1500
1490 PRINT"SUPER BOO-BOO! TRY AG
AIN!":GOTO1150
1500 CLS:GOSUB3230
1510 INPUT"# OF PLAYER YOU'RE SE
LLING TO";Y
1520 IFY=T ORY<1 OR Y>P THEN PRI
NT"OOPS!":GOTO1150
1530 INPUT"HOW MUCH ARE YOU GETT
ING";XX:XX=INT(XX)
1540 PRINT"SELLING "A$(S)" TO "P
$(Y),"FOR $"XX", CORRECT?"
1550 R$=INKEY$:IFR$="" THEN 1550
1560 IFR$<>"Y" THEN PRINT"HUMANS
!":GOTO1150
1570 IFXX>M(Y) THEN PRINT"WRONG-
HE HASN'T ENOUGH $$$!":GOTO1150
1580 M(T)=M(T)+XX:M(Y)=M(Y)-XX:A
1(S)=Y
1590 PRINT"TRANSACTION COMPLETE.
":S=P(T)
1600 PRINT"YOUR CASH IS NOW $"M(
T):GOTO1150
1610 CLS:PRINT@32,"YOU MAY ONLY
TRADE 1 FOR 1.":PRINT
1620 PRINT"(IF YOU'RE TRADING 2
OR MORE,","SELL' THOSE LOTS)"
1630 GOSUB2960:GOSUB2960:CLS:GOS
UB3230
1640 INPUT"# OF PLAYER YOU'RE TR
ADING WITH";Y
1650 IFY<1ORY>P ORY=T THEN PRINT
"OOPS!":GOTO1150
1660 CLS:GOSUB3140:INPUT"YOUR LO
T NO. (IF ANY)";S
1670 IFS<>0 THEN1690
1680 CLS:GOSUB3190:INPUT"YOUR LO
T NO. (IF ANY)";S
1690 IFS<=0 ORS>32 THEN 1490
1700 IF A2(S)=0 THENPRINT"CAN'T
TRADE "A$(S):GOTO1150
1710 IFA1(S)<>T THENPRINT"I NEED
YOUR LOT NO.":GOTO1660
1720 CLS:PRINT"NOW CHOOSE "P$(Y)
"'S LOT:"
1730 GOSUB3140:INPUT"LOT #";SS: I
F SS<>0 THEN 1750
1740 CLS:GOSUB3190:INPUT"LOT #";
SS
1750 IFSS<=0 OR SS>32 THEN1490
1760 IFA1(SS)<>Y THEN1490
1770 CLS:PRINT"IF YOU ARE ALSO R
ECEIVING CASH,","INPUT THAT NOW.
"
1780 PRINT"IF YOU PAY,INPUT A NE
GATIVE","FIGURE. (IF NO CASH IS
INVOLVED"
1790 PRINT"JUST HIT 'ENTER')."
1800 INPUTYY:IF YY<M(Y) THEN 182
0
1810 PRINT"DEAL'S OFF- NOT ENoug
H CASH!":GOTO1150
1820 M(T)=M(T)+YY:M(Y)=M(Y)-YY:A
1(S)=Y:A1(SS)=T:GOTO1590
1830 CLS:GOSUB3140:INPUT"LOT NO.
(IF NONE HIT 'ENTER')";S
1840 IFS<>0 THEN 1860
1850 CLS:GOSUB3190:INPUT"LOT NO.
(IF NONE, HIT 'ENTER')";S
1860 IFS<=0 ORS>32 THEN 1490
1870 IFA1(S)<>T THEN1490
1880 IFH(S)>0 THEN 1920
1890 XX=0:FORX=1TO32:IFA2(X)=A2(
S) AND A1(X)=T THEN XX=XX+1
1900 NEXT:IFXX=3 THEN1920
1910 PRINT"SORRY, YOU DON'T OWN
ALL OF","THAT GROUP OF LOTS!":GO
TO1140
1920 CLS(5):PRINT:PRINTA$(S)" HA
S"H(S)"APARTMENTS"
1930 PRINT"ON IT NOW. EACH APT.
IS $"INT(F(S)*.1),"APIECE.":PRIN
T
1940 PRINT"(YOUR CASH IS $"M(T)"

```

```

) "
1950 INPUT "HOW MANY DO YOU WISH
TO ADD"; XX
1960 IF XX <= 0 THEN 1150
1970 IF (XX * F(S) * .1) < M(T) THEN 199
0
1980 PRINT "SORRY, YOU ONLY HAVE
THE CASH", "TO BUILD" INT(M(T) / (F(
S) * .1)): GOTO 1140
1990 IF H(S) + XX <= 50 THEN 2010
2000 PRINT "SORRY, NO MORE THAN 5
0 APTS.", "PER LOT ALLOWED.": GOTO
1150
2010 PRINT XX "UNITS AT $" INT(F(S)
*.1) "IS $" F(S) * XX *.1
2020 PRINT "IS THAT OK WITH YOU?"
2030 R$ = INKEY$: IFR$ = "" THEN 2030
2040 IFR$ <> "Y" THEN 1160
2050 H(S) = H(S) + XX: M(T) = M(T) - (F(S)
) * XX *.1
2060 PRINT "OK, THEY'RE BUILT."
2070 PRINT "WANT TO BUILD SOME MO
RE?"
2080 R$ = INKEY$: IFR$ = "" THEN 2080
2090 IFR$ = "Y" THEN 1830 ELSE 1590
2100 GOSUB 3340: CLS(5): XX = XX *.5: P
RINT: IF XX > 10000 THEN XX = 10000
2110 PRINT: PRINT "YOUR CREDIT LIM
IT IS $" XX - LB(T): PRINT
2120 INPUT "HOW MUCH DO YOU WANT
TO BORROW"; Y
2130 IF Y > XX - LB(T) THEN 2360
2140 PRINT "LOAN TERMS:"
2150 PRINT "LOANS ARE FOR 10 TRIP
S AROUND", "THE BOARD. NEW LOANS
ARE CON-"
2160 PRINT "SOLIDATED WITH EXISTI
NG LOANS."
2170 PRINT "(SIMPLE INTEREST IS U
SED, THE", "RATE NOW IS" I "%.)"
2180 GOSUB 2960: GOSUB 2970: CLS(5)
2190 PRINT: PRINT "LOAN BALANCE NO
W IS $" LB(T)
2200 PRINT "(INTEREST CHARGES ARE
$" Y * I / 100)"
2210 PRINT "NEW BALANCE WILL BE $
" LB(T) + Y + (Y * I / 100)
2220 PRINT "NEW PAYMENTS: $" (LB(T)
+ Y + (Y * I / 100)) / 10
2230 PRINT: PRINT "IS THIS OK WITH
YOU?"
2240 R$ = INKEY$: IFR$ = "" THEN 2240
2250 IFR$ <> "Y" THEN 1160
2260 LB(T) = LB(T) + Y + (Y * I / 100): LP(
T) = LB(T) / 10
2270 LB(T) = INT(LB(T)): LP(T) = INT(
LP(T)): Y = INT(Y)
2280 M(T) = M(T) + Y: GOTO 1590
2290 CLS: PRINT @ 32, "YOUR LOAN BAL
ANCE IS $" LB(T): PRINT "YOUR CASH
IS $" M(T)
2300 : PRINT: INPUT "HOW MUCH WOULD
YOU LIKE TO PAY "; X
2310 IF X = 0 THEN 1160
2320 IF X < 0 OR X > M(T) OR X > LB(T) TH
EN PRINT "OOPS!!": GOTO 1160
2330 M(T) = M(T) - X: LB(T) = LB(T) - X
2340 IFLB(T) <= 5 THEN LB(T) = 0: LP(
T) = 0
2350 GOTO 1590
2360 PRINT "SORRY, YOUR LOAN IS D
ENIED DUE", "TO LACK OF ASSETS.":
GOTO 1150
2370 CLS: PRINT: PRINT P$(T)", "YOU'
RE ON "A$(S)
2380 PRINT "YOUR CASH IS $" M(T)
2390 PRINT "YOUR LOAN BALANCE IS
$" LB(T)
2400 IFLB(T) <= 0 THEN 2430
2410 PRINT "LOAN PAYMENTS ARE $" L
P(T)
2420 PRINT "PAYMENTS LEFT =" INT(L
B(T) / LP(T))
2430 GOSUB 3340: PRINT "CREDIT AVAI
LABLE IS $";
2435 XX = XX *.5
2440 IF XX > 10000 THEN XX = 10000
2450 IF XX - LB(T) < 0 THEN PRINT "0" E
LSE PRINT XX - LB(T)
2460 PRINT: PRINT "WOULD YOU LIKE
TO SEE A RUNDOWN", "OF ALL THE PR
OPERTIES?"
2470 R$ = INKEY$: IFR$ = "" THEN 2470
2480 IFR$ <> "Y" THEN 1160
2490 CLS: GOSUB 3140: PRINT "HIT ANY
KEY FOR THE REST."
2500 R$ = INKEY$: IFR$ = "" THEN 2500
2510 CLS: GOSUB 3190: PRINT "REPEAT
DISPLAY?"
2520 R$ = INKEY$: IFR$ = "" THEN 2520
2530 IFR$ = "Y" THEN 2490 ELSE 1160
2540 IF M(T) > 0 THEN H1(T) = 0: GOTO 2
830
2550 CLS: PRINT @ 96, "OH, OH! YOU'RE
BROKE! (" M(T) )"
2560 PRINT "(TIME NO. " H1(T) "!!)"
2570 IF H1(T) = 3 THEN PRINT "LAST T
IME!"
2580 PRINT "YOU HAVE SEVERAL OPTI
ONS:"
2590 PRINT "L) GET A LOAN"
2600 PRINT "S) SELL SOME PROPERTY
"
2610 IF H1(T) <= 3 THEN PRINT "I) G
O 'IN-THE-HOLE'"
2620 PRINT "R) LET THE BANK REPOS
SES ENOUGH", "PROPERTY TO GET
YOU AHEAD"
2630 PRINT " (AT 1/2 IT'S VALUE
)"

```

```

2640 PRINT"Q) (GULP!) QUIT!"
2650 PRINT"WHAT DO YOU WANT TO D
0?"
2660 R$=INKEY$:IFR$="" THEN 2660
2670 IFR$="S" THEN 1430
2680 IFR$="R" THEN 2730
2690 IFR$="L" THEN 2100
2700 IFR$="Q" THEN 2790
2710 IFR$="I" AND H1(T)<4 THEN 2
720 ELSE 2660
2720 P(T)=9:GOTO 410
2730 CLS:PRINT:FORX=1 TO 32:IFM(T)
>0 OR A1(X)<>T THEN 2760
2740 M(T)=M(T)+(F(X)+(H(X)*F(X)*
.1))/2:A1(X)=0
2750 PRINT"REPOSSED "A$(X)" W/"H
(X)"APTS."
2760 NEXT X
2770 PRINT:PRINT"YOU RECEIVED 1/
2 THE VALUE OF","THE ABOVE PROPE
RTIES."
2780 GOSUB 2960:GOSUB 2960:GOTO 159
0
2790 CLS:PRINT:PRINT"WELL, YOU T
RIED ANYWAY!":Q(T)=1:GOSUB 2970
2800 FORX=1 TO 32:IFA1(X)=T THEN M
(T)=M(T)+(F(X)+(H(X)*(F(X)*.1)))
2810 IFA1(X)=T THEN A1(X)=0
2820 NEXT X
2830 XX=0:FORX=1 TO P:IFQ(X)>0 THE
NXX=XX+1
2840 NEXT X
2850 IFXX=>P-1 THEN 2870
2860 GOSUB 3340:IFM(T)+XX-LB(T)>L
1 THEN 2870 ELSE 410
2870 CLS:PRINT@64," END OF GAME!
"
2880 PRINT"HERE'S THE STANDINGS
IN TOTAL","ASSETS, LESS ANY LOAN
S:":PRINT
2890 FORX=1 TO 32
2900 IFA1(X)>0 THEN M(A1(X))=M(A
1(X))+F(X)+(H(X)*F(X)*.1)
2910 NEXT X
2920 Z=1:FORX=1 TO P:PRINTP$(X)":$
"M(X)-LB(X):M(X)=M(X)-LB(X)
2930 IFM(X)>M(Z) THEN Z=X
2940 NEXT X:PRINT:PRINT"
"P$(Z)
" WON!!!!!"
2950 PRINT:PRINT:PRINT"HOPE YOU
HAD FUN!":END
2960 FORZ=1 TO 2000:NEXT
2970 FORZ=1 TO 2000:NEXT:SOUND 190,
1:RETURN
2980 CLS:FORZ=1 TO 25:SOUNDZ,1:PRI
NT@RND(400),"$:NEXT
2990 PRINT@107,"PAYDAY!"
3000 PRINT:PRINT@224,"YOU RECEIV
E A PAYCHECK EQUAL","TO 10% OF Y
OU HOLDINGS OR"
3010 PRINT"$2000, WHICH EVER IS

```

```

HIGHER.":GOSUB 3340
3020 XX=INT(XX*.1):IFXX<2000 THE
N XX=2000
3030 PRINT:PRINT"YOUR PAYCHECK I
S $"X
3035 PRINT"YOU NOW HAVE $"M(T)+X
X
3040 M(T)=M(T)+XX:GOSUB 2960:RETU
RN
3050 X=RND(40)
3060 PRINT"NEWS FLASH:"
3070 FORZ=1 TO 7:SOUND 200,2:NEXT
3080 PRINT"ECONOMIC INDEX CHANGE
OF";
3090 IFRND(20)>11 THEN X=-X
3100 PRINTINT(X*2):I=I+(X*.1):I=
INT(I)
3110 IFI<5 THEN I=5
3120 PRINT"LOAN INTEREST IS NOW"
I"%!"
3130 GOSUB 2960:CLS:RETURN
3140 PRINT"LOT GROUP NAME
OWNER APTS"
3150 PRINT
3160 FORX=1 TO 16:IFA2(X)=0 THEN 31
80
3170 PRINTX"-";TAB(5);A2(X);TAB(
6);A$(X);TAB(21);P$(A1(X));TAB(2
7);H(X)
3180 NEXT X:SOUND 180,1:RETURN
3190 PRINT"LOT GROUP NAME
OWNER APTS"
3200 FORX=17 TO 32:IFA2(X)=0 THEN 3
220
3210 PRINTX"-";TAB(5);A2(X);TAB(
6);A$(X);TAB(21);P$(A1(X));TAB(2
7);H(X)
3220 NEXT X:SOUND 180,1:RETURN
3230 FORX=1 TO P:PRINTX"-P$(X):NE
XT:RETURN
3240 LB(T)=LB(T)-LP(T):IFLB(T)<5
THEN LB(T)=0:LP(T)=0
3250 M(T)=M(T)-LP(T)
3260 SOUND 32,10:SOUND 32,10:SOUND
32,5:SOUND 32,10:SOUND 69,10:SOUND
58,5:SOUND 58,10
3270 SOUND 32,5:SOUND 32,10:SOUND 1
9,5:SOUND 32,10
3280 PRINT:PRINT"PAYMENT DUE:$"L
P(T)
3290 PRINT"NEW BALANCE=$"LB(T)
3300 IFLB(T)<=0 OR LP(T)<=0 THEN
3330
3310 PRINT"LOAN PAYMENTS LEFT:"I
NT(LB(T)/LP(T))
3320 PRINT"CASH IS NOW $"M(T):GO
SUB 2960
3330 GOSUB 2960:CLS:RETURN
3340 XX=0:FORX=1 TO 32:IFA1(X)=T T
HEN XX=XX+F(X)+(H(X)*F(X)*.1)
3350 NEXT X:XX=INT(XX):RETURN

```

COCO GRAPHICS

Exploring The Angles Of BASIC And LOGO

By Don Inman
RAINBOW Contributing Editor

LOGO and BASIC were created for very different purposes. Therefore, comparisons, such as I have made in the May and June issues of THE RAINBOW, mean little in determining which is the best language. I have not been trying to point out the superiority of one or the other.

When you are learning something new, it is quite often helpful to relate it to experiences that you have had in the past. BASIC has been around for a long time. LOGO, the new kid on the block, retains some of the features of BASIC but also has its own features. The purpose of this series of articles is to introduce some of these features using BASIC as a reference.

The rectangle is used in many ways when creating a graphics display. This month's article explores BASIC and LOGO graphics using the rectangle for comparison.

A BASIC Rectangle

There are several ways to draw a rectangle using BASIC. You may "turn on" each point with *PSET* commands:

```

10 PMODE 3: PCLS: SCREEN 1,0
20 Y=10
30 FOR X=10 TO 60
40 PSET (X,Y): PSET (X,Y+20)
50 NEXT X
60 X=10
70 FOR Y=10 TO 20
80 PSET (X,Y): PSET (X+50,Y)
90 NEXT Y
100 GOTO 100
    
```

← set Y
← top & bottom
← set X
← sides

You may draw a rectangle using the *DRAW* command:

```

10 PMODE 3: PCLS: SCREEN 1,0
20 DRAW"BM 10,10; R50 D10 L50 U10"
30 GOTO 30
    
```

↑ start ↑ right ↑ down ↑ left ↑ up

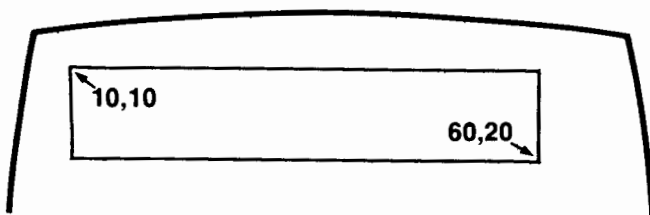
You may also use the *LINE* command with the box (B) option:

```

10 PMODE 3: PCLS: SCREEN 1,0
20 LINE (10,10)-(60,20),PSET,B
30 GOTO 30
    
```

↑ upper left ↑ lower right ↑ make a box

All three of these methods draw the same rectangle.



A LOGO Rectangle

The turtle drawings of LOGO most closely resemble the BASIC method that uses the *DRAW* command. A turtle procedure that draws a similar rectangle could be:

```

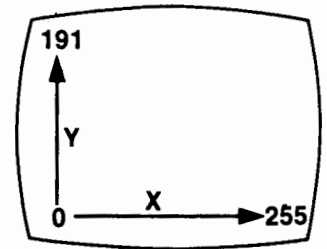
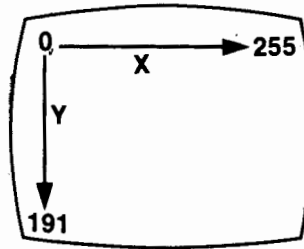
TO RECTANGLE
  CLEAR PU HT
  SX 10 SY 180
  RT 90 PD
  REPEAT 2(FD 50 RT 90 FD 10 RT 90)
END
    
```

← lift pen, hide turtle
← set X and Y
← turn right, lower pen
← draw rectangle

The resulting rectangle would look like those drawn in BASIC. However, note that the LOGO screen begins with Y = 0 at the bottom of the screen.

BASIC SCREEN

LOGO SCREEN



Color-filled Rectangles

Rectangles can be filled with color by Extended Color BASIC very easily by using the Fill option with the *LINE* command or by using the *PAINT* command in conjunction with any rectangle drawing method. LOGO does not have any easy way to fill an enclosed figure with color. However, it can be done by coloring each line inside the rectangle.

```

BASIC:
20 LINE(10,10) — (60,20),PSET,BF
   or
20 DRAW"BM10,10;R50D10L50U10"
30 PAINT(15,15),4,4
    
```

```

LOGO:
PC 2 SX 10 SY 180
REPEAT 4(FD 50 RT 90
  FD 1 RT 90 FD 50 LT 90
  FD 1 LT 90)
FD 50
    
```

BASIC and LOGO produce similar rectangles. However, the colors produced are not the same.

A Practical Program

The following BASIC program and LOGO procedures show the use of rectangles in producing a bar graph. Notice that BASIC uses subroutines in a similar way that LOGO uses subprocedures. The main program, or procedure, in each language is written as a series of subroutines, or subprograms, so that you can easily compare how the two languages produce similar results for each part of the program.

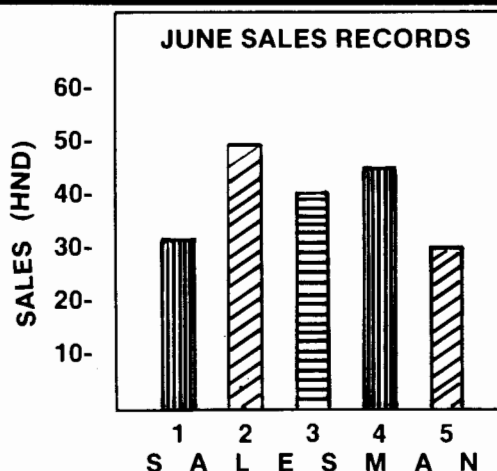
BAR GRAPH

BASIC PROGRAM	LOGO PROGRAM
<pre> 100 REM * MAIN PROGRAM * 110 PMODE 3: PCLS: SCREEN 1,0 120 CLEAR 1000: DIM L\$(21) 130 GOSUB 1000 140 GOSUB 2000 150 GOSUB 3000 'DRAW BOTTOM 160 GOSUB 4000 'DRAW TITLE 170 GOSUB 5000 'DRAW SIDES 180 GOSUB 6000 'DRAW BARS 190 GOTO 190 </pre>	<pre> TO GRAPH CLEAR HT RECT BOTTOM TITLE SIDES BARS END </pre>
<pre> 1000 REM * ASSIGN LETTERS * 1010 L\$(1)="R8U4L8U4R8BD8BR8" 'S OR 5 1020 L\$(2)="U4NR8U4R8D8BR8" 'A 1030 L\$(3)="NU8R8BR8" 'L 1040 L\$(4)="NR8U4NR8U4R8BD8BR8" 'E 1050 L\$(5)="U8F4E4D8BR8" 'M 1060 L\$(6)="U8F8NU8BR8" 'N 1070 L\$(7)="U8R8D4L4NL4F4BR8" 'R 1080 L\$(8)="NR8U8R8BD8BR8" 'C 1090 L\$(9)="NR8U8R8D8BR8" 'O OR 0 1100 L\$(10)="NR6U8R8F2D4G2BR10" 'D 1110 L\$(11)="NU2R8NU8BR8" 'J 1120 L\$(12)="NU8R8NU8BR8" 'U 1130 L\$(13)="U4NU4R8NU4D4BR8" 'H 1140 L\$(14)="BR4NU8BR12" '1 1150 L\$(15)="BU8R6D2G6R8BR8" '2 1160 L\$(16)="NR8BU4NR8BU4R8D8BR8" '3 1170 L\$(17)="BU4NU4R4NU4ND4R4BD4 BR8" '4 1180 L\$(18)="NR8U8R8BD4NL8D4BR8" '6 1190 L\$(19)="BU4NE4F4BR12" '< 1200 L\$(20)="BR4BU8F4G4BR12" '> 1210 L\$(21)="BU4R8BD4BR8" '- 1220 RETURN </pre>	<pre> LOGO can mix text and graphics. Therefore, no LOGO commands here. </pre>
<pre> 2000 REM * DRAW RECTANGLE * 2010 LINE(80,148)-(249,10),PSET, B 2020 RETURN </pre>	<pre> TO RECT SX 80 SY 42 REPEAT 2(FD 138 RT 90 FD 166 RT 90) END </pre>
<pre> 3000 REM * BOTTOM * 3010 DRAW"BM96,160;S2"+L\$(14)+"B R48"+L\$(15)+"BR48"+L\$(16)+"BR48" +L\$(17)+"BR48"+L\$(1) 3020 DRAW"BM96,172;"+L\$(1)+"BR22 "+L\$(2)+"BR22"+L\$(3)+"BR22"+L\$(4) +"BR22"+L\$(1)+"BR22"+L\$(5)+"BR2 2"+L\$(2)+"BR22"+L\$(6) 3030 RETURN </pre>	<pre> TO BOTTOM SX 99 SY 32 SH 90 PRINT"1 2 3 4 5" SX 10:8 SY 20 PRINT"S A L E S M A N" END </pre>

<pre> 4000 REM * TITLE * 4010 DRAW"BM96,22;" +L\$(11)+L\$(12)) +L\$(6)+L\$(4)+"BR20"+L\$(1)+L\$(2) +L\$(3)+L\$(4)+L\$(1)+"BR20" 4020 DRAW L\$(7)+L\$(4)+L\$(8)+L\$(9))+L\$(7)+L\$(10) 4030 RETURN </pre>	<pre> TO TITLE SX 94 SY 164 PRINT"JUNE SALES RECORDS" END </pre>
<pre> 5000 REM * SIDES * 5010 DRAW"BM56,132;" +L\$(14)+L\$(9))+L\$(21) 5020 DRAW"BM56,116;" +L\$(15)+L\$(9))+L\$(21) 5030 DRAW"BM56,100;" +L\$(16)+L\$(9))+L\$(21) 5040 DRAW"BM56,84;" +L\$(17)+L\$(9))+L\$(21) 5050 DRAW"BM56,68;" +L\$(1)+L\$(9)+ L\$(21) 5060 DRAW"BM56,52;" +L\$(18)+L\$(9))+L\$(21) 5070 DRAW"BMB,80;" +L\$(1)+L\$(2)+L \$(3)+L\$(4)+L\$(1) 5080 DRAW"BMB,92;" +L\$(19)+L\$(13))+L\$(6)+L\$(10)+L\$(20) 5090 RETURN </pre>	<pre> TO SIDES SX 56 SY 60 PRINT"10-" SY 76 PRINT"20-" SY 92 PRINT"30-" SY 108 PRINT"40-" SY 124 PRINT"50-" SY 140 PRINT"60-" SX 8 SY 120 PRINT"SALES" SY 108 PRINT"(HND)" END </pre>
<pre> 6000 REM * BARS * 6010 COLOR 2,1 6020 LINE(94,147)-(102,92),PSET, BF 6030 COLOR 3,1 6040 LINE(124,147)-(132,72),PSET ,BF 6050 COLOR 4,1 6060 LINE(158,147)-(166,83),PSET ,BF 6070 COLOR 2,1 6080 LINE(188,147)-(196,80),PSET ,BF 6090 COLOR 3,1 6100 LINE(218,147)-(226,102),PSE T,BF 6110 RETURN </pre>	<pre> TO BARS SX 94 SY 42 SH 0 PC 1 MAKE :F 52 RPT SX 125 SY 42 PC 2 MAKE :F 76 RPT SX 156 SY 42 PC 3 MAKE :F 66 RPT SX 190 SY 42 PC 1 MAKE :F 70 RPT SX 220 SY 42 PC 2 MAKE :F 50 RPT END TO RPT REPEAT 6(FD :F RT 90 FD 1 RT 90 FD :F LT 90 FD 1 LT 90) FD :F END </pre>

A summary of LOGO abbreviations used:

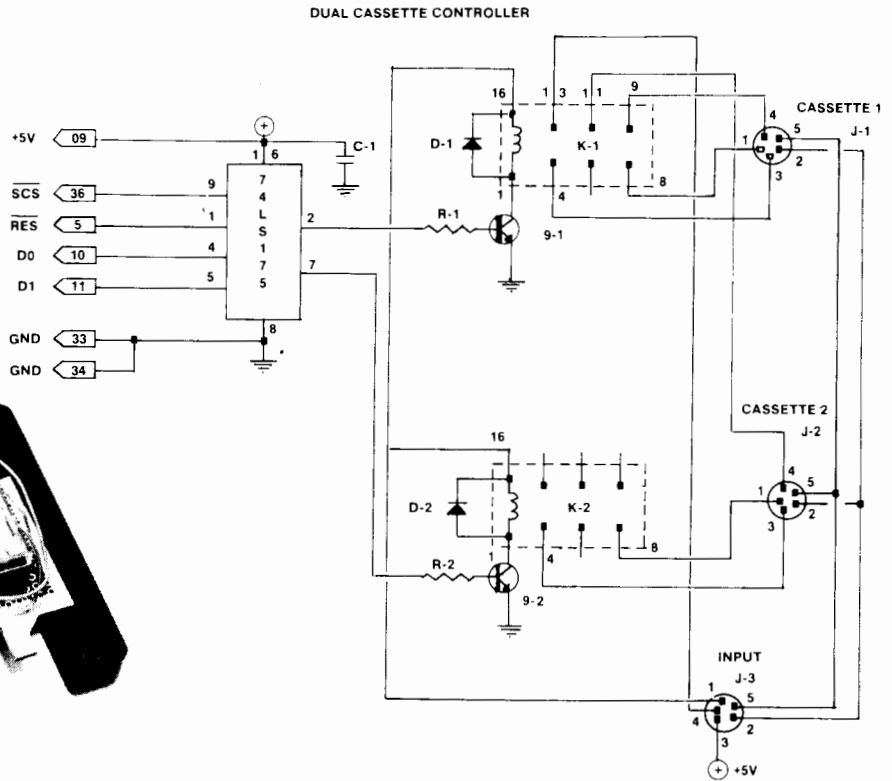
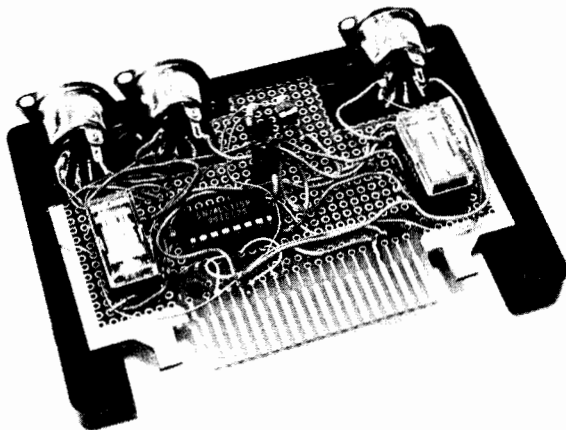
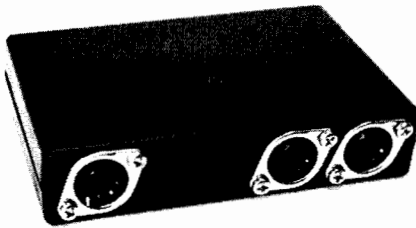
- HT = HIDE TURTLE
 - SX = SET X (coordinate of turtle)
 - SY = SET Y (coordinate of turtle)
 - FD = FORWARD (move)
 - RT = RIGHT (turn)
 - SH = SET HEADING (of turtle)
 - PC = PEN COLOR
 - LT = LEFT (turn)
 - :F = a variable
 - RPT = a subprocedure called by BARS
- NOTE: The heading must be set to 90 for *PRINT* commands in LOGO in order to print from left to right. (See the *BOTTOM* procedure)



TURN OF THE SCREW

Dualing Cassettes

By Tony DiStefano



I got the idea for this month's article from someone who gave me a call on a Monday night. He was working on a project that would control the motors of two cassette players and was having some problems with it. We spoke for a while, but I could not figure out what his problem was over the phone. I told him that I would put together one and present it in one of my articles. There is one thing — I cannot for the life of me remember his name. You know who you are, so give me a call and I'll give you credit for this idea.

First we must describe what this project is and what it does. It is what I call a Dual Cassette Controller, which fits in a small ROM pack, and plugs into the CoCo or CoCo 2 expansion port. It has three DIN connectors. One plug fits into your cassette connector in the back of the computer. The other two connectors connect to two tape recorders. This Dual Cassette Controller will enable the user to transfer files from one cassette to another. This could be useful in making backup copies of your software a lot easier than with one cassette. With the proper software, it could allow you to make complete backups of everything on one cassette to another. It could also be useful when sorting or changing ASCII text files. An example would be if you have a telephone list, and someone changed his or her address or

telephone number, it would be easier with two cassette recorders to update the file. The next few paragraphs will show you how to build and operate the Dual Cassette Controller.

The first thing to do in this project is to get the parts and tools necessary to construct the Controller. You will find a parts list later on in this article. The tools you will need this time are the "standard tool kit," drill, round file and a sharp knife.

This project is just as much electronic as it is mechanic. It involves cutting, drilling and filing things into shape. It is up to you to make it as nice as you can or want. Halfway into building it I thought of mounting the whole thing inside the computer. Then I thought there are always many ways of modifying your computer to suit your needs. Do it the way you please. I included a few photos to show you how I built my proto-type. You can do it the same way or come up with your own design. However the mechanics are done, the electronics are the same.

Following the schematic, solder all the components together. If you want the thing to fit in a ROM pack case, place the components as shown in the photos. Also, do not use sockets for the relays, it won't fit in the case. From past

experiences, there seems to be a difference in Radio Shack part numbers in Canada and the U.S. Some numbers do not always match, so be careful. When you are not sure, use the description to get the part. Use at least a 24-gauge wire for the connections to and from the relays that connect to the motor connections on all the connectors. There are no surprises in the circuit, it is quite simple, only the regular care for static sensitive IC's will do. Remember to clean the PCB when you are finished.

Mount the three connectors on the end of the case. Drill, cut and file the plastic case until they fit. Then cut the PCB until it fits in the case. Make sure that no wires touch together and all solder joints are solid. Use my photos as a guide.

To try out the controller, follow these simple steps. Turn off the computer. Plug the controller into the computer slot. Plug one end of the DIN to DIN wire into the computer's cassette port. Connect the other end into the controller's input and connect the two cassette recorders into the proper connectors on the controller. Next, turn on the computer. In order to test the relays, type this in:

MOTOR ON ENTER

The internal relay should click on.

POKE 65344,1 ENTER

Relay number 1 should click on.

POKE 65344,0 ENTER

Relay number 1 should click off.

POKE 65344,2 ENTER

Relay number 2 should click on.

POKE 65344,3 ENTER

Both relays should be on. If all this works then the relays work okay. Now try to *CSAVE* and *CLOAD* to each cassette. To access the first cassette you must first:

POKE 65344,1 ENTER

Then all I/O will be through cassette number one. If you want to access cassette number two you must first:

POKE 65344,2 ENTER

That will give you access to the second cassette. *CSAVEs* and *CLOADs* will be done through this cassette. There is one more interesting thing with this controller. If you *POKE 65344,3* and ENTER, you will be able to *CSAVE* to both cassettes. Since both motors are on and the output goes to both recorders, you will get two copies of whatever you *CSAVEd* or *CSAVEMd*. This will not, however work with *CLOADs* because the inputs are switched. With some good machine language code, a user could open two cassette files say, *OPEN "O", #3, "FILENAME"*. If you want to know where I got that proto-board and case, it was from Micro R.G.S. It is a great proto-board and suits CoCo projects quite well.

Parts List

ID #	Description	RS Part #
U1	74LS175	N/A
R1,R2	470 ohm 1/4w	271-1317
J1,J2,J3	5-Pin DIN Female	274-005
Q1,Q2	2N3904	276-2016
K1,K2	5V Relay DPDT	271-215
D1,D2	1N4004	276-1103
C1	.1uf 10V	272-111
MISC	Proto-board	N/A
	Case	N/A
	16-Pin Socket	276-1998
	5-Pin to 5-Pin wire	42-2151

High-Speed POKE

• *How does the fast POKE work? I recently purchased a CoCo 2 and am a subscriber to THE RAINBOW. I have seen many articles which mention POKE 65495,0 and have found it does speed up program execution time. My question is, if 65495 is located in ROM, why should a POKE to that location make any difference? In addition, after POKE 65495,0, I make a PEEK (65495) and the value has not changed!*

By the way, your column is very informative and professionally done. Keep up the good work!

*Eliot Weinman
Brookline, MA*

As a matter of record, Eliot, I do not recommend using the high-speed *POKE* in programs. The same results can usually be accomplished by more efficient programming.

The fast speed *POKE* is actually accomplished by setting a register in the SAM chip at addresses 65494 and 65495. This register determines the division of the master clock frequency before it reaches the microprocessor chip. Since the master clock frequency of the CoCo is 14,31818 MHz, if you divide by 16 the result is .895 MHz, or the normal operating frequency. If you divide by eight, you get 1.8 MHz, or the speed that is used for the high-speed *POKE*.

The reason you can *POKE* these addresses is that they are registers within the SAM chip for setting various memory and mode parameters. *POKE*ing even addresses clears the registers, and odd addresses sets the registers. They are write only registers, hence, you will get nothing if you *PEEK* them. For a full explanation of their functions obtain a copy of the TRS-80 Color Computer Technical Reference Manual, Cat. No. 26-3193, from your local dealer.

Making The Upgrade

• *I have an "E" board 4164 RAM chips, a drive 0 with 40K and Print Spooler.*

*After performing the upgrade, I've found:
A) 40K and Print Spooler lock up the CoCo.*

B) Diagnostic ROMPAC indicates 32K RAM.

C) Your "ROMRAM" program (March '84, Page 289) and FFDF POKE lock up the CoCo with drive 0 hooked up, but work fine in Extended BASIC.

D) Rob Rosen indicates (Sept. '82, Page 59) that other modifications are necessary to access the 64K, namely:

1) Pin 5 (ground) of 74LS138 to pin 4 of the unused gate (74LS02)

2) Pin 6 of that gate to pin 8

3) Pin 5 of that gate to test pin 1

I assume all of the above 5 pins must be removed from the socket and bent upward. My

questions are:

A) If my above assumption is correct, how does the 74LS138 maintain its ground return path?

B) What effect does removing Pin 8 of the existing gate have on the RAM address line between the SAM chip and decoder?

C) Will the above modifications actually give me 64K capability by software access?

D) Do I bend the pins upward?

*Gary Curto
San Rafael, CA*

Gary, your 64K chips are not enabled until you make the upgrade you mention. There are two problems with your description of the modification.

Pin 8 of the 74LS02 does not get bent upwards. The connection to this pin has to be tack-soldered.

Pin 5 of the 74LS138 is not the ground, but the G2A chip enable input. The ground on a 74LS138 is Pin 8.

The reason you bend the pins upward is that if they were sideways they would touch the shield. I put a piece of tape between these pins and the case, just in case.

For a summary of all upgrades see "RAM/ROM Upgrade Roundup" by Ed Ellers in May '84 RAINBOW.

As soon as you make the modifications all the programs you mentioned should work without a hitch.

RAINBOWTECH

RANDOM BASICS

Operating Systems — Another Point Of View

By Paul Searby

With apologies to Brian Dougan

(Paul Searby has been involved with computers and data processing since 1969, working primarily on larger IBM systems until 1975 when he bought, built and programmed the first "personal computer" ever made — an Altair 8800, which came in kit form with 1K of memory. In 1977 he left a position as project manager for a large corporation to devote his full efforts to his company, Computerware.)

This month's article is a departure from my series on Design and Development of Application Software, and will cover some *unbiased* points of view on the different operating systems available for the Color Computer. I must admit that it was prompted by Frank Hogg's article in the May issue of THE RAINBOW, since I plan on addressing a number of the statements that he made. I can honestly say that my opinions are unbiased, since Computerware® has roughly an equal number of products on both FLEX (TSC) and OS-9 (Microware and Motorola), and thus don't have any particular reason to promote one over the other. As to background, we've been in the business before either operating system was developed, and in fact, we contributed significantly to the development of the operating system that FLEX was patterned after, Smoke Signal DOS.

Before we even get into a discussion of FLEX versus OS-9, I think it's quite relevant to ask the question: Why use either one? If you look at the ads in the magazines, you can find virtually any product that you might want available for the CoCo under its native operating system Disk Extended BASIC. I had to use Disk Extended BASIC in my comparison because neither FLEX nor OS-9 support cassette. So when we start talking expense, let's take them all into consideration. Just could be that \$30 or even another \$60 for additional software is just a drop in the bucket compared with the startup "dues" that you have to pay to join the alternate operating system club.

Back to the why again. From all the people that I've talked to on the issue of other operating systems, I get the feeling that even though there are a number of reasons, a prevailing one is that they *want to learn* more. For many, a home computer is a personal extension course on one of the most fascinating subjects we will ever have an opportunity to learn about in our lifetimes. When we use phrases like "user friendly" in conjunction with computer operating systems, we are either kidding ourselves or are talking about the Macintosh. There is plenty of user-friendly software available for the CoCo, but it is in the form of application software, not operating systems. If you don't know any operating system at all, is FLEX really that much easier to learn than OS-9? At least, if you want to make a duplicate

disk on OS-9, you do it essentially the same way you did with Disk Extended BASIC. FLEX does not even have a *BACKUP* command. Since a fair amount of the software that is offered on FLEX is the more serious software, needing to be backed up by the user, the absence of a *BACKUP* command in FLEX is a very serious deficiency. True, you can use the *COPY* command to get the backup, but FLEX disk I/O is so slow, and now to avoid having even slower non-contiguous files, you must re-format the diskette even before you can get started.

I could go on picking on FLEX, but that's not the purpose of this article. The point I was just trying to make is that you can pick either one of the operating systems apart if you get to define your own standards. Each has strengths and weaknesses. If it is true that *learning* is an important part of going to a different operating system, then I think the time spent learning OS-9 is much better invested than that of learning FLEX. The design of FLEX is out of date, where OS-9 is patterned after current operating system theory. Although OS-9 doesn't follow UNIX (Bell Labs) exactly, the concepts are there and odds are that your next computer (or the one at work) will also be either running another UNIX look alike or UNIX itself. (It may also be easier to understand filament tube technology, but you'd find learning about integrated circuits more useful.)

An opinion that Frank and I do not differ on is that the initial implementation of OS-9 on the CoCo is second rate at best. Even though I'm sure some of the blame rests with Tandy, I would be inclined to place more blame on Microware since they developed and supposedly debugged it. However this does give Dale Puckett plenty of material for his well presented column! I should also point out that initially both the original FLEX and Frank's adaptation had a number of "bugs" which took time to get ironed out. One of the really nice things about OS-9 is that due to its modular structure, it is easy to update or add new features. Computerware also offers a true double-sided, all-tracks, any-step-rate driver for OS-9 which corrects the deficiencies found in Tandy's. In addition, our package includes a *DIRCOPY* command that solves the multiple files copy problem. True, it's not free, but just as *O-Pack* is reasonably priced, so is our *Disk Fix and Utilities* package.

As to the memory issue, several points need to be made. First, it is not necessarily a requirement that you use a Hi-Res display for everything you do, and if you take out the space used by a Hi-Res display, you have almost 43K left to use. By the microcomputer standard that I'm used to, this is a pretty healthy chunk of space. Even if you include a Hi-Res display, there still is ample space to run most serious

applications, including all of our business software. I won't deny that there have been times that I wished I had more memory under CoCo OS-9, but if I'm honest with you, I need to add that I've also made that same wish on every computer that I've worked on. You could give programmers several megabytes of memory, and at some point in time they would be back wanting more!

The last point of Frank's that I will specifically address is the one of cost. I noticed that FHL has added a "tiny editor" and assembler to their FLEX package, allowing him to make his comparison appear to be somewhat accurate. However, they were not included when most of us purchased FLEX during the last two years, and since they are not full-fledged versions, I do not feel that what you get with OS-9 and FLEX are truly comparable. I feel that this is again a situation of defining the standard to try and make your side appear preferable. In actuality, to get reasonably comparable packages, you need to add another \$70 to the FLEX price for FHL's full Editor and Assembler (or our Editor and Assembler Combo for \$65), which brings the price of each to roughly \$135-\$140. Same basic price, same basic features.

As I mentioned earlier, since you had to spend around \$500 or more for memory and a disk system to use either FLEX or OS-9, it's not clear that a final decision should be made based on a relatively small dollar difference anyway. More important items would include such things as the level of support that will be offered, the availability of products, etc. Since Radio Shack is offering OS-9, it will be more widely used, and thus, this gives a larger base of potential customers. This means more support from outside vendors. Ultimately, this translates into more products and competitive pricing. In the past, products on OS-9 have been more expensive, but that was attributable to the small base of SS-50 users. With thousands of copies being sold by Radio Shack, it won't take long for prices to come down and more products to be developed.

In summary, I'm going to go back to my original question: why buy either one? If you can't answer this question, then save your money. If the purchase is to be made so that you can use some specific software product that only operates on one of the alternate operating systems, then your decision is already made for you. If you are one of the remaining who wants to expand his or her understanding of computers, in the long run you will benefit more from OS-9. Whatever decision you make, remember this: With most of the other inexpensive personal computers available, you wouldn't even get the opportunity to make a decision. The CoCo is probably the most flexible inexpensive computer made!

Hint . . .

What To Do With Overlays

If you have a new keyboard (or one of the upgrade keyboards on the market), and you are using a program like *Scriptit* or *Platinum Worksaver* that uses a keyboard overlay, you can glue a piece of paper on the back of the overlay and mark the openings to indicate which key is which. You can then prop it against the keyboard and read the markings while you program.

*Ed Donovan
Worcester, MA*

STOPPING HUNGRY DATA

Editor:

Mike Fahy's "Boltype" (May 1984 RAINBOW Page 64) was very good. I like dot graphics and play around with it a lot. Although the program was written for a 32K CoCo, it will run nicely on a 16K machine. Change Line 40 to: `40 GOTO 1850`. Add 1850: `1850 PCLEAR 2:GOTO 50`.

Dot graphics do not use the graphics pages of memory and as many (or as few, depending on how you look at it) as are necessary can be *PCLEARED*.

Another way of saving memory when using RS printers is to subtract 128 from the sum of each column total in the *DATA* statements, then add it back in the `?#-2` command. Where *C* is the *READ: PRINT #-2, CHR\$(C+128);*.

DATA statements eat up memory and it is sometimes necessary to get a little "tricky" with the computer.

*Travis Aiton
Azle, TX*

Editor:

Those who have upgraded their E version CoCos to 64K might want to know that the mod does not bring these older machines quite completely up to look exactly like the newer A computers. The problem showed up when an associate of mine tried to run Radio Shack's latest diagnostic ROM pack on his upgraded E board and found that the memory portion of the test did not recognize his computer as having 64K (showed to test only 32K). The problem is the E board uses PB7 of PIA U8 to output a test of jumpers for 32, 64K, whereas the newer A board uses PB6 of the same PIA (which is called U18 on this board).

A simple cut and add to the RAM size jumpers changing PB7 to be PB6 cures the problem, and the new diagnostic ROM pack will now recognize the upgraded E board as having a full 64K. This is the only condition where I have found this difference to be a problem (Color BASIC sets several of the PB lines, both PB6 and PB7 included, when it tests for memory availability, so it sees no difference between the two revisions). I would like to know if anyone else has found any other variations in functional layout between the two versions.

*Richard C. Lawrence
Austin, TX*



The listing:

6.....	117
18.....	244
104.....	163
216.....	224
END	222

DEATH CAVERNS Bill Franks

Long ago during the age of magic and sorcery, there existed a set of caverns so deadly, so terrifying they were known simply as The Caverns of Death. Nothing that entered these caverns had ever returned alive. There was a curse on the caves so that once you entered you couldn't turn back. You had to go deeper and deeper until one of the perils in the caves killed you. Due to an error in a time machine, you were sent back to this age in the form of a bat.

One day while flying around, you unwittingly flew into these caves. At first it was easy, with plenty of room to fly between the stalactites and stalagmites and eat the plentiful bugs. As you went deeper, however, there became less and less room.

You must fly carefully to stay alive. How long will you last? Will you find another exit or will you perish like the others before you? Only time can tell!

The object of *Bats And Bugs* is to accumulate as many points as possible before the caves take their toll. Points are obtained by eating the bugs flying toward you. For each red or blue bug you devour, you will receive 50 points. Avoid the yellow bug — it's poisonous and will kill you if you eat it! Many times the yellow bug will be in your passageway or will jump in front of you, making death inevitable. Every time you gain a multiple of 400 points, you will increase a skill level to a harder cave. After completing level eight, each level thereafter will be of the same difficulty.

There will be times (particularly in the higher levels) when the caves look nearly impossible to navigate. However, there are *no* impossible caves. In this type of cave, you can let your back hit the protrusions just enough to knock off their points. Hitting the stalactites or stalagmites with your front always causes death, but if you aim your course correctly, your back can hit them safely.

You have three lives. Each time you die, a new cave of the same level is drawn. Before you begin each cave, your bat will be moving down the screen. When he gets to the height where you want to start flying, press any key or the fire button to begin play. To reset the game at the end, also press any key or the fire button.

You are given the choice of using a joystick or the Space Bar. Simply move the joystick up and down, or press the Space Bar to climb or don't press it and you will descend.

If the speed up poke (*POKE 65495,0*) doesn't work on your computer then just delete it.

Program Description

- Line 0 sets up the arrays.
- Line 1 puts the computer in the graphics mode.
- Line 2 draws and gets the bat.
- Lines 3 and 4 draw the cave.
- Lines 5 and 6 draw, get the bugs and pick starting places.
- Line 8 moves the bat down screen at start of each cave.
- Lines 10 to 20 are the main loop moving you, and bugs after making sure you haven't gone off board or died.
- Lines 23 to 56 are subroutines used by the main loop.
- Lines 100 to 107 are the death routine.
- Lines 200 to 219 are the score drawing routine.
- Lines 220 and 221 are the completed level routine.
- Lines 300 to 302 get the level and way of movement you wish to start with.
- Lines 400 to 410 draw the title page.

```

0 POKE65495,0:DIMA(1),B(1),C(1),
D(1),E(1):A$="T255V3005CDEFGAB":
GOSUB400
1 GOSUB300:PMODE1,1:PCLS:SCREEN1
,0:PMODE1,3:PCLS:SCREEN1,0
2 DRAW"BM101,100C3E4F4E4F4BM120,
100F4E4F4E4":GET(100,101)-(117,0
95),A,G:GET(120,100)-(137,106),B
,G:GOSUB5:PCLS:GOSUB3:GOTO7
3 FORI=0TO220STEP20:H=RND(3)+1:C
OLORH,1
4 SOUNDRND(255),1:F=RND(LE)+5:G=
190-((LE+5)-F)-5:F=F+15:LINE(I,1
5)-(I+10,F),PSET:LINE(I,15)-(I+2
0,15),PSET:LINE(I+10,F)-(I+20,15
),PSET:LINE(I,190)-(I+10,G),PSET
:LINE(I+10,G)-(I+20,190),PSET:PA
INT(I+5,17),H,H:PAINT(I+5,189),H
,H:NEXT:A=10:B=100:RETURN
5 PCLS:COLOR2,1:LINE(100,100)-(1
03,103),PSET,BF:GET(100,100)-(10
7,103),C,G:COLOR3,1:LINE(120,100
)-(123,103),PSET,BF:GET(120,100)
-(127,103),D,G:COLOR4,1:LINE(130
,100)-(133,103),PSET,BF:GET(130,
100)-(137,103),E,G
6 M1=RND(50)+200:M2=RND(70)+60:
M3=RND(100)+50:M4=RND(70)+60:M5=
RND(50)+100:M6=RND(70)+60:RETURN
7 GOSUB9:PCOPY3TO1:PCOPY4TO2
8 COLOR1,1:FORB=50TO150STEP5:GOS
UB41:FORI=1TO50:NEXT:LINE(A,B-6)
-(A+17,B),PSET,BF:P=PEEK(65280):
I$=INKEY$:IFI$<>"ORP=126ORP=254
THEN10ELSENEXT:LINE(A,B-5)-(A+17
,B+7),PSET,BF:GOTO8
9 DRAW"BM210,0C3R10D14L10U14":LI
=3:SC=0:GOSUB200:PMODE1,3:DRAW"
BM10,10C3E4F4E4F4BM35,10E4F4E4F4
BM60,10E4F4E4F4":COLOR1,1:PMODE1
,1:SCREEN1,0:RETURN
10 IFKE=1THENP=PEEK(345)ELSEJ=JO
YSTK(0):K=JOYSTK(1):IFK<33THENP=
247ELSEP=1
11 IFP=247THENA=A+5:B=B-5:PU=1
12 IFP<>247THENA=A+5:B=B+5:PU=2
13 GOSUB23:IFA>230THENCOLOR1,1:L
INE(A-5,B-6)-(A+17,B+12),PSET,BF
:A=10:GOTO15ELSEGOTO15
14 IFPPOINT(A+8,B-4)<>1ORPPPOINT(A
+8,B+8)<>1ORPPPOINT(A+20,B-2)<>1
ORPPPOINT(A+20,B+8)<>1ORPPPOINT(A+
2,B-4)<>1ORPPPOINT(A+2,B+8)<>1THE
NGOTO100ELSERETURN
    
```

```

15 M1=M1-15:M3=M3-15:M5=M5-15:IF
M1<5THEN50ELSEIFM3<5THEN51ELSEIF
M5<5THEN52
16 COLOR1,1:LINE(A-5,B-6)-(A+15,
B+12),PSET,BF:ONPU GOSUB40,41:PC
OPY3TO1:PCOPY4TO2:PMODE1,1:GOSUB
39
17 IFA>M1-17ANDA<M1+4ANDB>M2-6AN
DB<M2+5THENPLAYA$:GOTO100ELSEIFA
>M3-17ANDA<M3+4ANDB>M4-6ANDB<M4+
5THENPLAYA$:GOTO55ELSEIFA>M5-17A
NDA<M5+4ANDB>M6-6ANDB<M6+5THENPL
AYA$:GOTO56
18 PMODE1,3
19 GOSUB14
20 GOTO10
23 IFB<12THEN100ELSEIFB>180THENB
=180:GOTO100ELSEReturn
37 IFM1<0THENM1=0ELSEIFM3<0THENM
3=0ELSEIFM5<0THENM5=0
38 RETURN
39 GOSUB37:PUT(M1,M2)-(M1+7,M2+3
),C,PSET:PUT(M3,M4)-(M3+7,M4+3),
D,PSET:PUT(M5,M6)-(M5+7,M6+3),E,
PSET:RETURN
40 PUT(A,B)-(A+17,B+6),A,PSET:RE
TURN
41 PUT(A,B)-(A+17,B+6),B,PSET:RE
TURN
50 M1=240:M2=RND(70)+60:GOTO16
51 M3=240:M4=RND(70)+60:GOTO16
52 M5=240:M6=RND(70)+60:GOTO16
55 SC=SC+50:GOSUB200:GOTO51
56 SC=SC+50:GOSUB200:GOTO52
100 PMODE1,1:PLAY"01V30T7CFCFCF
FCFCFCFC":IFB<12THENB=12
101 COLOR1,1:SO=(180-B)/3+5:FORI
=B TO180STEP10
102 B=I:GOSUB41:LINE(A,B-11)-(A+
17,B-3),PSET,BF:FORJ=1TO3:SO=SO
-1:SOUND50,1:NEXT:NEXT
103 PMODE1,3:COLOR1,1:LI=LI-1:LI
NE(10+(LI*25),0)-(26+(LI*25),10)
,PSET,BF:IFLI=0THEN105
104 GOTO221
105 PMODE1,1:SCREEN1,1:COLOR1,1:
LINE(10,0)-(28,10),PSET,BF
106 FORI=1TO500:NEXT:I$=INKEY$
107 P=PEEK(65280):I$=INKEY$:IFI$
<>"ORP=126ORP=254THEN1ELSE107
200 PMODE1,3:SC$=STR$(SC):IFSC>9
950THENSC=0000:GOTO200
201 IFLN(SC$)<6THENSC$="0"+SC$:
GOTO201
202 B1=VAL(MID$(SC$,3,1)):B2=VA
L(MID$(SC$,4,1)):B3=VAL(MID$(SC$
,5,1)):B5=VAL(MID$(SC$,2,1))
203 COLOR1,1:LINE(90,0)-(230,15)
,PSET,BF:COLOR3,1:DRAW"BM210,0R1
0D14L10U14":DRAW"BM120,0":B4=B1:
GOSUB206:DRAW"BM150,0":B4=B2:GOS
UB206:DRAW"BM180,0":B4=B3:GOSUB2
06:DRAW"BM90,0":B4=B5:GOSUB206:I
FSC/400=INT(SC/400)ANDSC>0THEN22
0ELSEPMODE1,1:RETURN
206 ONB4+1 GOSUB210,211,212,213,
214,215,216,217,218,219:RETURN
210 DRAW"R10D15L10U15":RETURN
211 DRAW"D15":RETURN
212 DRAW"R10D7L10D7R10":RETURN
213 DRAW"R10D7L10R10D7L10":RETUR
N
214 DRAW"D7R10U7D14":RETURN
215 DRAW"R10L10D7R10D7L10":RETUR
N
216 DRAW"R10L10D14R10U7L10":RETU
RN
217 DRAW"R10M-10,+14":RETURN
218 DRAW"D14R10U14L10D7R10":RETU
RN
219 DRAW"R10D7L10U7D7R10D7L10":R
ETURN
220 SCREEN1,0:FORI=1TO15:PLAY"T2
55V3004CDEFGAB":NEXT::LE=LE+5:IF
LE>85THENLE=85
221 COLOR1,1:LINE(0,16)-(256,191
),PSET,BF:PMODE1,3:SCREEN1,0::GO
SUB3:A=10:B=100:PMODE1,1:PCOPY3T
O1:PCOPY4TO2:SCREEN1,0:I$=INKEY$
::GOTO8
300 CLSRND(8):PRINT@226,"ON WHAT
LEVEL DO YOU WANT";:PRINT@261,"
TO START?(1-8)";:INPUTLE::IFLE>8
ORLE<1THEN300ELSELE=45+(5*LE)
301 CLSRND(8):PRINT@256,"JOYSTIC
K OR SPACEBAR?(J OR S)";:INPUTKE
$:IFKE$="J"THENKE=2ELSEIFKE$="S"
THENKE=1ELSE301
302 RETURN
400 GOSUB420:LE=75:PMODE1,1:PCLS
:SCREEN1,0:GOSUB3
401 DRAW"BM160,90C3D20R15U10L15R
10U10L10BM180,90D20R15U20BM200,9
0R15L15D20R15U10L5BM220,90R15L15
D10R15D10L15"
402 DRAW"BM105,100R5BM115,90D20U
20F20U20BM140,100R6"
403 DRAW"BM10,90D20R15U10L15R10U
10L10BM30,90D20U10R15D10U20L15BM
50,90R16L8D20BM70,90R15L15D10R15
D10L15"
410 FORI=1TO3000:NEXT:RETURN
420 CLS:PRINT@104,"BATS -N- BUGS
"
421 PRINT@172,"BY"
422 PRINT@232,"BILL FRANKS"
423 PRINT@296,"4939 TUNLAW ST."
424 PRINT@360,"ALEXANDRIA,VA."
425 PRINT@424,"22312"
426 FORI=1TO2000:NEXT:RETURN

```

GAME

16K
ECB



REVERSE REVERSE

By Donald R. Clerc

I first saw this game in a very old issue of *Personal Computing* in the days when there were rumors about "microcomputers," and all games were played on huge mainframes. In the original game of *Reverse*, the player would arrange a list of numbers in ascending order from left to right. Since the CoCo has such excellent graphics, I modified the game so you reverse different lengths of colored bars to an ascending order from top (smallest) to bottom (largest). To move, you tell the computer how many bars (counting from the top) you want to reverse. Here is an example that may help my explanation. The numbers represent colored bars and are arranged from left to right.

2 3 4 5 6 1 7 8 9 0

If you reverse five numbers, the result will be:

6 5 4 3 2 1 7 8 9 0

(first 5 numbers reversed) (remainder stays the same)

Now, if you reverse six numbers, you win!

1 2 3 4 5 6 7 8 9 0

(first 6 numbers reversed)

Playing strategies

There are two main strategies in playing the game, using either an algorithmic or a heuristic approach. An algorithmic approach uses a specific pattern and guarantees a solution in a predictable number of moves. For example, an algorithmic approach to playing this version of *Reverse* would be to move the longest colored bar to the top, then move it down to the bottom. Then move the next longest bar to the top, and move it down to just above the bottom. This method guarantees a solution in $2N-3$ moves, where N is the number of bars in the list. In this game using 10 colored bars, it would take you no more than 17 moves to win. A computer can easily play this type of strategy.

On the other hand, a heuristic approach to solving problems can be thought of as a rule of thumb. Some rules of thumb are very good and lead to good solutions, while others are not so good. Consequently, using a heuristic approach does not guarantee the best possible solution, but for very complex problems (and even some simple ones) it may be more efficient than the algorithmic approach.

Reverse lends itself very well to this heuristic approach. There are many possible solutions to each game. One is best, but the mathematics to determine that solution are quite complex. The simpler algorithmic approach does guarantee a solution, but it is far from efficient (and it gets boring after a while). A good heuristic approach, which takes advantage of "partial orderings" in the list, generally yields a solution within 10 to 20 percent (one or two moves) of perfection.

When using a heuristic approach, your next move is dependent upon the way the list currently appears. No solution is guaranteed in a predictable number of moves, but if you are clever (and sometimes lucky!) you should come out ahead of the simpler algorithmic approach. A good

heuristic approach should solve this game in 10 moves or less.

Good luck!

Variables Used in the Program

- A Array to hold current sequence of numbers
- B\$ INKEY\$ to record your response
- C Color of bars
- D Used in FOR ... NEXT loop to randomize numbers
- E Random number used to randomize list
- J Used in array A to check for repeated numbers
- K Used in array A to generate and keep track of number list
- M\$ Message at end of game; based on total score
- R\$ String input from INKEY\$ for move
- R Numeric value for move; derived from R\$
- S Used to produce ascending sounds
- T Current number of turns (moves)
- W Numbers (1-0) printed on screen
- X X-coordinate to print bars on screen
- Y Used for Y-coordinate to SET colored bars
- Z Used in array A to reverse positions of numbers

Program Line Description

- 10-160 Initialization and instructions
- 170-210 Randomizing numbers
- 220-260 Input move and reverse bars
- 270-290 Check to see if in numerical order
- 300-370 Display score and ask to play again
- 380-410 Subroutine for printing bars on screen

130..... 255
250..... 53
END 144

The listing:

```
10 REM ADAPTED BY DONALD CLERC
   LOUISVILLE, KY
20 CLS: PRINT: PRINT "  reverse
   -- A GAME OF SKILL": PRINT
30 POKE 65495,0: FOR S=1 TO 30:
   SOUND S*5+100,1: NEXT
40 PRINT "DO YOU WANT THE RULES
   (Y/N)? ";
50 B$=INKEY$: IF B$="" THEN 50 E
LSE IF B$="N" THEN PRINT B$: GOT
O 170
```

```

60 CLS:SOUND 100,2: PRINT: PRINT
  "THIS IS THE GAME OF 'REVERSE'.
  TO WIN, ALL YOU HAVE TO DO IS"
70 PRINT "ARRANGE A RANDOM LIST
  OF TEN COLORED BARS (NUMBERED
  FROM 1 THROUGH 0) IN ASCENDIN
  G ORDER FROM TOP (SMALLEST) TO
  BOTTOM (LARGEST)."
```

80 PRINT: PRINT "TO MOVE, YOU TE	LL ME HOW MANY BARS (COUNTING
FROM THE TOP)	YOU WANT ME TO
REVERSE."	
90 PRINT @ 483, "PRESS ANY KEY T	O CONTINUE";
100 B\$=INKEY\$: IF B\$="" THEN 100	
110 CLS: SOUND 100,2: PRINT "FOR	EXAMPLE, IF A LIST OF NUM
BERS IS:	2
3 4 5 6 1 7 8 9 0"	
120 PRINT "AND YOU REVERSE FIVE	NUMBERS, THE RESULT WILL BE:
6 5 4 3 2 1 7	
8 9 0"	
130 PRINT "NOW, IF YOU REVERSE 6	, YOU WIN! 1 2 3 4 5 6 7
8 9 0"	
140 PRINT: PRINT "NO DOUBT YOU W	ILL LIKE THIS GAME OF SKILL, BUT
IF YOU WANT TO STOP, PRESS <Q	> TO QUIT."
150 PRINT @ 483, "PRESS ANY KEY	TO CONTINUE";
160 B\$=INKEY\$: IF B\$="" THEN 160	
170 SOUND 150,2: PRINT @ 480, "	THANK YOU...ONE MOMENT PLEASE ";
180 REM RANDOMIZING LIST	
190 FOR D=1 TO RND(TIMER/100): E	=RND(10): NEXT D: FOR K=1 TO 10
200 A(K)=RND(10): IF K=1 THEN NE	XT ELSE FOR J=1 TO K-1: IF A(K)=
A(J) THEN 200 ELSE NEXT J,K	
210 T=0: GOSUB 390: REM GOTO PRI	NTING ROUTINE
220 PRINT @ 0, " HOW MANY SHALL	I REVERSE? ";
230 R\$=INKEY\$: IF R\$="" THEN 230	ELSE R=VAL(R\$):IF R=0 THEN R=10
240 IF R\$="Q" THEN 370 ELSE IF R	\$<"0" OR R\$>"9" THEN SOUND 1,10:
PRINT @ 448, "PLEASE INPUT ONLY	A NUMBER FROM 0 TO 9.";: GOTO 2
20 ELSE T=T+1	
250 REM REVERSING BARS	
260 FOR K=1 TO INT(R/2): Z=A(K):	A(K)=A(R-K+1): A(R-K+1)=Z: NEXT
K: GOSUB 390: REM GOTO PRINTING	ROUTINE
270 REM CHECK TO SEE IF IN	NUMERICAL ORDER
280 FOR K=1 TO 10: IF A(K)<>K TH	

```

EN 220 ELSE NEXT K
290 PRINT @ 0, " YOU WON IN ONLY
  "T"MOVES. ";
300 REM DETERMINE RESPONSE BASED
  ON NUMBER OF MOVES
310 IF M$(1)="" THEN FOR M=1 TO
  6: READ M$(M): NEXT
320 DATA " WOW!! THAT'S FANTASTI
  C! ", " EXCELLENT SCORE!! ", " VE
  RY GOOD SCORE! ", " THAT'S NOT A
  BAD SCORE. ", " THAT'S OK, BUT YO
  U CAN IMPROVE.", " TRY TO DO BETT
  ER NEXT TIME. "
330 IF T<8 THEN M=1 ELSE IF T>15
  THEN M=6 ELSE M=INT(T/2-2)
340 PRINT @ 32, M$(M);: FOR S=10
  0 TO 235 STEP 5: SOUND S,1: NEXT
350 PRINT @ 448, " TRY AGAIN (Y/
  N)? ";
360 B$=INKEY$: IF B$="" THEN 360
  ELSE IF B$="Y" THEN PRINT B$ " "
  !: GOTO 170
370 PRINT @ 416, "THANK YOU. I
  HOPE YOU HAD FUN!!";: POKE 65494
  ,0: END
380 REM SUBROUTINE FOR PRINTING
  BARS ON SCREEN
390 CLS(0): FOR Y=1 TO 10: IF Y=
  10 THEN W=0 ELSE W=Y
400 PRINT @ (Y+2)*32, W;: IF A(Y
  )>8 THEN C=A(Y)-8 ELSE C=A(Y)
410 SOUND 200-10*A(Y),1: FOR X=1
  0 TO 10+5*A(Y): SET(X,Y*2+4,C):
  NEXT X,Y: RETURN
```

BITS AND BYTES OF BASIC

Variables Revisited

Richard White

Well, another Anniversary Issue is here and we stop to think how far we have come and maybe where the future may lead. Back when Lonnie Falk started THE RAINBOW we were all quite new at computing and any sources of information or programs were viewed with delight. Reading articles on how to program in BASIC on the Model I helped, but there were things in CoCo BASIC that no one else had and things that CoCo BASIC did not have. Study the manuals and experiment was the usual course.

Maybe life is somewhat easier for newcomers now. Perhaps there is too much information for one to digest, and much of it is too technical for the beginner. So, let's take one

of our occasional trips back to basics and look at variables in detail to bring our new programmers on board.

In a high level language like BASIC, variables reference and organize the data used in the program. A variable is simply a name given to a piece of data. Think of data being assigned to a variable and not of the variable equalling the data. Early versions of BASIC sought to reinforce the assignment idea by making one write *LET X=10* rather than *X=10*. *LET* is in Extended Color BASIC but is virtually never used.

Color BASIC supports two types of variables — string, which holds a string of characters, and real or numeric. BASIC constructs variable tables to keep data about active variables and an analysis of the variable table will help you understand how variables work. The variable table starts at the end of your BASIC program and extends upward into memory. The first table entries are real variables, each occupying seven bytes. The first two bytes for each entry are the ASCII values of the first two letters in the variable name. Note that Extended BASIC lets you use variable names longer than two letters, but only the first two are used in the variable table and hence have any meaning. The following five bytes carry the value of the variable in a form readable by BASIC's floating point decimal routines. The CoCo is much more adept at reading these bytes than I am, so let's let it do its thing and not try to guess what it is doing.

However, it might prove interesting to look at the variable table entries. Following is a short program that does just that.

```
5 A=0:B=0:AB$="100":AB=10000
10 B=VARPTR(AB):FORA=B-2 TO B+4
:PRINTPEEK(A);CHR$(PEEK(A));:NEXT
```

In Line 10, *VARPTR(AB)* returns the address of the first data byte associated with variable AB. When the program is run, the following is printed on the screen.

```
65 A 66 B 142 64 @ 28 0 0
```

There will be a graphics character after 142 which I have omitted and will omit when they occur later. The ASCII codes for A and B show up and then three numbers which are all that BASIC needs to store 10000. If you change the value assigned to AB in Line 5 and rerun the program, you can see how the stored values change. Here are some samples to get you started:

```
5 A=0:B=0:AB$="100":AB=100000
65 A 66 B 145 67 C 80 P 0 0
5 A=0:B=0:AB$="100":AB=123456000000
65 A 66 B 165 101 e 244 104 h 128
5 A=0:B=0:AB$="100":AB=2E37
65 A 66 B 252 112 p 189 194 30
```

Strings are stored from the top of RAM down within the string space you define with *CLEAR*. *CLEAR 1000* will reserve 1000 bytes for string storage. Each string is listed in the variable table. We can change our program to look at the table entry for string AB\$ by listing that variable in the brackets after *VARPTR*.

```
5 A=0:B=0:AB$="100":AB=10000
10 B=VARPTR(AB$):FORA=B-2 TO B+4
:PRINTPEEK(A);CHR$(PEEK(A));:NEXT
```

When we *RUN* the program, we get the following on the

screen. Again, any graphics characters printed are not shown below.

```
65 A 194 3 0 38 & 18 0
```

The 65 for A is there but not 66 for B. Instead, we see 194, which is the ASCII value for B plus 128. This flags BASIC that the listing is for a string-type variable. Next is a three, which is the number of characters — our string was 100. The second byte is not used and is set to zero. The third and fourth bytes are the high and low bytes of the address of the beginning of the string. The fifth byte is not used and is set to 0. This is all BASIC needs to find the string and read it.

If we had another line like *15 AB\$="NEW DATA"*, BASIC would write *NEW DATA* to an unused part of the string space and put the new length and address data under the AB\$ listing in the variable table. The old AB\$ string is still in the string space, but reference to it in the variable table is gone. After a while, new string entries will fill up string space, even though it contains some "lost" strings. At that point, CoCo stops to "collect the garbage." Strings listed in the variable table are rewritten over unlisted space moving the free space to the end of string space. This may take a few seconds during which the computer seems to go dead, but it is only cleaning house.

Let's come back for a moment to variable names. In Color BASIC you may use any one- or two-letter combination for a variable except reserved words. What is a reserved word? It is one that is also a BASIC statement or function command. *ON*, *TO*, *GO* and *FN* are examples. When the computer encounters an *ON*, it starts looking for a variable representing a number to use in a following *GOSUB* or *GOTO* action. If your statement had been *ON=20*, no variable comes next, the computer gets confused and registers a complaint as a Syntax Error.

Extended Color BASIC allows you to use whole words as variables, but we now know how the variable table works and that only the first two letters are used. The objective is to allow writing clearer programs, but there are drawbacks that keep people from using the capability. First, there is the added memory used, one byte for each added letter each time the variable is used. Secondly, the number of reserved words (BASIC commands, remember?) become much more numerous. Last is the trouble in devising meaningful words which always are different from any other in the first two letters. If I had two *FOR TO NEXT* loops, one within the other, I might like to name the variable in the outer loop *COUNTONE* and the inner loop variable *COUNTTWO*. Since the first two letters are the same the computer cannot tell the difference and the loops won't work the way you expect. So, we will try *ONECOUNT* and *TWOCOUNT* instead. The first two letters are different, but *ONECOUNT* contains *ON*, a reserved word, and SN Error results. Another loser is *TWO-COUNT*. The computer sees it as *TW-CO* without a variable to assign the result or the equal sign — SN Error.

Real variables represent numbers, and are used directly in equations making calculations resulting in some number. Some BASIC dialects let you define whether a variable will be an integer, a single-precision, floating decimal number or a double-precision decimal number. The higher the precision, the more memory that is necessary to store the number. CoCo will accept positive or negative numbers up to 10 to the 37th power and will display nine significant digits. This is fine for nearly all programming you are likely to do. I miss the ability to define integer variables and benefit from the

memory saving the results. Simple counting and other integer number operations are encountered all the time. Where a wealth of integer data is to be used, it can be put into strings and recovered using methods we have discussed in previous columns and will discuss in the future.

A string variable references a string of characters. In the assignment statement for a string variable, characters must be between quotes or defined using *CHR\$(XX)* or *STR\$(Y)*. Here XX is the ASCII number for the character. Y is a real variable that is converted to a string having a leading space. Examples are *A\$ = " THIS IS AN example"*; *B\$ = CHR\$(191)*, which is a solid red block and *N\$ = STR\$(20)*.

Strings can be added to each other in a process called concatenation. *C\$ = A\$ + " FOR THE ARTICLE ON VARIABLES"*. Now C\$ represents "THIS IS AN example FOR THE ARTICLE ON VARIABLES". If we concatenate C\$ with B\$ like this *D\$ = C\$ + B\$* we would get the same string as before but with a red block after the period. Enter this program and run it.

```
10 A$ = "CHARACTERS"
20 FOR X=1 TO 10 : A$ = A$ + CHR$(8) : NEXT X
30 PRINT A$
```

We know there are characters in A\$, but they don't print. The trick is that *CHR\$(8)* is the backspace or left arrow character. As soon as "CHARACTERS" was printed, 10 backspaces were printed which erased "CHARACTERS." This may not be good for much, but it does give food for thought. There is a watch-out here in that you can add non printing characters to strings that give unexpected results. Also note the *A\$ = A\$ + CHR\$(8)*. A\$ appears on both sides of the equality. We can also write *A = A + 10*. This comes back to the assignment idea. The right side is evaluated and the result is assigned to the variable on the left. The computer finishes its work on the right portion using whatever A or A\$ represent initially before it redefines them.

Both real and string variables can be viewed as either global or local. Actually BASIC variables are always global since they can be used anywhere in a BASIC program. In languages like PASCAL, C and BASIC09, variables have a value only in the particular subroutines or procedures where they are declared. Further, the variable X in one procedure is a different variable from the variable X in a different procedure. We cannot define our variables like this in BASIC, but we can view how we use them as local or global.

In a file program, the variables that refer to the data records are used as global in that they are defined in the input section, changed in the editor saved in another section to tape or disk, used in search and sort sections and in various subroutines. A variable used in a *FOR-TO-NEXT* loop in a subroutine means nothing when you exit that subroutine and may be re-used elsewhere.

There is a savings of memory if certain variable names are prechosen and used only for local purposes. They can be redefined and used again and again in other parts of the program. This serves to help clarify the program if it is known that J and K are always used locally, generally serve counting purposes and never have meaning once the using routine is left. This will work well with some careful discipline and I think is much better than trying to find an unused variable each time a short loop is needed. You should also define string variables and other real variables for local temporary data holding purposes. Just make sure all local variables are defined when they are first used in the routine and do not contain data needed elsewhere when the routine is exited.

Numbers and strings may also be kept in subscripted or

array variables. Here one array name is used to refer to a series of data items. For example *A(1)=23*, *A(2)=45 . . . A(20)=14*. BASIC then sets up a separate portion of the variable table above regular variables in memory for array variable entries. An array variable table for *A\$(10)* look like the following.

-7	ARRAY	65
-6	NAME	128
-5	DISPLACEMENT	VX
-4	TO NEXT ARRAY	YZ
-3	# OF DIMENSIONS	1
-2	NUMBER OF	00
-1	ENTRIES	11
VARPTR(A\$(0)) > A\$(0) LENGTH		
		0
	A\$(0)	—
	ADDRESS	—
		0
VARPTR(A\$(1)) > A\$(1) LENGTH		
		0
	A\$(1)	—
	ADDRESS	—
		0
VARPTR(A\$(10)) > A\$(10) LENGTH		
		0
	A\$(10)	—
	ADDRESS	—
		0

The array for A(N) is similar except the value for each member of the array is in each five-byte block. When a subscripted variable is first used, an eleven entry block is established in the variable table for that variable. Note that A\$(0) is a member of the array. If you need more entries, you must dimension the variable, e.g., *DIM A\$(100)* or *DIM A\$(X)*. You can also have multi-dimensioned arrays in Extended BASIC. A\$ could be dimensioned *DIM A\$(50,10)*. Note that such arrays use memory space. A\$(50,10) requires 7 + 50*5*10 or 2507 bytes of memory for the variable table alone. Finally, if you know you are only going to use a few members of an array, say four or five, then dimension the array, say *DIM A\$(5)* to keep memory use to only what you really need.

Arrays are most useful where the program itself must choose which data item to use. You are permitted and even encouraged to use a variable within the parentheses (A\$(X)) so that a number determined by the program selects the desired array member. Some good examples of array usage have appeared in recent issues of THE RAINBOW. III

GO CO

Readers of GoCo will have to excuse us this month.

During the move, we managed to lose P.C.M. Magazine. How this happened remains somewhat of a mystery. Never the less, we trust that you have already found something in this magazine that has taken your fancy!

Next month we will make it up to you --- promise!

In the meantime, we are reprinting several articles from recent GoCo magazines. Many of you have never seen this magazine and I for one think that that is a shame, because there is a wealth of information in the magazine.

We will continue to feature the Model 100, but we intend to also make this magazine the one to watch for news and reviews on the Model 2000.

This computer has had excellent reviews so far and we are looking forward to doing a review on it soon ourselves.

The other improvement we mean to make for you is to provide readable Bar graphs. We had intended to start with this issue -- oh well!!

CoCoConf has only been briefly mentioned hitherto, but I will take the opportunity that this bit of space affords to say that CoCoConf will include definite opportunities for those of you with Model 100's to get together.

As well, we expect to be able to provide tutorials on the Model 2000 with 'hands on' teaching where possible.

See you in June '85!

Blaxland

P.O. Box 125 BLAXLAND 2774



Computer

Package Deals

MAIL ORDER

Services

64 K Ram Upgrade Kit

posted **\$ 89**

FULL INSTRUCTIONS INCLUDED +2 TAPE PROGRAMS TO USE YOUR 64K COCO
(PLEASE STATE OLD, NEW OR CC2 BOARD WHEN ORDERING)

10 x C-10's posted \$ 11 10 x C-30's posted \$ 16

DATA LIFE VERBATIM SS DD DISKETTES. Box of 10. **\$ 35**

Mailing Labels per 1,000 \$ 16

Cassette Labels per 100 \$ 4

PRINTER PAPER \$33 A BOX 2,000 sheets. freight extra.

GOOD QUALITY PAPER \$70 A BOX 3,200 SHEETS.

Large Range of Software

Acting as BROKER for your TANDY Purchases

Phone: 047 39-3903

A

'Magic Wand' For Your Model 100

By Jim Hawk



Have you been wondering what good you'll get from that socket on the Model 100 protected by a black plastic cover labeled BCR (for bar code reader)? Well, wonder no more.

Another reason your 100 was such a great buy is that there are two bar code readers now available for the PoCo that not only promise to open up the technology to smaller businesses, but to individuals as well. Besides the usual applications like sales and inventory, what that means (in the very near future) is no more keyboard entry for program

listings! Magazine program listings will soon carry extra bar code pages to allow you to "wand-in" the program. It also means you can create your own bar codes on a good quality dot-matrix printer . . . opening up all kinds of possibilities. *PCM* is out in the forefront with a review of the first bar code reader to hit the market for the Model 100 (from B-T Enterprises . . . more on the that in the Product Review section).

First, some bar code basics

Bar codes have been around since the late '60s. Today, 144 companies now list themselves in the bar code manufactur-

er's guide, and there's a bimonthly for the industry called *Bar Code News*. But its very popularity has created dozens of different "languages"—the white and black lines that end up representing numbers and letters, or even entire BASIC statements. UPC, or Universal Product Code, has become the retail standard. Supermarket scanners are now routine, but it was only the summer of 1974 that the first UPC scanners were commercially installed. Bar codes are now used by many other industries—going by names like 3 of 9, 2 of 5, 2 of 7, NATI,EADN,JAN,LOGMARS. Coda-

bar, and Plessey. Although fulfilling different functions, each bar code symbology operates on the same principal: pass a light source and a light detector close enough to the alternating white and black areas to be read, literally, as a code (sort of like the old Morse telegraph code of dots and dashes). In this case, it's a computer "telegraph operator" that then matches up the proper numbers and words to tell us humans what that code means.

But telegraph codes are to bar codes as a biplane is to a 747. For example, the bar code pattern dubbed "3 of 9" (in which there are nine vertical bars in each code and three of them are wide) has the possibility of 512 combinations or characters (twice as many as the entire ASCII set of 256). Mainly because the bar codes would be impractically long, only 43 alphanumeric characters are presently used.

What's it for?

The main intent is for applications like billing, inventory, retailing and manufacturing. But most exciting to the non-business BCR user is the promise of cheaply reproduced software. Word is out that direct programming of the 100 via the bar code reader is just around the New Year's corner. With a few zips of your BCR "wand," you'll soon be able to directly enter program listings without all the manual entry and near-inevitable human error, wear and tear to man and machine, and general frustration from a program that won't run because a colon was typed in as a semicolon. (Bar codes have a built-in "check-digit," theoretically meaning the reader will get the information exact, or beep at you to try it again.) Gutenberg would be proud—his printed page (in this case, bar-coded page) offers to revolutionize software publishing, which up to now has mainly been confined to magnetic form. Whereas hundreds of cassettes of a piece of software might be turned out in a day, millions of pages of bar code could be printed in the same time.

This is new to just about everyone, so don't be embarrassed if you've never heard the term "wand" before. I discovered there's a whole new technological lingo.

A few definitions

The wand is industry jargon for the hand-held unit that actually "reads" the bar codes. The wand, or scanner, connects to a decoder that times the binary

AUSTRALIAN RAINBOW

1's and 0's. It's connected via a 9-pin "data communication interface" (plug) and those signals enable the 100's BASIC to make the final interpretation. The combination of scanner, decoder and interface make up the device accurately termed a "reader." Trouble comes when you start having poor "resolution"—in this context, the minimum width that can be accurately scanned. Even high resolution scanners can be derailed by "voids," or light areas in the bar, often caused by printing errors. "First Read Rate" is defined as the percentage of correct readings that will be obtained in one pass of the scanner over the symbol. (Once you master the steady, sweeping motion needed to get an acceptable pass, that rate should be around 80 percent.) Another measure of BCR performance is "substitution error rate"—the ratio of the number of incorrect entries. The widely tested 3 of 9 code (called Code 39) is said to offer one substitution error out of 3 million characters read. That compares to human/keyboard "first read" error rate estimated around 1 to 250.

Invasion of the Bar Codes

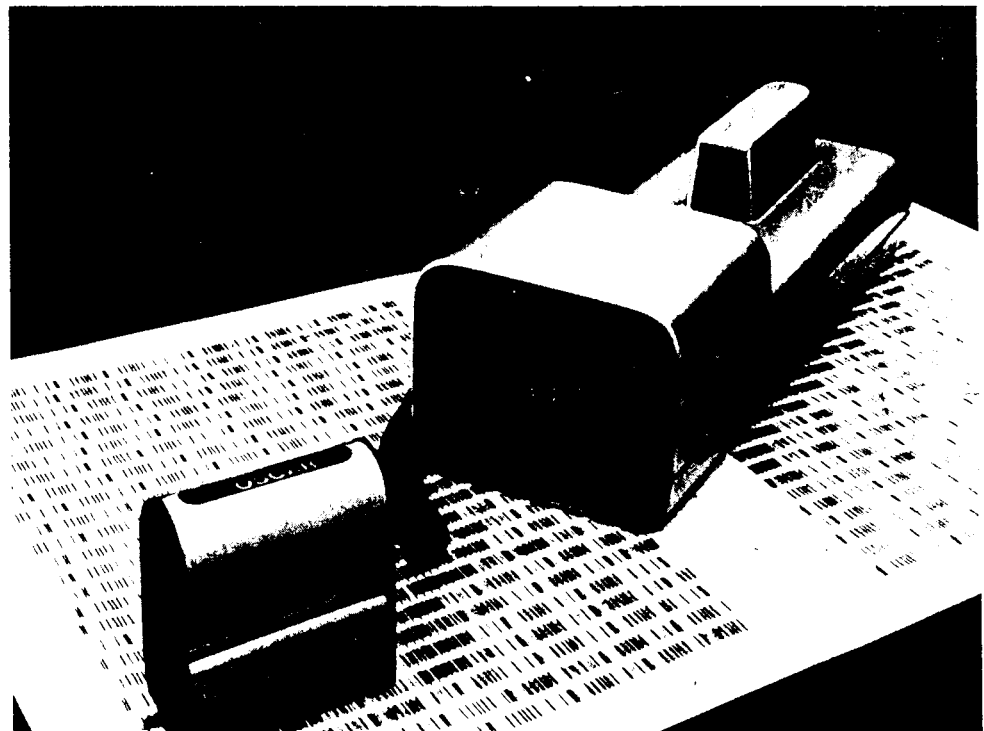
Once you start looking for bar codes, they're everywhere . . . like the one on the front of this very magazine—and, in fact, on virtually everything you might buy in the grocery store. Most of us use bar code technology so casually, we're hardly aware of it—like supermarket

checkout systems using laser bar code readers that tell what the item is and its price. (Instead of wands, they use a super-sophisticated moving laser beam to read the code.)

Libraries, hospitals, auto manufacturing, film developing, package delivery—the list of bar code applications in business is astounding. But all that seems too high-tech and/or high priced. That supermarket BCR cost thousands. Now, the stunning news from Tandy is that Radio Shack will sell a bar code reader (as of Nov. 30, 1983), about the size and shape of a small carrot, for \$99.95 including the necessary software. Meantime, also hitting the market is B-T Enterprises' version of a bar code reader, selling for the subject-to-change price of \$279.

Both firms are taking advantage of solid-state LEDs (or Light Emitting Diodes) that allow small size and low power consumption (as well as low price). Both products are to decode UPC and 3 of 9, meaning small businesses can finally get in on a technology formerly for the Big Guys only.

A bar code device can be important in dealing with the biggest buyer of all: Uncle Sam. The Pentagon now requires bar coding on many contractor-supplied parts and documents. It wouldn't be Department of Defense without an abbreviation: DOD's standard is called LOGMARS, which stands for Logistics Applications of Automated Marking



Databar's OSCAR

and Reading Symbols. And you won't just have to depend on other people's bar codes—a dot matrix printer with graphics capability can let you make your own. (Maybe I'll finally untangle my videocassettes!)

Once the software becomes available, Radio Shack says any DMP-120 or up can do the job of printing bar codes, which are really just fancy graphics. As for program entry, it's a matter of getting a standardized bar code system compatible with Microsoft BASIC . . . and therein lies a story.

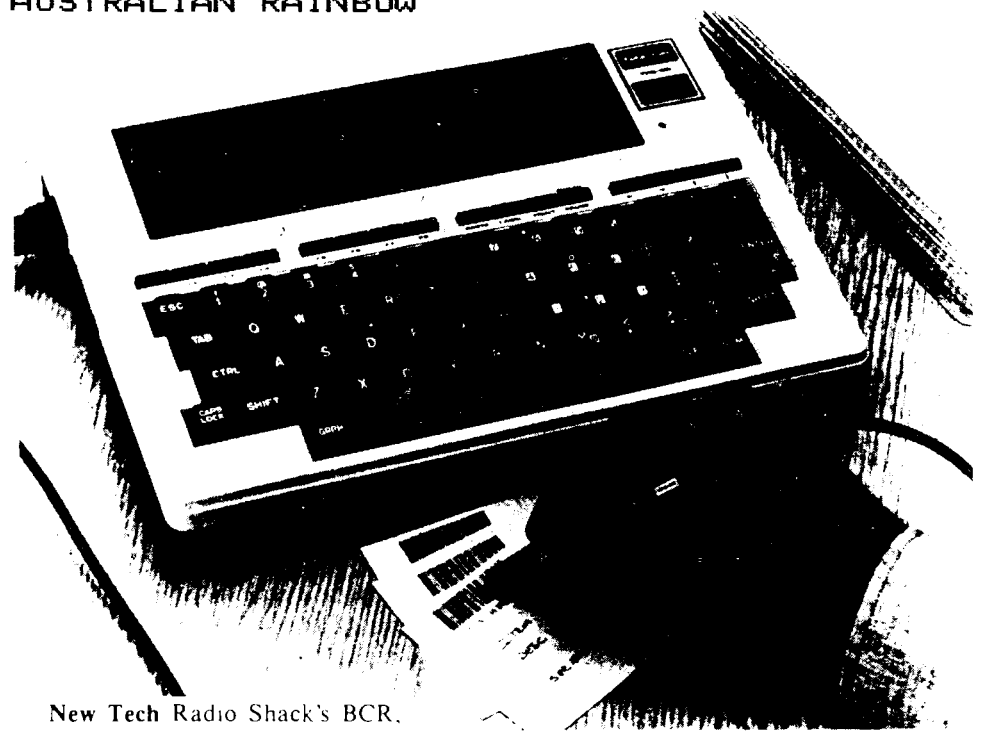
The Chicken or the Egg Syndrome

This idea of bypassing keyboard entry in favor of bar coding has been around a long time. But the advent of actually programming personal computers by bar code is relatively new, and there have been some false starts.

Carl Helmers, cofounder of *Byte Magazine*, made the first effort back in the fall of 1976 when he got the idea to put out a magazine of bar-coded programs called *Paper Bytes*. It never caught on because home computes were more bulky as well as expensive. "It's a chicken and egg thing," says Helmers. But while *Paper Bytes* bit the dust, Helmers went on to form North American Technology Institute and the NATI bar code. Helmers is responsible for a PASCAL program/report that I'm told went into Radio Shack's engineering development of the M100. NATI bar codes appeared to be the best way of squeezing the most programming in the least space, but it wasn't quite right for Tandy's applications. That may explain why Radio Shack decided to adopt its own Tandy Code Standard. It is primarily the same as Helmers' NATI, but has an extra "bit" to indicate whether it's a machine code or a BASIC code. Obviously, Tandy has high hopes for its bar code reader.

One good sign for the new generation of BCRs comes from Tandy's Texas neighbor: Hewlett Packard. HP is the primary supplier of "wand" technology, and currently offers a super-sophisticated calculator computer (HP-41) that can be programmed via an optional wand. The limitation is that it reads HP-41 bar code only, and the programs are mostly engineering-oriented.

Another factor pushing BCR progress is the home computer competition: a firm called Databar out of Minneapolis intends to market its own Optical Scanning Reader (dubbed OSCAR) alleged to be compatible with Atari, Commodore, the no-longer-manufact-



New Tech Radio Shack's BCR.

ured Texas Instruments 99/4A, and possibly Radio Shack's own Color Computer.

The company also has announced it will publish a \$120-a-year bar code magazine, containing programs that OSCAR will theoretically transfer to compatible computers via the cassette port. Release date is early 1984, although no actual hardware has yet been demonstrated.

Bar codes have even become "child's play." The Texas Instruments product called "Speak and Read" is still being sold (slowly), touting storybooks with bar codes on the bottom of each page. The child is supposed to rub the toy's wand over the bar code, and a voice-synthesis speaks the words to children. It seems like a good idea, but wand readers are fussy about the speed they're passed over bar codes, and kids get fussy when their toys act up; my own spot-check found only a few in stores this season.

Leave it to Tandy to come up with a true price vs. performance breakthrough. Although the 1984 computer catalog says the new bar code reader will decode NATI (as well as UPC and 3 of 9), Tandy decided enough modifications were needed to justify a new code.

Roy Irvine is Tandy's bar code buyer, and says the so-called "Tandy code" will be broad-based: allowing either direct entry of machine code, ASCII text, or "tokenized" BASIC. Tokenizing is a way of cutting down on the number of bar codes needed for a given size program: instead of using at least five characters for "GOSUB," you turn it into a one or

two character bar code. On a 4K program this adds up quickly.

Experiments have already been done on the 3 of 9 code, allowing a person to wand-in a program listing. But it's a very inefficient code for that kind of application. Tandy's plan is to market its own bar-code system, with software that can properly tokenize the various forms of BASIC used in the TRS-80 computer line. In other words, with the eventual listing of programs in Tandy bar code, a guy with a Model 100 and a Model II could "wand it in" on his PoCo, and then transfer it to his mini-mainframe. It means bar coding would be opened up to a much wider market than just the 100. It also likely will mean further price cuts for BCRs. Irvine sums it up: "We're trying to get the cost down so everybody can buy one."

What can this new PoCo peripheral do or not do? It can decode several of the most popular industrial bar codes. It will soon be able to enter BASIC programs into the M100. It won't replace the cassette machine, or disk storage; it's still a manual system, and one estimate says it will take two to four minutes to enter a 4K program. No, the bar code reader will just be an excellent addition to an already excellent computer. And, there's no law against saving the program to cassette after you've entered it once by BCR. Soan, you won't have to decide that you *really* want a particular program listing before laboriously typing it in—just wave your "magic" wand.

continued on page 73 . . .

PC WORLD

REVIEW LANGUAGES-4 Forths

Forth, a Programming language invented in the early 1970's by Charles Moore, is a structured interactive extendable high level programming language. This article reviews FOUR versions of Forth. Those Reviewed are Forth - Level 2 from Forth Technology California, PC/Forth Plus from Laboratories Micro Systems Corp California, MMS/Forth from Miller Micro Computer Services, and Forth/32 from Quest Research Corp. Most of these software packages require 64-96K, and at least one disk drive. The article begins by describing how Forth is used. Except for low level commands, Forth is written in Forth. A program, using Forth Language, is continually expanded when it is necessary to extend the compiler. although the language is compiled it is still interactive, as each Forth command is compiled when it is defined. The command can either be temporary or permanent, and can also be used to develop new commands. As the programmer is the one who develops the language, the language is easily maintained and developed. Some of the Packages can read and write PC/DOS Format. However, usual Forth practice is to read and write a disk in one kilobyte blocks displayed on the screen as 16 lines of 64 characters each. One of the programs, MMS Forth, uses special techniques to read and write disks in a

variety of formats. Each of the Forth packages includes Assembler, and in applications where speed is a requirement, Assembly Language definitions run faster. Some of the packages give examples of Forth, that readers can find in Assembly language easily. None of these packages use the new Forth 83 Standard, which supersede previous standards - they are written in the 79 Standard and this varies from package to package. This article goes through each Forth package in great detail and details the advantages and disadvantages of each one. The right choice in Forth packages depends on the user's skill level and the objectives the user wishes to achieve. Forth Level Two is recommended for beginners, as the documentation supplied offers the most help. Forth Level Two is fast and compacted. If PC/DOS compatability is important, then PC/Forth or PC/Forth Plus and Forth 32 are recommended. both of these versions of Forth enable the user to go beyond 64K program size. Those who already know how to use Forth are recommended to use MMS Forth. Documentation of this package is inadequate but a large collection of *pre defined commands may be used. The writer of the article suggests that all packages are continually being refined and enhanced. It is important that users look at them and assist themselves thoroughly.

AUSSIE

make GOOD NEWS

in Rainbow GoCo and MiCo



for ads

Peggy

5283391

Annabel

Sydney

Tandy Apples: The Model 100 Apple Connection

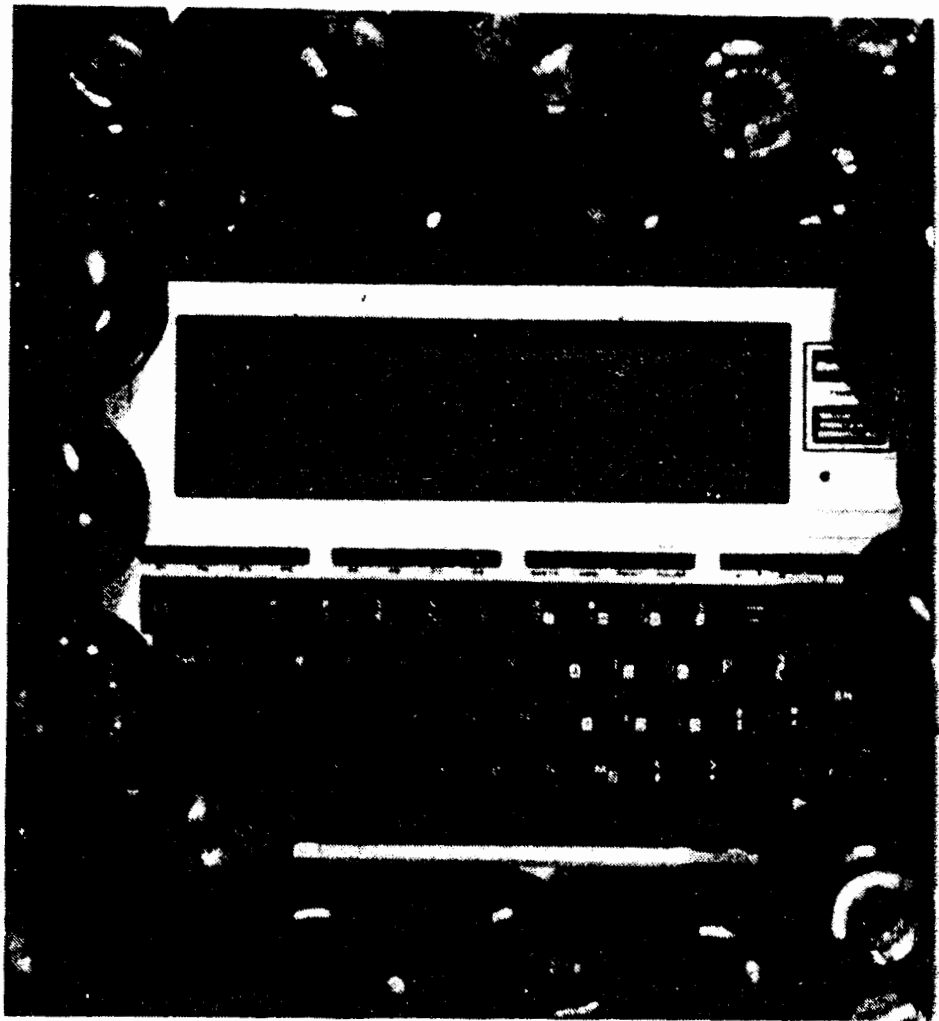
By Gene Cassidy
and
Bill Blue

Radio Shack's Model 100 is quite a machine, and its notebook-sized portability and built-in software answered a real need in my daily travels. But as a happy Apple II+ owner, I missed the disk storage, full-screen editing, word processor formatting and CP/M capability of the at-home system.

One of my first projects after acquiring the Model 100 was to interface it to the Apple. This proved to be more difficult than anticipated. The solution to this problem may be helpful to others. It was done without adding a second phone line and calling modem to modem, which is an obvious answer but a continuing expense.

A friend with a Hayes modem and ASCII Express software (operating under Apple D.O.S. 3.3) was able to transfer data by connecting the modular plugs of the modems on each machine with a double female connector, then dial a non-existent number and connect with the Model 100 in ANSWER mode. To do this make sure the modem status in word length, parity, stop bits, and Baud rate are the same on each machine! Data can then be sent back and forth using each machine's software features. This must be done at 300 Baud using the Model 100's internal modem.

This did not work for me because I use Z-Term "The Professional," which is the Apple CP/M twin of ASCII Express "The Professional." These excellent programs, published by Southwestern Data Systems, share similar commands. But Z-Term Pro (like ASCII Express Pro) has the added feature of looking for a dial tone before dialing. Defeating this by plugging into the telephone system as dialing begins and then



(Gene Cassidy is a pathologist at Scripps Hospitals in Encinitas and La Jolla, California. His special interests include surgical pathology, medical microbiology, online databases, French wines and diesel engines.)

(Bill Blue is president of Marilla Corp Santee, California, specializing in computer hardware and software. Well known computer communications programs he has authored include ASCII EXPRESS, Z-TERM, and the PUBLIC MESSAGE SYSTEM bulletin board.)

AUSTRALIAN RAINBOW

quickly disconnecting from the telephone system did not seem technically elegant (and also required a lot of crawling around under the table).

I also wanted to avoid taking apart the Apple to get at the serial card which usually drives my serial printer, change microswitches, etc., every time data was transferred. Since modems are serial devices, there must be some way to bypass the modem part and get to the serial part as an input/output device.

Since I have an Apple Cat II modem by Novation, I started there. This modem has performed flawlessly for me. Bill Blue, author of the two above communications software programs, came to my rescue. He analyzed, and then bench-tested with a breakout box to follow the connections to and from the Apple Cat II modem card and Model 100 serial port. The problem was solved. My chief contribution was finding the female inline header socket needed to connect to the Novation card. This was harder than expected; success required visiting three electronics stores, culminating in a TV repair supply shop. Here's how:

Materials

4-pin (6 or 8 will work as well, some pins are left unused) female inline header socket

Several feet of flat multiwire cable (only three wires are used)

Female RS-232 plug with fittings hardware

Model 100 serial output cable with male RS-232 plugs at each end (Radio Shack Part #26-4403 or equivalent).

Procedure

From the Novation Apple Cat II Installation Manual, identify expansion I/O multipin connector J2 and pin numbering (page 3).

From the Model 100 manual, identify the RS-232 connector and pin numbering (page 205).

Connect and solder the flat cable at each end as follows:

Female Inline Socket	RS-232
for Apple Cat II	Female
Expansion I/O J2	Connector

1	3
(Output Transmit Data)	(Receive Data)
2	2
(Input Received Data)	(Transmit Data)

4	7
(Signal Ground)	(Ground)

Make good solder joints. Double check the pin connections, especially of the Apple Cat II modem, to avoid potential damage. Pins should not touch. Shield connections with insulating tape as necessary. There you have it. This custom cable will now connect the Apple Cat II expansion pins to the Model 100 serial output cable. Modifications of this connection pattern should be possible for other types of modems and microcomputers; check your manuals for modem pin designations.

It is imperative that the communications program in the Apple and the TELCOM program in the Model 100 both be set to compatible data transmission parameters of word length, parity, stop bits, and Baud rate. In the TELCOM program, remember to configure the STAT function to something other than the M default, which drives input/output of the machine through the built-in modem. Thus, bypassing the internal modem will send the input/output of the Model 100 to the serial port. I have set this to 58N1D; for some reason my Model 100 does not like (I)gnore parity with 8-bit words but must have (N)o parity. See Page 86 of the Model 100 manual.

Now, on the Apple, set the Apple Cat port command in Z-Term for Cat Remote and set the Baud rate to 1200. From this, using appropriate software commands on both machines, data can be sent back and forth. The Model 100 text and BASIC programs can then be stored on Apple disks. One final problem, the Model 100 sends data without linefeeds; the CP/M operating system requires linefeeds. After incoming data has been captured by the Apple and written to disk, the CP/M public domain program FILTEX.COM done to the data now on disk will add linefeeds. Some CP/M word processing programs may be able to do this (especially with text) by formatting the linefeed-free data which has been received.

BULLETIN BOARD SYSTEMS

Editor:

The Department of Aviation Technology at Purdue University, West Lafayette, Ind. is operating an aviation bulletin board system on weekends and holidays. The system will operate from 5 p.m. Friday until 8 a.m. on Monday, and all day on holidays. The system is operating on a 64K Radio Shack Color Computer, and will be restricted to aviation topics only. It can be accessed by calling (317) 743-3897.

Michael S. Nolan
West Lafayette, IN

GOGOCONF

PLAN TO BE THERE!
GOLD COAST QUEENSLAND

JUNE '85.

The Perfect Companion

The TRS-80 Model 2000 looms on the horizon as the most-likely home-base companion to the peregrinating PoCo.

By Danny Humphress

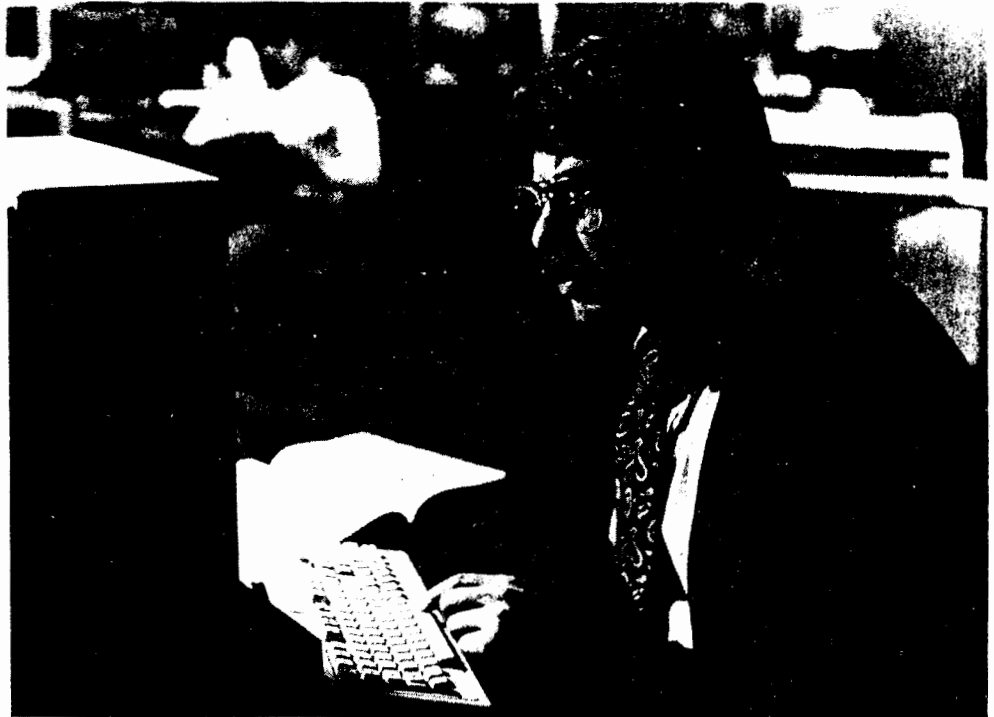
You no doubt already know about Tandy's newest child, the TRS-80 Model 2000. With all the excitement surrounding the introduction of a new computer, it is difficult to get an understanding of just what a system will (and will not) do. Now that the dust has settled, let's take a closer look.

On November 28 Tandy officially unveiled the TRS-80 Model 2000 Personal Computer at Comdex in Las Vegas and, on December 1, in Radio Shack Computer Centers and Computer Plus Centers all around the country. I am naturally skeptical when a company uses phrases like "ultra performance," "dramatic speed," and "exciting" in its sales brochures for a new computer, but upon close examination of the slick new computer, I found that those words were not exaggerated as much as I suspected.

Raw Facts

The Tandy TRS-80 Model 2000 is a 16-bit "personal computer" that uses the popular Microsoft MS-DOS disk operating system. Although it is similar in features to the IBM Personal Computer, Tandy is quick to point out that it is not another IBM PC compatible but a much more advanced machine that just happens to run most IBM PC software.

The basic computer comes with 128K



A serious business machine for the office.

memory, two 720K disk drives, detachable keyboard, RS-232C port, Centronics parallel port, and monochrome display connector. All this for \$2,750. For \$4,250 you get a Model 2000HD which has the same features except with an internal 10-megabyte hard disk and a single floppy disk drive. All you need for a working system is a VM-I monochrome monitor for an additional \$249. A minimum workable Model 2000 system costs just under \$3,000.

MS-DOS and Microsoft's GW BASIC are included with the Model 2000 along with a "Getting Started" book that helps you get going right away without diving into the two other hardbound BASIC and MS-DOS reference manuals. A small Reference Guide is also provided.

Because the Model 2000 uses a fast 16-bit Intel 80186 processor running at 8 MHz, its speed is noticeably better than any other TRS-80 available to date except the Model 16B. The built-in 5¼" thin-line drives sport a 720K storage capacity each for a total of almost 1.5 megabytes. The minimum memory of 128K is expandable to a 768K in 128K increments.

Expansion Options

Radio Shack has finally turned away from the "only we shall see the inside of your computer" policy. This is a very welcome change and one that should open the way for many third parties to manufacture nice user-installable options. On the back of the Model 2000 are four panels which can be removed to reveal slots for four option boards which can be easily slid in or removed. When you purchase an expansion board such as high resolution graphics, you take it home and spend 15 minutes installing it yourself (14½ minutes are spent reading the installation instructions). There will no doubt be many companies making the 2000 do some amazing things by just sliding in expansion boards. This is part of what made the Apple IIe and IBM PC so popular.

You may choose from two monitors for your Model 2000. The VM-I Monochrome Monitor gives you a sharp 80 x 25 display with a low-glare green phosphor 12" screen. It connects directly to the computer requiring no additional boards. The CM-I Color Monitor requires the High Resolution Monochrome Graphics board and the Color

Danny Humphress, PCM's Technical Editor, is the owner of a computer software and consulting firm in Louisville, Ky. Danny brings to PCM his extensive experience with small business computers and applications software.

Graphics Chip Set to operate and offers high resolution color graphics. The screen measures 14". While the color graphics are exceptional, the image is difficult to see when the sun shines through your office window in the morning. This isn't the monitor of choice unless you plan to use graphics often. A solution is to have both monitors to the computer. Yes, it can be done — IBMers often use two monitors on their systems. The VM-1 sells for \$249 and the CM-1 for \$799.

A mere \$449 buys you the user-installable Monochrome Graphics option. With it, you can easily access 600 x 400 one-color graphics on either the VM-1 or CM-1 monitors. If you have a CM-1 Color Monitor, you'll want to add the Color Graphics Chip Kit for an additional \$199. This kit installs on the Monochrome Graphics board. Radio Shack suggests that you let them install it for you, but if you can follow the included installation instructions, there is no reason that you cannot install it yourself. The prices of these graphics options seem high, but the total system price for a comparable IBM PC configuration is still quite a bit more.

If you plan to use your Model 2000 primarily in the home, which I doubt many will, you can purchase the TV/Joystick Adapter for \$249.95 and connect your Model 2000 to your home television. This adapter is, however, a low cost way to get color graphics from your 2000, sacrificing half the resolution of the more expensive Hi-Res and color board. This option was not available at the time of writing, so I have not had a chance to see it first hand.

Radio Shack has jumped on the "mouse bandwagon." For those of you who've been vacationing on Jupiter for the past two years and are not aware that computer mice are not related to Mickey and Minnie, I'll give you a short update. The mouse is a small hand-held device connected to your computer that looks very similar to nature's counterpart. As you move the mouse with your hand across your desk, a corresponding pointer lets you point to and select options on the computer screen. The mouse gives a computer user a more natural way to use computers by using the oldest form of human communication — pointing.

The Digi-Mouse and Digi-Mouse/Clock Controller Board allow you to use "mouse-driven" software such as

AUSTRALIAN RAINBOW

Microsoft *Word* on your Model 2000. The new Microsoft *Windows* operating environment software is included with the controller board. *Windows* lets you view several working programs simultaneously on the computer screen by sectioning the screen into "windows." The size of the windows and functions of the *Windows* software is controlled by the mouse. This software alone is almost worth the price of the board. Totally unrelated to the mouse functions, this board also gives your computer a battery operated clock which frees you from having to enter the date and time each time you turn on the computer. The Digi-Mouse will cost you \$99.95 in addition to \$119.95 for the controller board. It was not available at the time of the writing of this review for evaluation.

When you upgrade your computer to 256K from 128K, you need to purchase the \$299 Internal 128K Kit and have it installed by Radio Shack. It's a shame that Radio Shack does not either already include 256K, since many software packages require 256K, or at least make this a user-installable option. After you have the first 256K, you can install up to two External 256K Expansion Boards. These boards come with 128K already installed for \$499 and you can install the 128K RAM upgrade chips on the board yourself for another \$299 per set. To have a 512K system, you need to have Radio Shack install the first 128K option which goes directly on the computer's main board, buy an External 256K Expansion Board and install an additional 128K RAM upgrade on the board.

One of the few expansion features that is not user installable is the internal 10-Megabyte Hard Disk. For a surprisingly low \$1,699, the hard disk installs inside the computer without taking up space normally used for a floppy disk drive. The IBM-PC XT's hard disk, for example, takes the place of one of the floppy disk drives, making it necessary to add an expensive expansion box if you want two floppy disk drives in addition to the hard disk. Smart move, Radio Shack. What may not be so smart, though, is the lack of connectors on the hard disk controller board (which, by the way, uses one of the four available expansion slots) for adding external hard disks. Although the manual shows these connectors on the board, the finished product does not have them. Radio Shack says they will have another way of adding hard disks. We'll see.

One of the more nifty options available for the 2000 is the Floor Stand and matching Monitor Pedestal. The Floor Stand lets you get your 2000 off the desk and out of the way standing vertically on the floor. You can even rotate the name plate on the computer so it looks like it is meant to stand on its side. Very cute, indeed. The floor stand costs a modest \$145 and includes a keyboard extension cable so you don't have to sit on the floor to use the computer. The monochrome VM-1 monitor may be placed on the Monitor Pedestal (\$89.95) which allows you to adjust it for the best viewing angle. The CM-1 monitor does not work with the pedestal, though.

Software

It seems that Tandy has recently discovered that you need software to sell computers and that their in-house software production efforts are not enough to keep up with the growing demand for quality software.

Thus, they have begun an aggressive adaptation of popular software for their computers — especially the Model 2000.

Now available through Radio Shack are software packages I never thought I would hear Tandy employees openly discuss, let alone see on Radio Shack shelves. Among the popular third-party software packages for the Model 2000 now bearing a Radio Shack catalog number are *PFS:File*, *PFS:Report*, *dBASE-II*, *The Home Accountant Plus*, and *The Witness*. All have their copyright holders' trademarks. This is not a minor move for Tandy, which should mean that you'll be seeing a lot more software available for all TRS-80 computers through the Shack.

Radio Shack is also starting a program where you may order many popular private-label software packages directly through your local Computer Center for fast delivery. Radio Shack will act only as a dealer for these packages, thus you'll need to go directly to the publisher for support for the software. Initially, most of this software will be for the Model 2000, but they expect to have software available for all models through this innovative program. The future certainly looks bright for Model 2000 software.

Radio Shack has many packages already available for the 2000 and many more are listed in the RSC-11 catalog that will be available soon. Included are *MultiMate* word processing, *dBASE-*

II. MA!/*BASIC Four* integrated accounting software, and Microsoft programming languages such as MS-FORTRAN, MS-PASCAL, and MS-GW BASIC Compiler. Also available is RM-COBOL which is the same COBOL package Radio Shack sells for its other computer.

IBM Software Compatibility

Most frequently asked is the question of compatibility with IBM PC software. While the 2000 will not run all packages designed for the IBM PC, it will run a vast majority of them with little problem. You cannot just put in an IBM disk and boot up, but you can copy programs from an IBM format disk to the higher-density Model 2000 format very easily. Most programs will then run as normal.

Tandy has done a service to potential Model 2000 owners by providing a list of software packages that are known to work on the Model 2000. They've even gone one step further by publishing a list of software that will not work on the Model 2000. A note here: The current list of non-working software includes the granddaddy of word processors, *WordStar*. It has recently been discovered, though, that the version of *WordStar* for IBM PC works perfectly well on a Model 2000. Tandy has sent an update to their stores to this effect.

Problem software packages include those that use a copy protection scheme that will not allow you to copy it to a Model 2000 disk and software that directly accesses the IBM's hardware. Packages that use only standard BASIC and/or published MS-DOS system calls should work with no problems on the Model 2000. Some graphics packages will not work or will work differently because of the higher resolution and extended range of colors on the 2000.

Notable exceptions to software that will not run on the Model 2000 include *VisiCalc* and *Lotus 123*. These are two very popular IBM packages. *Lotus 123* is a spreadsheet program that offers sorting and instant graphics. A more advanced program, *Ovation*, is coming for the Model 2000 this summer and it was recently disclosed that Lotus will make *123* available for the Model 2000 shortly thereafter. For now, though, this sends many potential Model 2000 buyers to their IBM dealer.

MS-DOS

MS-DOS is the disk operating system that runs the Model 2000. It is fast

AUSTRALIAN RAINBOW

becoming the standard 16-bit single-user operating system and the majority of today's most innovative software packages are made available first for MS-DOS. The IBM PC uses a slightly different implementation of MS-DOS called PC-DOS which accounts for the compatibility between these two machines. Outstanding among MS-DOS's special features is the ability to create multiple directories of fields which in turn may contain other directories and so on. Mainly designed for hard disk use, this feature allows you to have a number of files only limited by the amount of storage you have available on the disk. With so many files, it is convenient, and in some cases necessary, to organize the disk into multiple directories of related files. The many features of MS-DOS are too numerous and complex to try to describe in one article. Perhaps we'll look further into MS-DOS in a future article.

How Does The 2000 Compare To Big Blue?

The TRS-80 Model 2000 is in many ways superior in features to its obvious competitor, the IBM Personal Computer. It is almost as if Tandy took the IBM PC concept and made it better. You don't think that Tandy pays attention to IBM, do you? Sure they do! And they have managed to overcome most of the shortcomings of the IBM PC while adding a few nice touches here and there.

The most common complaint from PC users is the seemingly thoughtless placement of some of the keys on the keyboard. The left shift key, for example, is moved one space to the left and has a backslash key in its place. The numeric keypad doubles as the arrow cursor control keys making it necessary to toggle the Num Lock key to use both arrows and the numbers on the keypad. This is very frustrating. Another gripe is the lack of indicators on the Caps Lock and Num Lock keys. The Model 2000 addresses all these problems while throwing in an additional two function keys, a Hold key, and an ENTER key on the numeric keypad.

The 2000's keyboard is not perfect. It has a very weird placement of the Alt key which is used like the Shift key to alter the meaning of the other keyboard keys. I personally like the positive firm "click" of the IBM's keyboard, but some people prefer the softer touch of the

Model 2000's. Another small gripe is that Tandy put the darn keyboard connector where no human hand can reach without tilting the computer. You can't please everyone!

The Model 2000 brightly outshines the IBM in the graphics department. The IBM's highest resolution graphics mode gives you 640 x 200 with two colors (one foreground, one background). The color mode on the IBM offers four colors with 320 x 200 screen points. The Model 2000, in its highest resolution, gives you 640 x 400 with a selection of eight out of the 16 available colors. This is twice the resolution of the IBM with four times the number of colors! The IBM monochrome monitor is not capable of displaying graphics while Tandy's monochrome monitor can (without color, of course).

A nifty feature of the Tandy 2000 screen is the ability to do "smooth scrolling." This means that the screen does not scroll up a single line at a time, but by a single screen scan line. This gives the effect of a smooth scrolling just as if you were slowly pulling a scroll behind the glass on the tube! This feature may be turned on and off and it does not seem to work in BASIC. I use it only to show it off and I doubt that many Model 2000 owners will ever use it because it slows the scrolling to the point of annoyance.

Perhaps the most important improvement over the IBM is the increased speed of the Model 2000. This is a result of the advanced 16-bit processor hidden within. The IBM uses the Intel 8088 microprocessor as its brain while the 2000 uses the new Intel 80186 which is a newer generation of the 8086 (the bigger brother of IBM's 8088). The overall performance of the machine is up to three times faster than the IBM, partly because of the processor and partly the result of faster disk input/output with the high density Model 2000 drives.

IBM offers either single- or double-sided track disk drives with 180K or 360K each when using PC-DOS 2.0 or 2.1. The Tandy 2000 uses faster double-sided 80-track thin-line drives that boost the storage to 720K. This alone is good reason to choose the 2000 over the PC.

I have experienced intermittent problems with the Tandy drives, however, especially when reading IBM format disks. This problem is not just on my own 2000, but on just about any Model 2000 I've used. It is not a major prob-

lem, just a minor nuisance since I can always recover from these errors. I am sure Tandy is aware of this problem and that it is working on a solution.

One area in which the Model 2000 lacks in comparison with the IBM is the availability of third-party hardware expansion options. You can put boards in an IBM PC to do anything from letting you run Apple II software to computer networking. There are no such expansion products available for the 2000 yet from third parties. This situation is sure to change soon.

How Does This Affect Other Tandy Computers?

Radio Shack seems inconclusive on where the Model 2000 fits in its current computer line. It costs about the same as a Model 12, yet it can do much more. It is aimed at the same personal/business market as the Model 4/4P. The truth is that it fits somewhere between the Model 12 and the Model 16.

The Tandy 2000 offers many more features than the Model 12 for about the same price. It's going to be hard to sell the "old technology" Model 12 when there's a flashy new Model 2000 across the showroom. The Model 12 still technically has more software available through Radio Shack, but this is changing quickly. And there is always the fact that you can upgrade the Model 12 to the multi-user Model 16B, although it is not unthinkable that this could become an option for the 2000 at some date. It doesn't look as if the Model 12 will be able to compete with its younger brother for long.

The Model 4/4P may be looking over its shoulders, too. If many people who are considering the 4/4P spend a little more and move over to the Model 2000 camp, we may be saying goodbye to it. The III/4/4P format seems to be very popular, so I don't see this happening soon.

Implications For Tandy

The two best-selling computers today are the Apple IIe and the IBM Personal Computer. Tandy really didn't have anything that directly addressed the features of these two computers until the Model 2000. Now it has a product that is in many ways superior to both. Let's hope they made their move in time.

It is interesting that you will not find "Radio Shack" printed anywhere on this computer. The computer is named the Tandy TRS-80 Model 2000. It has

long been said that many people associate Radio Shack with consumer electronics and gadgetry rather than business machines. Tandy hopes that by using the Tandy name on the Model 2000 and future computers, it will set apart the computer products as serious business tools. It is interesting to note, though, that Radio Shack's line of hand-held computer games and their video games also bear the Tandy label. So much for serious business machines.

Conclusions

If you're looking at a personal computer that, through expansion, can offer just about anything you want on a single-user computer, you owe it to yourself to take a look at the TRS-80 Model 2000. If you plan to use IBM PC software, check it on a Model 2000 before you buy.

The Model 2000 is an exciting computer that can only be appreciated by getting to know it personally. It is destined to become Tandy's best selling computer and a strong competitor for the IBM Personal Computer and Apple IIe. You can count on the Model 2000's being around for a long time. Unlike many other companies making similar machines, Tandy has the sales network to make theirs an instant success. This means a bright future for Tandy and for people who are wise enough to own the TRS-80 Model 2000.

Mounting A Remote Reset

- I've wanted to install a switch for Reset on the front of my CoCo, alongside my reverse video switch that I put in for John Skora's reverse modification. On Page 64 of the technical manual, in the upper-left corner, there is a Reset switch diagram. This looks to be nothing more than a simple contact set.

The one thing I am concerned about is the model number of the computer described in the manual. I can't find any indication whether this is an "F" board. I have an "F" board. 32K ECB.

John C. Burke
San Francisco, CA

It really doesn't make any difference which CoCo you have, John. The Reset switch on the rear of your CoCo is nothing more than your description, a simple contact set. Mounting a remote Reset on the front of your computer should not present any problems. This switch actually grounds the Reset line on your 6809E, and other LSI chips, through a diode. On the "F" board the set of terminals nearest the keyboard are the two used for the Reset contacts.

Purchasing Parts

- I am planning to purchase disk drives for my 64K ECB CoCo II. However, I don't plan to buy a complete system. I'm going to buy

from page 66

Future Tech

Market Street Systems out of Portsmouth, N.H., is already selling a dedicated Model 100 and industrial-grade bar code printer that generates every major bar code, but the price is industrial-grade, too: \$6,800.

A glimpse of what the future holds comes from Spectra-Physics' industrial hand-held laser scanner. It looks exactly like something from the set of Star Trek, resembling the fictional "Phaser." And the advantages over a wand are numerous: it can be held up to 38 inches away and still read bar codes; it's faster, and can read through plastic wrapping and smudged codes that a wand would never be able to. This particular unit is for industrial applications, and costs more than the Model 100. But with advances in solid-state lasers (the one now marketed uses a miniature helium-neon gas laser) both the size and price will likely come down.

At the very least, future-tech products like this point to ever-expanding use of bar codes by all kinds of retailers. And when the price is right, can the hand-held laser scanner then be far off?

PCM

parts separately.

I plan to purchase a J&M Disk Controller with Disk Extended BASIC, drive 1 (less disk controller), and the cable to connect the controller and the drive. Can I buy drive 1, the controller and cable, and hook all of them to assimilate a drive 0 system? Where can I get the cable? What is the address of Radio Shack National Parts?

This is going to cost more than a drive 0 system through mail order, but I don't want to purchase a drive 0 system and a J&M controller separately. I want the gold contacts that are standard with the J&M controller.

Brian S. Graham
Cleveland, TN

If you are going to buy a J&M controller, Brian, I would buy a drive from them also. You will get 40 tracks instead of 35 and six ms. track-to-track access. Of course only 35 tracks are available using Radio Shack Extended Disk BASIC.

Along the lines of your question, the cable you are referring to is normally supplied with the controller. The part number of the cable is 8709205.

I know of no special address other than National Parts, Radio Shack, Tandy Center, Fort Worth, Texas 76102.

If you indeed buy a Radio Shack drive 1 and use it for drive 0, you will need a termination resistor, RA2. It can be ordered as manufactured part no. ECM00-13500.

MiCo

The Magazine for the Tandy MC-10

Greg's greatest joy during the last days was to receive material for MiCo. He loved this magazine and it's contributors more than anything else.

So what more can I say than that MiCo lives and will continue to grow! There doesn't seem to be much that you can't do with a MiCo and a few minutes, does there!

LETTERS

The Editor,
Australian MiCo.

Dear Sir/Madam

I really was shocked to hear of Greg's death recently. I have enclosed a sympathy card for Mrs Wilson. Would you please forward it to her.

I do hope someone will continue to publish and edit AUSTRALIAN MiCo. In the hope that this occurs I include a redraft of my article on "VARPTR". I must say that the "Little e" program in May was a lifesaver!

My deepest sympathy,
Kindest regards,
Michael Turk.

Dear Greg,

Just a short note to ask a couple more questions.

1. Re Little E programs on pages 23 and 25 of May '84 issue of MiCo. What alterations would I have to do to get the little E and the Rainbow Check plus and what is the first program on page 23 for and what is the one on page 25 for.
2. How can I program in Machine Language to the MC-10. As was listed in the TRS-80 Newsletter, page 6 "Merge BASIC Program" on the MC-10.
3. Have you a Check Plus program suitable for the MC-10.

Keep up the good work with the MiCo, I have certainly learnt a lot from it.

By the way Greg, how do I make use (and how would I do it) of the listing on pages 18 and 19 of the May '84 issue of MiCo.

Thanking you,
Yours faithfully,
Col Paton.

AUSSIE

make GOOD NEWS

in Rainbow GoCo and MiCo

for ads

Peggy

5283391

Annabel

Sydney



Dear sir,

I recently bought a Tandy's MC-10 color computer and was given your name to contact in regards to information concerning both hardware and software for the MC-10.

The main areas I am interested in are:
 a) of technical information concerning the 'memory' expansion port (pin outs etc.) - and in general - the whole machine. (circuit diagrams - details on type of memory chips used etc.)

b) architecture and applications data on the 6803 - including the machine code listing.

c) hardware and software available.

d) extra commands or instructions that may have been left out of the Manual supplied with the MC-10. (eg. useful peeks, pokes and memory maps etc.)

e) any literature concerning the MC-10 (magazines etc.)

Another point I was wondering about, is if it is possible to use machine language programs, either through an assembler or machine code monitor already on board - or whether it would be possible to do so through an external custom designed monitor - if none is already available on the MC-10. Any information on these points or more generally, any information concerning further expansion - in fact any information on the MC-10 would be welcomed gratefully.

Thank you,
 T.L.NITSCHKE.

ps: If postage is a problem I could arrange for payment of costs.

Dear Greg,

Lately I have had to question myself about some unusual occurrences happening with programs. It would be greatly appreciated if you could help me with

the following question. I recently purchased a MiCoOz tape (MiCoOz #1, in fact) and it contained on it a program called 'Surround'. I think that this program is brilliant, not just the ability of getting the computer to do what it does but also the idea, it's brilliant. Back to the question, one day I was CLOADing 'Surround' normally and after the file name was printed on the screen the program had already 3/4 played on, I accidentally slipped on the Reset button. At this, the OK prompt came up and I thought, I wonder if you can list an amount of the program. I tried and a fair amount of the program listed. So I thought, why not run it, and so I did and then all funny things happened. To my amazement the whole screen turns or CLS's green and hundreds of multi-colored dots and lines appear. Both of these happening are out of the MC-10's ability as far as I know. I would like to know what is happening.

Still staying on MiCoOz, the program MC-10 CONV. is presenting a problem to me. I realize it doesnt just CLOAD on, there is a special way, could you please tell me and explain.

I also have another question from the June edition of MiCo. In this edition, is an excellent program called "CUBE" by John Kellett. My question is, what does Line 2200 read? In my edition it reads:

```
2200 I1=1:I2)5)Yf=oysf&=s0
```

The absence of this line means that you cannot use the "F" command. I would appreciate if you could explain any errors or how to type it in as I do not understand the line. I know I am probably rushing it, but just one last question; It is about "Little E". To me it sounds like a great program but unfortunately I cannot make head nor tails of it. I just dont know what to type in to program the computer. I am sorry to bother you but, I would appreciate it very much if you can answer my questions.

Yours sincerely,
 Shane Herbert.

DEMON*John Kellett*

This program illustrates Maxwell's Demon. For those of you who have not come across this in physics, let me explain that Maxwell put forward the idea that if an imaginary demon had the ability to open and shut a door at the right time between two bottles containing gases, the demon could mix the molecules any way he wanted, if he was quick enough. In particular, he could concentrate the colder molecules in one bottle and the hotter ones in the other. If this happened to any extent at all, the second law of thermodynamics was broken, said Maxwell. The second law says, "Heat can never pass spontaneously from a colder body to a hotter one. In this program you are the Demon. You shut the door by pressing the space bar. You open it by pressing the "/" key. The door does not open or shut at once, so you have time to change your mind. Remember it is the last key you press before the green dots on the right reach the middle. These green dots indicate which red and blue dots are being agitated at the moment. There is also a phase counter, which records the time taken, in roughly 3 second intervals.

When the door is open the red and blue molecules will jump across when they get as near as possible to the door. When you see one move in near to the door, you have a split second to decide whether you want that one to get across or not. When you close the door by pressing the space bar, the randomly moving dots will not get past the barrier you have erected, which says "STOP". You can let them pass again by pressing the "/" key.

What is really happening, if you do it right, is that the increasing disorder, or entropy, is being reversed, merely by issuing a veto at the right time. The on-screen instructions suggest that you try to get all the molecules into one half or the other. What is even harder, because you have control only of a door, is to get all the reds on one side, and all the blues on the other.

Physicists have argued over this apparent anomaly. It seems the reason this does not contradict the laws of nature is that the effort involved in deciding when to

open and close the door makes the action of the gasses not spontaneous. If you think about it, the attention you have to give the screen to make up your mind is quite small, so it is not surprising this explanation has been ignored by many scientists.

Even if you do not get the dots sorted out, you can see that what you are doing

reduces the disorder in the molecules, and that is what Maxwell is talking about in his theory.

now let me explain bits of the program itself. At 5 it goes off to a subroutine which just gives the on-screen instructions. Now that these programs are being distributed on tape, I think a program should explain itself if possible on the screen. When it comes back from there, the program in line 20-40 inserts all the words which have to show on the screen. In line 60 two variables are set up to show the door is open to start with. The meaning of K1 is the line on the screen to which the dots on line 12 of the screen will jump, if any dots get to line 12, and if there is an empty spot to jump to. The other variable, K2, does the same for any of the other dots that get to line 18 on the screen, which is on the other side of the door.

In line 60, J is the line number on the screen, and in line 70 variable 12 is the width of the area where the molecules are, to start with. The variable K is the increment or decrement used for updating 12 later. As you will see, by varying K at various times, I can control the shape of the two areas, top and bottom, where the gasses are kept. The numbers 200 and 153 in line 100 are just to make the sound you hear when the picture is being set up on the screen. Lines 100 to 110 set up the picture and the random red and blue dots. 105 does the dots, two per line. Note that 12 is used to stop dots getting outside the two areas they should be in. 106 to 109 sets up the outline of the hourglass. Line 210 starts a loop which takes us through the top and bottom of the hourglass, screen line by screen line. Line 220 arranges the printing of the phase, on the right hand side of the screen. The loop starting at line 230 covers all the positions across each line. This more than covers the two areas of the hourglass we are interested in, but I could see no quick way to limit, the search to the exact area I wanted it

does no harm except wasting time. Lines 235 and 240 ensure that we do not move the picture of the hourglass as well as the molecules inside. It does this by looking at the color of the dot it is thinking of moving. If the dot is yellow or magenta, it leaves it well alone.

Line 250 agitates a molecule, by pushing it randomly up, down, left or right. If it chooses to leave it in the same spot, or to put it on top of another dot, or on the background picture of the hourglass, line 254 makes it try again. Only when it has moved properly does line 256 eliminate the dot from the old position. Line 260

stops dots migrating off the top or the bottom. 262 and 263 make sure the dots do not slip through the sides of the hourglass. At line 264 the program sets up the new display of the molecule, using the same colour, in variable X, as it had when we found it back at line 235.

Line 280 is the end of the position-on-the-line loop, and lines 290 and 295 make the program do the bottom half of the screen, using the instructions we have just looked at for the top half. It just manipulates J and then goes back to 230.

In line 296, the markers which flash up and down the right hand side of the screen are handled. They are there to show which line is being agitated at any time, and they give a clue as to when the door will

need to be opened or closed. Line 300 is the end of the screen line loop. 305 is cleaning up the right hand side marker at the end of the loop.

Now we are up to the operator interface. If he or she presses the space bar or the "/" key, lines 310 to 370 take care of it. The computer knows whether the door is open or closed by using K1 and K2, which are both described earlier.

In lines 380 to 410 there is a fancy bit of code which jumps the red and blue dots through the door if they are near enough, and if the door is open, and if there is a blank spot to jump to. It also makes the dots jump back if the door is closed and they are trying to get across.

Variable M is set in lines 385 and 395, and it tells us the colour of the thing trying to get across, either to the top, or down to the bottom half. If it is magenta, lines 387 and 397 stop it jumping, because magenta bits are part of the background picture and have to stay put. If the point where they want to jump to is occupied, then lines 390 and 400 stop them jumping this time.

Finally line 420 makes it all happen again.

I hope this gives you some insight into what Maxwell was on about when he raised this philosophical point, which is quite subtle, and yet it can be demonstrated on the screen in a way which brings out the theory.

```

5 GOSUB1000
10 CLS 0
20 PRINT2224,"SPACE BAR";
22 PRINT2256,"TO STOP EM";
24 PRINT2288,"/ TO GO I";
26 PRINT2244,"PHASE ";
30 PRINT2235,"\\GO\\";
40 PRINT2267,"●●●●●●";
50 K1=20:K2=10
60 J=2
70 I2=62
80 K=4
100 FOR I=200 TO 153 STEP -4
102 I1=I1+K:I2=I2-K-K:IFJ>19 GOTO 105
103 IF J>15 THEN K=-4:GOTO106
104 IF J>11 THEN K=0:GOTO106
105 SET(I1+RND(12-12)+4,J,3+RND(2)):SET(I1+RND(12-12)+4,J,3+RND(2))
106 J=J+2:SET(I1-3,J,7)
107 SET (I1-3+K/2,J+1,2)
108 SET (I1+I2,J,7)
109 SET (I1+I2-K/2,J+1,2)
110 SOUND 1,RND(6):NEXT
200 S=80
210 FOR J=2 TO 14 STEP 2
220 PH=PH+1:PRINT2250,PH
230 FOR I3=2 TO 61
235 X=POINT(I3,J)
240 IF X=0 OR X=2 OR X=7 GOTO 280
250 J1=I3-2+RND(3):J2=J-4+RND(3)*2
254 IF POINT(J1,J2)<>0 GOTO 250

```


THE BRIAN McLACHLIN VIDEO SHOW

(Brian McLaughlin presents a number of programs for the MC-10. Brian lives in cool Cooma, high in the mountains - maybe that's the key to his clever programming - he must stay in nights a lot!)

I have used my version of MiCoOz as my Menu. There is an exec instruction (address 64874). This allows the menu to 'new' and 'cload' the following program on the tape. The next program is an original 'MICO MATH', designed to be used as a teaching aid in mathematics for Primary School students.

Any skill level over 5 will be suitable for students over the age of 10.

Function X will auto cload the next program 'MiCoMiMo'.

This program came from a program of Frank Rees, who got it from the Jan Issue of Micro 82. I have altered the program some, removing certain instructions to suit the learner who wishes to dabble in machine language. Commands are:

- R - return to command level.
- M - modify or inspect.
- E - exit from program.
- D - decimal.
- C - change.
- P - to printer.
- X - to new and load the next program on tape.

- J - jump to exec address given.
- C - change to D then hex number.
- C - change to hex, dec number.

You will notice little bits and pieces I have used from MiCo's programs and I have many contributors from this magazine to thank. In particular Bob T for the little face (with a few additions of my own) and Jeremy Gans for the MiCoOz. I hope that those concerned like the changes I've made to their programs.

The rest of the programs apart from the MiCo Blackjack are my own.

MiCoPoker is for the learner again to dabble in the memory, but in decimal. An exercise for the program would be run, when (E,P) appears on screen then Select (p). The program will ask for the address and the data.

For example, input 17026, enter, then input 22. The cursor color will have changed!

MiCoBip is an interesting game too. The

key beep was created by my friend Ian Bishop and myself using the MiCo Exposed. This is a great publication!

I thank you for your effort in MiCo, it is a great publication Brilliantly put together.

```

1 REM***MICO-MATH**BY BRIAN MCLAUGHLIN.-7/84.
2 REM
3 REM *****NOTE***** TO NEW PROGRAM AND AUTO LOAD NEXT PROGRAM ON TA
PE, PRESS DIFFICULTY LEVEL THEN (X).THEN READY CASSETTE.
4 REM THEN PRESS (PLAY) ON CASSETTE. THEN PRESS (ENTER) ON COMPUTER
5 CLSO: CLEAR50
6 A=RND(14)*32: B=RND(15)
7 Z$=CHR$(143+RND(7)*16)
8 PRINT A+B, Z$, :PRINT A+31-B, Z$, :PRINT 448-A+31-B, Z$,
9 P=P+1: IF P=50 THEN 10
10 CLSO: B$=CHR$(128): W$=CHR$(207)
11 PRINT 8, W$+W$+W$+W$+W$+B$+W$,
12 PRINT 40, W$+B$+W$+B$+W$+B$+B$+B$+W$+W$+W$+B$+W$+W$+W$,
13 PRINT 72, W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+B$+B$+W$+B$+W$,
14 PRINT 104, W$+B$+W$+B$+W$+B$+W$+B$+W$+W$+W$+B$+W$+W$+W$,
15 PRINT 167, W$+W$+W$+W$+W$,
16 PRINT 199, W$+B$+W$+B$+W$+B$+B$+B$+B$+B$+W$+B$+B$+W$,
17 PRINT 231, W$+B$+W$+B$+W$+B$+W$+CHR$(140+16)+W$+B$+CHR$(140+16)+W$
+CHR$(140+16)+B$+W$+CHR$(140+16)+W$,
18 PRINT 263, W$+B$+W$+B$+W$+B$+W$+CHR$(131+16)+W$+CHR$(130+16)+B$+W$
+CHR$(131+16)+B$+W$+B$+W$,
19 FOR I=1 TO 1000: NEXT I
20 A=16
21 FOR X=1 TO 7: C=143+(16*X): FOR Y=1 TO 32: PRINT 127+(X*32)+Y, CHR$(C), :NEX
TY, X
22 K1$=' MICO-MATH '
    
```

AUSTRALIAN RAINBOW

```

23 K$=K1$:GOSUB31
24 K2$='BY BRIAN MCLAUGHLIN '
25 K$=K2$:GOSUB31
26 K3$=' PRESS ANY KEY '
27 K$=K3$:GOSUB31
28 K4$=' TO CONTINUE '
29 K$=K4$:GOSUB31:IFINKEY$()' THEN45
30 GOTO22
31 PRINT#231,' ',:FORI=1TO19:L$=LEFT$(K$,I):PRINT#
250-I,L$, :FORD=1TO50:NEXTD,I:FORD=1TO250:NEXTD:RETURN
45 CLS
47 PRINT#136,'MULTIPLICATION'
48 PRINT#168,'ADDITION'
49 PRINT#200,'SUBTRACTION'
50 PRINT#232,'DIVISION'
55 PRINT#262:PRINT:INPUT'DIFFICULTY LEVEL 1 TO 10 ',DL:IFDL(10RD)10
THEN55
59 PRINT'ENTER SELECTION '
60 I$=INKEY$
65 X=INT(RND(DL*10)):Y=INT(RND(DL*10))
68 IFMID$(I$,1,1)='X'THEN5000
69 IFMID$(I$,1,1)='A'THEN120
70 IFMID$(I$,1,1)='S'THEN200
75 IFMID$(I$,1,1)='M'THEN300
80 IFMID$(I$,1,1)='D'THEN400
110 GOTO60
120 AN=X+Y
130 PRINTY'+ 'X'='
140 INPUTYA
150 IFYA=ANTHEN800
160 GOTO1000
200 AN=X-Y
210 PRINTX'- 'Y'='
220 INPUTYA
230 IFYA=ANTHEN800
240 GOTO1000
300 AN=X*Y
310 PRINTX'* 'Y'='
320 INPUTYA
330 IFYA=ANTHEN800
340 GOTO1000
400 IFDL)5THENX=X*10
405 AN=X/Y
406 IFINT(AN)(ANTHEN65
410 PRINTX'/ 'Y'='
420 INPUTYA
430 IFYA=ANTHEN800
440 GOTO1000
515 IFX/Y(1THEN410
800 PRINTYA'IS CORRECT':FORI=1TO500:NEXTI:GOTO5500
1000 CLS:PRINT#36:PRINTYA'IS WRONG'
1001 PRINT#100:PRINTAN'IS THE CORRECT ANSWER'
1002 SOUND100,2:SOUND100,1:SOUND50,2:SOUND10,1:SOUND1,4:FORI=1TO2000
:NEXTI:CLS:GOTO45
5000 GOTO20000
5500 CLS:POKE16555,137:ST=16384:FORD=ST+172TOST+178:POKED,131:NEXT
5505 POKEST+178,134
5510 POKEST+202,137:POKEST+211,134
5515 POKEST+233,137:POKEST+236,206:POKEST+241,206:POKEST+244,134
5520 POKEST+265,133:POKEST+276,138
5525 POKEST+297,133:POKEST+308,138
5530 POKEST+329,134:POKEST+332,134:POKEST+337,137:POKEST+340,137
5535 POKEST+362,134
5540 FORD=ST+365TOST+368:POKED,131:NEXTD
5545 POKEST+371,137
5550 POKEST+395,134:POKEST+402,137
5555 FORD=ST+428TOST+433:POKED,131:NEXTD
5560 FORD=1TO10:POKEST+236,205:POKEST+241,205:GOSUB6000:POKEST+236,2
06:POKEST+241,206:GOSUB6000:NEXTD
5700 RESTORE:FORD=1TORND(7):READQ$:NEXTD:PRINT#75,Q$:GOSUB6100:GOTO1
0000
6000 IFFL=0THEN6004
6001 POKEST+302,136:POKEST+303,132:POKEST+204,140:POKEST+209,140
6002 POKEST+366,128:POKEST+367,128:POKEST+398,139:POKEST+399,135:FL=
0:GOTO6005
6004 POKEST+366,131:POKEST+367,131:POKEST+398,143:POKEST+399,143:FL=
1
6005 FORTL=1TO50:NEXTTL:SOUND100,1:SOUND230,1:RETURN
6100 FORTL=1TO150:NEXTTL:RETURN

```


AUSTRALIAN RAINBOW

PAGE 81

10000 DATA WELL DONE, VERY GOOD, EXCELLENT, TERRIFIC, FABULOUS, KEEP GOING, YOU'RE DOING FINE:FOR I=1 TO 500:NEXT I:CLS:GOTO 45
 20000 CLS:PRINT#226,'PROGRAM WILL NOW NEW AND LOAD'
 20001 EXEC 64874

```

1 CLSO: CLEAR 44
2 A=RND(14)*32: B=RND(15): Z$=CHR$(143+RND(7)*16)
3 PRINT#A+B, Z$, :PRINT#A+31-B, Z$, :PRINT#448-A+31-B, Z$, :P=P+1: IF P=50 THEN 250
4 CLSO: B$=CHR$(128): W$=CHR$(207)
5 PRINT#R, W$+W$+W$+W$+W$+B$+W$, :PRINT#40, W$+B$+W$+B$+W$+B$+CHR$(140+16),
6 W$,
7 PRINT#72, W$+B$+W$+B$+W$+B$+W$+B$+W$, CHR$(140+16)+B$+W$+CHR$(140+16)+W$,
8 PRINT#104, W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(131+16)+B$+W$+CHR$(131+16)+W$,
9 PRINT#166, W$+W$+W$+W$+W$+B$+W$, :PRINT#198, W$+B$+W$+B$+W$+B$+CHR$(140+16),
10 W$,
11 PRINT#230, W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(140+16)+W$+CHR$(140+16)+W$+B$+W$+CHR$(140+16)+W$,
12 PRINT#262, W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(131+16)+W$,
13 W$,
14 A=16
15 FOR X=1 TO 7: C=143+(16*X): FOR Y=1 TO 32: PRINT#127+(X*32)+Y, CHR$(C), :NEXT Y, X
16 K1$='      MICO-MIMO '
17 K$=K1$: GOSUB 25
18 K2$='BY BRIAN MCLAUGHLIN '
19 K$=K2$: GOSUB 25
20 K3$='  PRESS ANY KEY '
21 K$=K3$: GOSUB 25
22 K4$='    TO CONTINUE '
23 K$=K4$: GOSUB 25: IF INKEY$()'' THEN 26
24 GOTO 16
25 PRINT#231, '          ', :FOR I=1 TO 19: L$=LEFT$(K$, I): PRINT#250-I, L$, :FOR D=1 TO 50: NEXT D, I:FOR D=1 TO 250: NEXT D: RETURN
26 CLS
27 GOTO 00
28 H$='': GOSUB 60
29 HB$=H$: RETURN
30 V$=HB$: DA=0: GOSUB 90
35 DB=DA: RETURN
40 ER=0
42 IF V(4)OR V(7) THEN ER=1
44 IF V(5) THEN V=V-48
46 IF V(6) THEN V=V-55
48 RETURN
50 H$='': A=DA/4096: A=INT(A)
52 GOSUB 64
54 DB=DA-4096*A
56 A=DB/256: A=INT(A): GOSUB 64
58 DB=DB-256*A
60 A=DB/16: A=INT(A): GOSUB 64
62 A=DB-16*A
64 AA=A
66 A=A+48-7*(A)*9
68 A$=CHR$(A): H$=H$+A$: A=AA
70 RETURN
72 HB$=HA$
74 V=ASC(HB$)
76 GOSUB 40: IF ER=1 THEN 110
78 DA=4096*A
80 V$=MID$(HB$, 2, 1)
82 V=ASC(V$)
84 GOSUB 40: IF ER=1 THEN 110
86 DB=256*A: DA=DA+DB
88 V$=MID$(HB$, 3, 1)
90 V=ASC(V$)
92 GOSUB 40: IF ER=1 THEN 110
94 DB=16*A: DA=DA+DB
96 V$=RIGHT$(HB$, 1): V=ASC(V$)
98 GOSUB 40: IF ER=1 THEN 110
100 DA=DA+V
105 RETURN
110 PRINT'ERROR, NOT HEX': RETURN
120 PA$=''
122 FOR X=1 TO 4
    
```

AUSTRALIAN RAINBOW

```

124 DB=PEEK(DA):GOSUB29
126 PA$=PA$+HB$:DA=DA+1:NEXT
128 RETURN
130 A$=''
132 FORX=1TO9
134 DB=PEEK(DA):IFDBM32THENDB=46
136 P$=CHR$(DB)
138 A$=A$+P$:DA=DA+1:NEXT
140 RETURN
200 HA$=MID$(O$,3,4)
205 GOSUB72:IFER=1THEN27
210 SDA=DA
215 DB=PEEK(DA):XDB=DB:IFDB(32THENXDB=46
220 B$=CHR$(XDB):GOSUB29
221 IFFL)OTHEMLPRINTHA$'HB$:HB$='H':GOTO259
225 PRINTHA$'HB$'B$,
230 INPUTHB$:IFHB$='H'THEN260
231 IFHB$=''THENHB$='H':GOTO260
235 IFHB$='L'THEN275
240 IFHB$='R'THEN27
245 GOSUB30:IFER=1THEN255
247 POKESDA,DB:COB=PEEK(SDA)
250 IFCOB()DBTHENPRINT'NO CHANGE'
255 DA=SDA
259 IFINKEY$()'THENFL=0
260 DA=DA+1
265 GOSUB50
270 HA$=H$:GOTO210
275 DA=DA-1:GOTO265
300 BA$=MID$(O$,3,3):BA$=BA$+'0'
305 EA$=MID$(O$,2,4)
310 HAPRINT$=EA$
311 GOSUB72
312 LA=DA:IFER=1THEN27
313 IFMO=1THEN405
314 GOSUB120
325 FI$=PA$:GOSUB120
330 SE$=PA$
335 DA=DA-8:GOSUB130
340 PRINTBA$'FI$'SE$'A$
345 IFDA)=LATHEN10
350 GOSUB50:BA$=H$:GOTO320
500 CO$=MID$(O$,3,1)
505 IFCO$='D'THEN530
510 H$=MID$(O$,5):DA=VAL(H$)
515 IFDA)65535THENPRINT'BEYOND RANGE':GOTO27
520 GOSUB50:PRINT'H$:GOTO27
530 HB$=MID$(O$,5):L=LEN(HB$):IFL()4THENPRINT'4 HEX CHARS ONLY':GOTO
27
540 GOSUB74
545 PRINTDA:GOTO27
550 HA$=MID$(O$,3,4):GOSUB72
555 EXECDA
560 GOTO27
800 PRINT
805 PRINT'MIMO COMMANDS:M D C X J E P'
806 CLEAR
810 INPUTO$:C$=LEFT$(O$,1)
815 IFC$='M'THEN200
818 IFC$='P'THENCLS:PRINT'SET TO LPRINT':FL=1:GOTO805
820 IFC$='D'THEN300
830 IFC$='C'THEN500
835 IFC$='J'THEN550
845 IFC$='E'THENCLS:END
850 IFC$='X'THENCLS:GOTO1000
900 PRINT'WHAT?':GOTO27
1000 PRINT*226,'IF YOU WISH TO NEW THIS PROGRAM':PRINT*260,'AND LOAD
THE NEXT ONE':PRINT*282,'READY TAPE,AND PRESS ENTER.'
1300 INPUT'ENTER?',A$
1400 A$=INKEY$:IFA$=''THEN1400:IFA$()'THENCLS:GOTO2000
2000 EXEC64874

```

```

0 REM *****MICO*BLACK*JACK*****BY*BRIAN MCLAUGHLIN*JULY*1984
2 REM TAKEN FROM COCO MANUAL AND ALTERED TO SUIT MC-10 WITH OTHER CH
ANGES.
5 GOTO7000
6 REM
7 DIM S$(5),N$(13),D(52),P(5),C(5)

```

```

10 DATA 16,32,48,96,1
20 DATA ♠ACE♠♠, ♠TWO♠♠,THREE♠,♠FOUR♠, ♠FIVE♠, ♠SIX♠♠,SEVEN♠, EIGHT♠, ♠
NINE♠, ♠TEN♠♠, ♠JACK♠,QUEEN♠, ♠KING♠
30 FOR X=1TO5:READS:S$(X)=CHR$(143+5):NEXTX
40 FORX=1TO 13:READN$:N$(X)=N$:NEXTX
45 CLS(6):PT=0:CT=0:FORX=1TO5:P(X)=0:C(X)=0:NEXT
50 FORX=1TO52:D(X)=X:NEXTX
60 FORX=1TO5:GOSUB1005:P(X)=Z:NEXTX
70 FORX=1TO3:GOSUB1005:C(X)=Z:NEXTX
80 L=257
90 FORM=1TO2:C=P(M):GOSUB500:PT=PT+T:NEXT
100 FORM=1TO3:S=5:GOSUB2005:NEXT
110 L=10
120 S=5:GOSUB2005
130 C=C(2):GOSUB500:CT=CT+T
160 PRINT♠ 267, 'YOUR HAND',
200 L=269:K=3
205 PRINT♠226, 'ANOTHER CARD (YES=Y,NO=N)'
206 PRINT♠0, 'POINTS IN HAND='PT
210 R$=INKEY$:IFR$=' 'THEN210
220 IFR$='N'THEN255
230 C=P(K):GOSUB 500
240 PT=PT+T
242 FORX=1TOK
243 IFJB()OTHEN246
244 IFPT)21AND(P(X)-1)/13=INT((P(X)-1)/13)THENPT=PT-10:JB=1
246 NEXTX:PRINT♠0, 'POINTS IN HAND='PT
247 IFPT)21THENPRINT♠481, 'YOU'VE BUSTED HA. HA. HA.', :SOUND100,4:SOU
ND1,8:GOTO400
250 K=K+1:IFK(6)THEN205
255 L=10
260 C=C(1):GOSUB 500:CT=CT+T
360 IFPT(=CT)THEN380
369 FORZ=150TO200STEP5:SOUNDZ,1:NEXTZ:FORZ=200TO150STEP-4:SOUNDZ,2:N
EXTZ:SOUND50,5
370 PRINT♠484, 'CONGRADULATIONS WINNER.',
371 IFPT=21THENSOUND235,4:SOUND200,2:SOUND210,4:SOUND220,4:SOUND235,
4
375 GOTO390
376 SOUND100,2:SOUND10,8
380 PRINT♠487, 'TOUGH LUCK, KID',
381 SOUND20,4:SOUND1,12
390 REM
400 PRINT♠226, 'ANOTHER GAME (YES=Y N=NO)'
401 JB=0
410 R$=INKEY$:IFR$=' 'THEN410
420 IFR$='Y'THEN45
421 IFR$() 'Y'THENEND
430 IFN=1THENT=11
500 GOSUB 4005:GOSUB2005
TQP GOSUB 3005
1005 Z=RND(52)
1010 IFD(Z)=OTHEN1005
1020 D(Z)=0
1030 RETURN
2005 L1=L
2006 SOUND100,1
2010 FORX=1TO6
2015 L1=L1+32
2020 FORY=1TO5
2030 PRINT♠L1+(Y-1),CHR$(128),
2040 NEXT Y,X
2045 L1=0:L=L+6
2050 RETURN
3005 L1=L-6
3010 FORX=1TO6
3011 SOUND235,1
3020 L1=L1+32
3030 PRINT♠L1+2,MID$(N$(N),X,1),
3040 NEXTX
3045 L1=0
3050 RETURN
4005 S=INT((C-1)/13)+1
4010 N=C-(S*13-13)
4011 IFN=1THENT=11:RETURN
4015 IFN=11THENN=10
4016 IFN=12THENN=10
4017 IFN=13THENN=10

```

AUSTRALIAN RAINBOW

```

4018 IFN()11ORN()12ORN()13THEN=N
4019 RETURN
7000 CLSO
7005 CLEAR38
7010 A=RND(14):B=RND(15)
7015 Z$=CHR$(143+RND(7)*16)
7020 PRINT#A+B,Z$, :PRINT#A+31-B,Z$, :PRINT#448-A+B,Z$, :PRINT#448-A+31
-B,Z$, :P=P+1:IFP=50THEN7038
7038 CLSO:B$=CHR$(128):W$=CHR$(207)
7039 PRINT#8,W$+W$+W$+W$+W$+B$+W$, :PRINT#40,W$+B$+W$+B$+W$, :PRINT#72
,W$,B$+W$+B$+W$+B$+W$+B$+W$+CHR$(156)+B$+W$+CHR$(156)+W$,
7042 PRINT#104,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+CHR$(147)+
W$,
7045 PRINT#167,W$+CHR$(156)+W$+B$+W$+B$+B$+W$+CHR$(156)+W$+B$+W$+CHR
$(156)+B$+W$+B$+W$,
7046 PRINT#199,W$+CHR$(156)+CHR$(147)+B$+W$+B$+B$+W$+CHR$(156)+W$+B$
+W$+B$+B$+W$+W$,
7047 PRINT#231,W$+CHR$(147)+W$+B$+W$+CHR$(147)+B$+W$+B$+W$+B$+W$+CHR
$(147)+B$+W$+B$+W$,
7048 PRINT#296,CHR$(156)+W$+CHR$(156)+B$+W$+CHR$(156)+W$+B$+W$+CHR$(
156)+B$+W$+B$+W$,
7049 PRINT#329,W$+B$+B$+W$+CHR$(156)+W$+B$+W$+B$+B$+W$+W$,
7050 PRINT#360,CHR$(147)+W$+B$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+B$+
W$,
7999 FORI=1TO1000:NEXTI:CLS:GOTO6

```

```

1 REM *****MICO-POKER*****BY BRIAN MCLAUGHLIN. JULY 1984

```

```

10 CLEAR40
12 CLSO
13 A=RND(14)
14 B=RND(15)
15 Z$=CHR$(143+RND(7)*16)
16 PRINT#A+B,Z$,
17 PRINT#A+31-B,Z$,
18 PRINT#448-A+B,Z$,
19 PRINT#448-A+31-B,Z$,
20 P=P+1:IFP=50THEN25
25 CLSO:B$=CHR$(128):W$=CHR$(207)
26 PRINT#8,W$+W$+W$+W$+W$+B$+W$,
27 PRINT#40,W$+B$+W$+B$+W$,
28 PRINT#72,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(156)+B$+W$+CHR$(156)+W$,
29 PRINT#104,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+CHR$(147)+W$
30 PRINT#167,W$+CHR$(156)+CHR$(154)+CHR$(147)+CHR$(147)+CHR$(147)+B$
+CHR$(147)+B$+CHR$(147)+B$+CHR$(147)+CHR$(147)+B$+CHR$(147),
31 PRINT#182,CHR$(147)+CHR$(146),
32 PRINT#199,W$+CHR$(147)+CHR$(154)+W$+B$+W$+B$+W$+CHR$(147)+CHR$(15
6)+B$+W$+CHR$(147)+B$+W$+CHR$(147)+CHR$(154),
33 PRINT#231,W$+B$+B$+W$+CHR$(147)+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$
+W$+B$+W$,
50 FORI=1TO2000:NEXTI
60 A=16:FORX=1TO7:C=143+(16*X):FOR Y=1TO32:PRINT#127+(X*32)+Y,CHR$(C)
,:NEXT Y,X
65 K1$=' MICO-PEEKER-POKER '
66 K$=K1$:GOSUB97
67 K2$=' BY BRIAN MCLAUGHLIN '
68 K$=K2$:GOSUB97
69 K3$=' TO PEEK OR POKE '
70 K$=K3$:GOSUB97
71 K4$=' ADDRESS IN DECIMAL '
72 K$=K4$:GOSUB97
73 K5$=' E= EXAMINE PEEK '
74 K$=K5$:GOSUB97
75 K6$=' P= POKE ADDRESS '
76 K$=K6$:GOSUB97
77 K7$=' S=POKE SAME ADDRESS '
78 K$=K7$:GOSUB97
79 K8$=' R=RETURN TO MODE '
80 K$=K8$:GOSUB97
81 K9$=' H=HIGHER LOCATION '
82 K$=K9$:GOSUB97
83 K10$=' L=LOWER LOCATION '
84 K$=K10$:GOSUB97:K11$=' IF YOU WISH TO NEW '
85 K$=K11$:GOSUB97
86 K12$=' AND LOAD THE NEXT '
87 K$=K12$:GOSUB97

```

```

88 K13$=' PROGRAM ON TAPE
89 K$=K13$:GOSUB97
90 K14$=' THEN AT E ADDRESS 1 '
91 K$=K14$:GOSUB97:K15$=' PRESS (X) ':K$=K15$:GOSUB97
92 K16$=' AND START CASSETTE ':K$=K16$:GOSUB97:K17$=' PRESS ANY KEY
':K$=K17$:GOSUB97
93 K18$=' TO CONTINUE ':K$=K18$:GOSUB97
95 IFINKEY$()' THEN99
96 GOTO65
97 PRINT#231,' ',:FORI=1TO19:L$=LEFT$(K$,I):PRINT#
250-I,L$,:FORD=1TO50:NEXTD,I:FORD=1TO250:NEXTD:RETURN
99 CLS
100 INPUT'E P',A$
200 IFA$='E'THEN500
300 IFA$='P'THEN1000
400 GOTO100
500 INPUT'ADDRESS',A
600 PRINTA$='PEEK(A):INPUT'HIGHER/LOWER PEEK',A$
700 IFA$='L'THENA=A-1:GOTO600
800 IFA$='H'THENA=A+1:GOTO600
850 IFA$='X'THENGOTO2000
900 GOTO100
1000 INPUT'ADDRESS / DATA',A,D
1100 PRINTA$:POKEA,D
1200 INPUT'HIGHER/LOWER/SAME POKE',A$
1300 IFA$='H'THENA=A+1:INPUTD:GOTO1100
1400 IFA$='L'THENA=A-1:INPUTD:GOTO1100
1500 IFA$='S'THENINPUTD:GOTO1100
1600 GOTO100
2000 GOTO5000
5000 CLS:PRINT#32,'PRESS X TO NEW AND LOAD.'
5500 A$=INKEY$:IFA$='' THEN5500
5600 IFA$()' THEN6000
6000 EXEC64874

```

```

1 REM-LETTER WRITER-BY BRIAN MCLAUGHLIN. 6/84-FOR 4K MC-10 AND TTY-P
RINTER.
2 REM-TO BACKSPACE AND ERASE ERRORS, PRESS CONTROL A. TO DUMP LETTER
TO PRINTER, CONTROL Z.
3 REM-LETTER WILL AUTOMATICALLY DUMP TO PRINTER AFTER SOUND WHICH IS
THERE AS A WARNING OF DUMP.
4 REM-ERASE REM STATEMENTS AFTER THEY ARE NOTED AND UNDERSTOOD.
6 CLS
7 INPUT'DATE ',D$
8 CLS:GOSUB999
9 CLEAR2000
10 DIMP$(25)
30 A$=INKEY$:IFA$='' THEN150
40 IFA$=CHR$(10) THEN200
41 IFA$=CHR$(8) THEN160
45 IFC)64 THEN100
47 IFZ)1600 THEN200
48 IFLEN(P$(P))250 THENP=P+1
49 IFC)60 ANDA$=CHR$(32) THENC=0:P$(P)=P$(P)+CHR$(13):GOTO30
50 P$(P)=P$(P)+A$
51 Z=Z+1:C=C+1
52 IFA$=CHR$(13) THENC=0
55 PRINTA$,
70 GOTO30
100 P$(P)=P$(P)+CHR$(45)+CHR$(13)+A$
105 C=1
106 IFP)1600 THEN200
110 GOTO30
150 POKEPEEK(17024)+256+PEEK(17025),128
151 GOTO30
160 L=LEN(P$(P)):P$(P)=MID$(P$(P),1,L-1):C=C-1
170 GOTO55
200 SOUND2,50:PC=0
210 LPRINTP$(PC),
220 IFPC)PTHENRUN9
230 PC=PC+1
240 GOTO210
999 CLS:LPRINTTAB(10)D$
1000 LPRINTTAB(48)'BRIAN MCLAUGHLIN'
1001 LPRINTTAB(48)'P.O. BOX 7,'
1002 LPRINTTAB(48)'COOMA N.S.W.'
1003 LPRINTTAB(48)' 2630 50'
1004 RETURN

```

AUSTRALIAN RAINBOW

```

1 REM:DECIMAL TO HEX TO DECIMAL BY BRIAN MCLAUGHLIN. JUNE 1984.
9 CLEAR100
10 A=RND(14)*32:B=RND(15)
12 Z$=CHR$(143+RND(7)*16)
14 PRINT#A+B,Z$,:PRINT#A+31-B,Z$,
15 P=P+1:IFP=50THEN30
16 CLSO:B$=CHR$(128):W$=CHR$(207)
17 PRINT#9,W$+W$+W$+W$+W$+B$+W$,
18 PRINT#41,W$+B$+W$+B$+W$,
19 PRINT#73,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(156)+B$+W$+CHR$(156)+W$,

20 PRINT#105,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+CHR$(147)+W$
,
21 PRINT#164,W$+CHR$(156)+CHR$(155)+B$+W$+CHR$(156)+B$+W$+CHR$(156)+
B$+W$+B$+W$+CHR$(156)+W$+CHR$(156)+W$+B$+W$+CHR$(156)+W$+B$+W$,
22 PRINT#196,W$+B$+W$+B$+W$+CHR$(156)+B$+W$+B$+B$+W$+B$+W$+B$+W$+B$+
W$+B$+W$+CHR$(156)+W$+B$+W$,
23 PRINT#228,W$+CHR$(147)+CHR$(158)+B$+W$+CHR$(147)+B$+W$+CHR$(147)+
B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147),
24 PRINT#301,CHR$(156)+W$+CHR$(156)+CHR$(149)+CHR$(156)+CHR$(157),
25 PRINT#334,W$+B$+CHR$(149)+CHR$(147)+CHR$(151),
26 PRINT#395,W$+B$+W$+B$+W$+CHR$(156)+B$+CHR$(156)+CHR$(147)+B$+CHR$
(147)+CHR$(156),
27 PRINT#427,W$+W$+W$+B$+W$+CHR$(156)+B$+B$+B$+W$,
28 PRINT#459,W$+B$+W$+B$+W$+CHR$(147)+B$+CHR$(147)+CHR$(156)+B$+CHR$
(156)+CHR$(147),
30 FORI=1TO2000:NEXTI
35 A=16
40 FORX=1TO7:C=143+(16*X):FORY=1TO32:PRINT#127+(X*32)+Y,CHR$(C),:NEX
TY,X
45 K1$=' MICO DEC TO HEX '
46 K$=K1$:GOSUB60
47 K2$=' BY BRIAN MCLAUGHLIN '
48 K$=K2$:GOSUB60
49 K3$=' PRESS ANY KEY '
50 K$=K3$:GOSUB60
51 K4$=' TO CONTINUE '
52 K$=K4$:GOSUB60:IFINKEY$()' THEN99
53 GOTO45
60 PRINT#231,' ,:FORI=1TO19:L$=LEFT$(K$,I):PRINT#
250-I,L$,:FORD=1TO50:NEXTD,I:FORD=1TO250:NEXTD:RETURN
99 CLS:PRINT#4,'DECIMAL TO HEX TO DECIMAL'
100 INPUTA$
110 IFRIGHT$(A$,1)='H'THENL=LEN(A$)-1:GOTO250
120 IFASC(A$)(48ORASC(A$))57THEN350
130 REM DEC/PART
140 A=VAL(A$)
150 Q=INT(A/16)
160 R=A-16*Q
170 IFR)9THENR$=CHR$(R+55):GOTO190
180 R$=RIGHT$(STR$(R),1)
190 S$=(R$+S$)
200 A=Q:IFA=0THEN220
205 L=L+1:B$=LEFT$(A$,L)
210 GOTO150
220 PRINT:PRINTA$,'DECIMAL=',S$,'HEX'
230 GOTO360
240 REM HEX/PART
250 B$=LEFT$(A$,L)
260 FORI=0TOL-1
270 T$=MID$(A$,L,1)
280 IFASC(T$)=32THEN350
290 IFASC(T$)64THENT=ASC(T$)-55:GOTO310
300 T=VAL(T$)
310 REM
315 W=T*16+I+W
320 L=L-1:NEXTI
330 PRINT:PRINT B$,'HEX=',W,'DECIMAL'
340 GOTO360
350 PRINT'INPUT ERROR TRY AGAIN'
360 W=0:S$="":A$="":GOTO100
3101 FW=0 THEN W=1

```



```

#00 REM ***MICO-MENU***BY BRIAN MCLAUGHLIN.6/84.
125 CLSO: CLEAR40
130 A=RND(14)*32: B=RND(15)
150 Z$=CHR$(143+RND(7)*16)
160 PRINT A+B, Z$, :PRINT A+31-B, Z$, :PRINT 448-A+31-B, Z$,
200 P=P+1: IF P=50 THEN 250
210 CLSO: B$=CHR$(128): W$=CHR$(207)
260 PRINT A, W$+W$+W$+W$+W$+B$+CHR$(147), :PRINT 40, W$+B$+W$+B$+W$+B$+
CHR$(156),
271 PRINT 72, W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(156)+B$+W$+CHR$(156)+W$
,
280 PRINT 104, W$+B$+W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+CHR$(147)+W$
,
285 PRINT 168, W$+W$+W$+W$+W$,
286 PRINT 200, W$+B$+W$+B$+W$+B$+CHR$(147)+CHR$(147)+B$+CHR$(147)+CHR$
$(147)+B$+CHR$(146)+CHR$(145),
287 PRINT 232, W$+B$+W$+B$+W$+B$+CHR$(155)+CHR$(151)+B$+CHR$(151)+CHR$
$(149)+B$+CHR$(154)+CHR$(149),
288 PRINT 264, W$+B$+W$+B$+W$+B$+CHR$(155)+CHR$(147)+B$+CHR$(154)+CHR$
$(149)+B$+CHR$(155)+CHR$(151),
289 PRINT 326, 'BY BRIAN MCLAUGHLIN', :FOR I=1 TO 2000: NEXT I
290 A=16
300 FOR X=1 TO 7: C=143+(16*X): FORY=1 TO 32: PRINT 127+(X*32)+Y, CHR$(C), :NE
XT Y, X
310 K1$=' MICO-MENU '
311 K$=K1$: GOSUB 328
315 K2$='BY-BRIAN MCLAUGHLIN '
316 K$=K2$: GOSUB 328
320 K3$=' PRESS ANY KEY '
321 K$=K3$: GOSUB 328
325 K4$=' TO CONTINUE '
326 K$=K4$: GOSUB 328: IF INKEY$()' THEN 330
327 GOTO 310
328 PRINT 231, ' , :FOR I=1 TO 19: L$=LEFT$(K$, I): PRINT
250-I, L$, :FOR D=1 TO 50: NEXT D, I: FOR D=1 TO 250: NEXT D: RETURN
330 CLS
335 PRINT 12, 'THE MENU'
341 PRINT '(1) MICO MENU. BY B.MCLAUGHLIN. (2) MICO MATH. BY B.MCLAUG
HLIN. (3) MICO MIMO. BY B.MCLAUGHLIN.'
342 PRINT '(4) MICO DEC2B. BY B.MCLAUGHLIN. (5) MICO-POKE. BY B.MCLAUG
HLIN. (6) MICO/D/2/H. BY B.MCLAUGHLIN.'
343 PRINT '(7) KEY-BEEP. BY B.MCLAUGHLIN. (8) LETTER-$. BY B.MCLAUG
HLIN. (9) MICO-BIP. BY B.MCLAUGHLIN.'
344 PRINT '(10) MICO/JACK. BY B.MCLAUGHLIN. (11) PROGRAM.. BY WHOEVER.
.....(12) PROGRAM.. BY WHOEVER....'
370 INPUT 'PRESS ENTER TO CONTINUE', A$
375 POKE 33356, 200
380 CLS: PRINT 36, 'DEAR GREG. PROGRAM (1) IS THE MICOMENU, CREATED FR
OM J.GANS'S MICO-OZ. IT CONTAINS ALL OF THE PROGRAMS ON TAPE'
390 PRINT 164, 'NOTE THE AUTO NEW AND LOAD ON EXEC FROM WITHIN THE PR
OGRAM. COULD USE SOME POINTERS ON (FILE NAME) FROM WITHIN PROG'
391 PRINT 324, 'ARE YOU ABLE TO ADVISE ME IN THIS AREA ??? GREG I COU
LD USE A LISTING OR TAPE OF HUMBUG OR SIMILAR PROGRAM ??'
395 PRINT: PRINT: INPUT 'PRESS ENTER TO CONTINUE', A$
396 CLS: PRINT 36, 'ALL BUT MICOMENU, MICOMIMO, MICOBLACKJACK... ARE ORIG
INAL PROGRAMS. I REFRAINED FROM USING DATA STATEMENTS (W$, B$)'
397 PRINT 164, 'THIS METHOD OF JEREMY'S IS BETTER FOR THE LEARNER TO
UNDERSTAND. NEXT TAPE WILL CONTAIN A SIMILAR IDEA IN DATA'
398 PRINT 292, 'FOLLOW THE USE OF THE EXEC(64874) VERY HANDY FOR LOAD
ING, I HAVE TO PRESS PAUSE FOR THE TIME BEING, BUT NOT FOR LONG'
400 PRINT: PRINT: INPUT 'PRESS ENTER TO CONTINUE', A$
420 CLS: POKE 33356, 200
460 PRINT 192, 'W A R N I N G . PROGRAM WILL NEW AND LOAD WITH NEXT E
NTER'
490 PRINT: PRINT: INPUT 'PRESS ENTER TO CONTINUE', A$
500 CLS: PRINT 226, 'PROGRAM WILL NOW NEW AND LOAD': FOR I=1 TO 1000: NEXT I
: CLS: PRINT 226, 'DOING IT NOW': GOTO 505
505 EXEC 64874
    
```

```

0 REM ***MICO-DEC-BIN***
1 REM BY BRIAN MCLAUGHLIN JULY 1984.
5 CLSO: CLEAR40
10 A=RND(14)*32: B=RND(15)
12 Z$=CHR$(143+RND(7)*16)
    
```

AUSTRALIAN RAINBOW

```

14 PRINT#A+B,Z$, :PRINT#A+31-B,Z$,
15 P=P+1:IF P=50 THEN 30
16 CLSO: B$=CHR$(128):W$=CHR$(207)
17 PRINT#B,W$+W$+W$+W$+W$+B$+W$,
18 PRINT#40,W$+B$+W$+B$+W$,
19 PRINT#72,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(156)+B$+W$+CHR$(156)+W$,

20 PRINT#104,W$+B$+W$+B$+W$+B$+W$+B$+W$+CHR$(147)+B$+W$+CHR$(147)+W$

21 PRINT#166,W$+CHR$(156)+CHR$(156)+CHR$(146)+B$+B$+B$+B$+B$+CHR$(14
5)+B$+B$+B$+B$+B$+B$+CHR$(154),
22 PRINT#198,W$+B$+B$+CHR$(154)+CHR$(147)+CHR$(147)+B$+CHR$(147)+CHR
$(147)+CHR$(145)+B$+CHR$(147)+CHR$(147)+CHR$(146)+CHR$(147),
23 PRINT#213,CHR$(147)+B$+CHR$(154), :PRINT#230,W$+B$+B$+CHR$(154)+CH
R$(155)+CHR$(151)+B$+CHR$(154)+B$+CHR$(149)+B$+CHR$(154),
24 PRINT#242,CHR$(154)+CHR$(154)+CHR$(154)+CHR$(154)+CHR$(149)+B$+CHR$(154),
25 PRINT#262,W$+CHR$(147)+CHR$(147)+CHR$(152)+CHR$(155)+CHR$(147)+B$
+CHR$(155)+CHR$(147)+CHR$(149)+B$+CHR$(154)+CHR$(154)+CHR$(154),
26 PRINT#276,CHR$(155)+CHR$(151)+CHR$(146)+CHR$(154),
27 PRINT#300,CHR$(147)+CHR$(147)+CHR$(147)+CHR$(145)+CHR$(147),
28 PRINT#333,W$+B$+CHR$(149)+CHR$(151),
29 PRINT#358,W$+W$+W$+B$+W$,
30 PRINT#390,W$+CHR$(147)+W$+CHR$(146)+CHR$(147),
32 PRINT#422,W$+B$+CHR$(149)+CHR$(154)+W$+CHR$(149)+CHR$(154)+W$+B$+
W$+CHR$(156)+W$+B$+W$+CHR$(156)+B$+W$+CHR$(147)+W$,
33 PRINT#454,W$+W$+W$+CHR$(154)+W$+CHR$(149)+CHR$(154)+W$+B$+W$+CHR$
(147)+W$+CHR$(146)+W$+B$+B$+B$+W$, :PRINT#502,W$+W$,
50 FOR I=1 TO 2000: NEXT I
55 A=16
56 FOR X=1 TO 6: C=143+(16*X): FOR Y=1 TO 32: PRINT#127+(X*32)+Y, CHR$(C), : NEX
TY, X
57 K1$='MICO DEC TO BINARY '
58 K$=K1$:GOSUB 80
59 K2$='BY BRIAN MCLAUGHLIN '
60 K$=K2$:GOSUB 80
61 K3$=' PRESS ANY KEY '
62 K$=K3$:GOSUB 80
63 K4$=' TO CONTINUE '
64 K$=K4$:GOSUB 80:IF INKEY$()'' THEN 99
65 GOTO 57
80 PRINT#231,' ', :FOR I=1 TO 19: L$=LEFT$(K$, I):PRINT#
250-I, L$, :FOR D=1 TO 50: NEXT D, I:FOR D=1 TO 250: NEXT D: RETURN
99 CLS
150 PRINT#64,'ENTER DECIMAL NUMBER'
175 PRINT#96,'FOR CONVERSION',
200 INPUT A
300 FOR I=14 TO 0 STEP -1
400 Z=A AND 2*I
500 A=A-Z:Z=Z/2*I
600 Z=SGN(Z)
700 Z$=RIGHT$(STR$(Z),1)
800 PRINT Z$,
900 NEXT I
1000 PRINT:GOTO 200

```

```

0 CLEAR 29, 20450
1 CLS:REM ***KEY-BEEP*** BY BRIAN MCLAUGHLIN. JUNE 1984.
2 FOR I = 20451 TO 20479
3 READ A
4 POKE I, A
5 CS = CS + A : NEXT I
6 IF CS ( ) 3553 THEN PRINT 'DATA ERROR':END
7 EXEC 20451 : NEW
8 DATA 134 , 242 , 183 , 66 , 135 , 134 , 79 , 183 , 66
9 DATA 134 , 134 , 126 , 183 , 66 , 133 , 60 , 54 , 55 , 134
10 DATA 224 , 198 , 1 , 189 , 255 , 171 , 51 , 50 , 56 , 57

```

```

0 CLS:REM-BY BRIAN MCLAUGHLIN. 6/84.
1 PRINT#32,'***** C A T C H 2 *****'
2 PRINT#161,'USE ARROWS TO MOVE YOUR MAN':PRINT#193,'THE OBJECT OF T
HE GAME IS TO '
3 PRINT#225,'HIT THE BLACK SQUARE FOR POINTS'
4 PRINT#96,'SELECT LEVEL OF SKILL TO BEGIN'
5 PRINT#65,'LEVEL OF SKILL ? (1-4)'
6 Q=0:U=0:S=C:SL=0
7 L$=INKEY$:IF L$=' ' THEN 7

```

```

8 L=VAL(L$):IFL)4THENL=4
9 P=16384
10 CLS
11 PRINT#490,'SKILL='L,
12 FORD=OT010
20 X=RND(400)
22 FORSL=1TOL#20
25 PRINT#Y,' '
30 IFPEEK(P+X)()143THENPRINT#X,CHR$(129),
40 P$=INKEY$
45 IFP$=''THENP$=Z$
50 IFP$='S'THENU=U+1
60 IFP$='W'THENU=U-32
70 IFP$='A'THENU=U-1
75 IFP$='Z'THENU=U+32
76 Z$=P$
81 IFPEEK(P+U)=12#THENGOTO1000
82 IFU(OTHENU=U+32
83 IFU)478THENU=U-32
90 PRINT#Q,' '
100 PRINT#U,CHR$(79)
105 PRINT#500,'SCORE='S,
106 Y=X
110 Q=U
111 PRINT#480,'TIME='D,
115 NEXTSL
120 NEXTD
130 GOTO2000
1000 S=S+10:SOUND100,1
1001 P$=''
1002 Z$=''
1003 IFS=100THEN5000
1004 GOTO20
2000 PRINT#226,'ANOTHER GAME (Y/N)'
2001 K$=INKEY$:IFK$=''THEN2000
2002 IFK$='Y'THENGOTO 0
2003 IFK$='N'THENCLS:END
5000 CLS2:SOUND210,1:CLS4:CLS1:CLS8:SOUND210,2:CLS0:CLS7:CLS1:SOUND2
10,1:CLS
5001 PRINT#226,'YOU WIN BUSTER 100 POINTS'
5002 PRINT#258,'TRY A HIGHER LEVEL CHEAT'
5005 FORI=1TO3000:NEXTI:GOTO 0
    
```

	Ⓒ=CHR\$(209)	Ⓓ=CHR\$(219)	Ⓔ=CHR\$(229)	Ⓕ=CHR\$(239)
	Ⓖ=CHR\$(210)	Ⓗ=CHR\$(220)	Ⓖ=CHR\$(230)	Ⓒ=CHR\$(240)
Ⓙ=CHR\$(201)	Ⓙ=CHR\$(211)	Ⓛ=CHR\$(221)	Ⓜ=CHR\$(231)	Ⓝ=CHR\$(241)
Ⓚ=CHR\$(202)	Ⓟ=CHR\$(212)	Ⓜ=CHR\$(222)	Ⓝ=CHR\$(232)	Ⓖ=CHR\$(242)
Ⓛ=CHR\$(203)	Ⓠ=CHR\$(213)	Ⓝ=CHR\$(223)	Ⓖ=CHR\$(233)	Ⓛ=CHR\$(243)
Ⓞ=CHR\$(204)	Ⓡ=CHR\$(214)	Ⓖ=CHR\$(224)	Ⓞ=CHR\$(234)	Ⓞ=CHR\$(244)
Ⓜ=CHR\$(205)	Ⓢ=CHR\$(215)	Ⓞ=CHR\$(225)	Ⓞ=CHR\$(235)	Ⓞ=CHR\$(245)
Ⓝ=CHR\$(206)	Ⓣ=CHR\$(216)	Ⓞ=CHR\$(226)	Ⓞ=CHR\$(236)	Ⓞ=CHR\$(246)
Ⓞ=CHR\$(207)	Ⓤ=CHR\$(217)	Ⓞ=CHR\$(227)	Ⓞ=CHR\$(237)	Ⓞ=CHR\$(247)
Ⓞ=CHR\$(208)	Ⓡ=CHR\$(218)	Ⓞ=CHR\$(228)	Ⓞ=CHR\$(238)	Ⓞ=CHR\$(248)

The White and the Gray

• I just purchased a Radio Shack disk drive (white case). The dealer told me that it would work on the gray Color Computer. It worked fine for a few days, then it stopped. I then took it to a Radio Shack service center, and they said that a white drive wouldn't work on the gray CoCo.

Dan Schoenbaum
Hollywood, FL

If you are talking about the disk drive and controller package, then the white version

will work on all Color Computers. The older model in a gray case won't work on the Color Computer 2 without modification, or addition of a Multi-Pak Interface, because it requires a +12 volt power supply that the CoCo 2 doesn't have. If you are talking about the drives themselves, the white drives should be used only with a new controller and the gray drives with an old controller. (TDP drives and controllers were the same as the older Radio Shack products.) The new drives are made by a different supplier, and the two have different connections.

• I would like to know if the MC-10's internal board can be upgraded to 32K or 64K. I would also like to know if there are any books or articles about the MC-10's 6803 microprocessor.

Gaston V. Webb
Las Vegas, NV

I don't see any way that the MC-10 can be upgraded internally, since the RAM chips it uses are not a normal type like the ones used in the CoCo. As for the 6803, the only book I know of is the programming manual put out by Motorola Semiconductor in Phoenix.

MORE TRIBUTES

Tandy AUSTRALIA LIMITED (INCORPORATED IN N.S.W.)
 91 KURRAJONG AVENUE, MT. DRUITT, N.S.W. 2770, P.O. BOX 254

When our Company was informed of Greg Wilson's passing on June 8, 1984, great personal loss was felt by everyone who had known him. More than that -- a sadness pervaded the entire Tandy organisation to the point where even now, as the shock of his death subsides, people who never knew the man are deeply aware that their Company lost a friend and advocate, a true computer buff who became a leader of their industry.

Tandy's relationship with Greg began many years ago and grew steadily out of a common cause we shared. He was indeed a champion of that cause.

He brought with him honesty and always called "a spade a spade" but, to be honest, never without first hearing both sides of the story. If afterwards he still disapproved, Greg objected -- frankly and in no uncertain terms -- anyway.

To do less than speak truthfully and forthrightly about Greg Wilson here would be an injustice to a fine man. His dedication could border on obsession, his enthusiasm mistaken for egotism. He could be stubborn and, let's also be frank, obstinate. He had faults as we all have faults. His can be easily forgiven.

Greg was a hard worker, a smart worker, for a selfless cause. But foremost he was a gentleman, a man of his word, and anything divulged to him in confidence remained precisely that way.

Greg's many virtues, and even his faults, were a tremendous contribution to the microcomputer industry whether everyone liked it or not. He was misunderstood by many, but remained dedicated to his cause. His opinions were often unsettling, but always respected. He sought neither fame nor fortune, and was rewarded with friendship and admiration. He was a good man, and left behind a legacy that will continue to grow. He said farewell far too soon, and he will never be forgotten.

From the Management and Staff of Tandy Australia Limited.

Some of the colour has gone from the COLOR COMPUTER. The brightness that was GREG is gone.

The AUSTRALIAN RAINBOW, COCO OZ and the Australian wide network are all GREG. A part of him is imbedded in all of them.

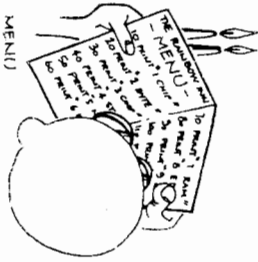
I never met him face to face but over the last three years we had many phone conversations. He could fairly rant on at times but didn't expect you to agree with all his ramblings. He expected you to go out and find out for yourself. I'll miss those conversations. GREG was more than just a voice on the end of a telephoneHe was a friend.

Lets keep the network wide open as a fitting tribute to the man who did all the spadework.....
 GREG WILSON

We'll all miss him

ROBBIE DALZELL

MEET CONTACTS



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

Adelaide	John Haines	08 278 3560	Alaffra	Max Huckerby	051 45 4315
Adelaide Nth Stvn	Leo Eisenberg	08 250 6214	Naitland	Lyn Dawson	049 49 8144
Albury	Ron Duncan	060 43 1031	Melbourne	Jeff Sheen	03 528 3724
Fairnsdale	Colin Lehmann	051 57 1545	Melton	Mario Gerada	03 743 1323
Ballarat	Mark Bevelander	053 32 6733	Mildura	Doug Matthews	050 23 5701
Bankstown	Ken Hayward	02-759-2227	Moe	Stephen Semple	051 27 6841
Blacktown	Keith Gallagher	02-626-9936	Norwell	George Francis	051 34 5175
Blaxland	Bruce Sullivan	047-39-3903	Mt Isa	Tom Parkhouse	077 43 7622
Bowral	Max Settridge	048 83 5203	Mudgee	Brian Stone	063-72-1958
Brisbane East	Barry Cawley	07 390 7946	NambuccaHds	Wendy Peterson	065 68 6723
Brisbane Nth	Jack Fricker	07 262 8869	Newcastle	Lyn Dawson	049 49 8144
Brisbane Sth	Patric Simonis	07 209 3177	Nowra	Roy Lopez	044 48 7031
Brisbane SW	Graham Butcher	07 376 3400	Parkes	David Small	068 62 2682
Brisbane West	Brian Dougan	07 30 2072	Penrith	Tom Lehane	047-31-5303
Bundaberg	Jim McPherson	071 72 8329	Perth	John Christou	09 344 6745
Camberwell	Tony Baldwin	03 728 3676	Port Macquarie	Ron Lalor	065 83 8223
Campbelltown	Leo Ginley	02 605 4572	Port Noarlunga	Rob Dalzell	08 386 1647
Canberra	Shaun Wilson	062 51 2339	Port Pirie	Kevin Gowan	086 32 1368
Caulfield	Jeff Sheen	03 526 3724	Ringwood	Andrew Rawlings	03 726 6521
Chatswood	Bill O'Donnell	02 411 3336	Rockhampton	Keiran Simpson	079 28 6162
Churchill	Geoff Spowart	051 22 1389	Rockhampton MiCo	Tim Shank	079 28 1846
Collinsville	Tony Evans	077 85 5858	Roseville	Ken Uzzell	02 467 1619
Colyton Teens	Dwayne Manson	02 623 5805	Sale	Bryan McHugh	051 44 4792
Cooma	Sheila Hamill	064. 2.3905	Singleton	David Nichols	065-73-1222
Dandenong	Brett Cruickshank	03 798 5406	Springwood	David Seamons	047 51 2107
Darwin	Brenton Prior	089.81.7766	Sturt	Mary Davis	08 296 7477
Deniliquin	Wayne Patterson	058 81 3014	Sunbury	Jack Smit	03.744.1355
Dubbo	Graeme Clarke	068 89 2095	Sutherland	Ian Annabel	02 528 3391
Emerald	Carol Cathcart	059 68 3026	Swan Hill	Barrie Gerrard	050.32.2836
Forster	Gary Bailey	065 54 5029	Sydney East	Bob Jones	02-331-4621
Frankston	Bob Hayter	03.783.9748	Sydney Teens	Rod Hoskinson	02 48 5948
Gippsland Sth	Pat Kermode	056 74 4563	Tamworth	Robert Webb	067 65 7256
Gold Coast	Graham Morphett	075-51-0015	Tongala	Tony Hills	058 59 2251
Gosford	Peter Seifert	043 32 7874	Townsville	John O'Callaghan	077 73 2064
Grafton	David Hulme	066.42.0627	Traralgon	Morris Grady	051 66 1331
Greenacres	Betty Little	08 261 4083	Upper Hunter	Terry Gravalin	065 45 1698
Hobart	Bob Delbourgo	002 25 3896	Wagga Wagga	Bruce King	069 25 3091
Ipswich	Milton Rowe	07 281 4059	Westleigh	Athalie Smart	02 848 8830
Kenmore	Graham Butcher	07 376 3400	Wollongong	Brian McCauley	042 71 4265
Lithgow	Stuart Rayner	063 51 4214	Wonthaggi	Pat Kermode	056 74 4563
Liverpool	Leonie Duggan	02-607-3791			
MacKay	Len Maloney	079511333x782			
Macleod	Robin Ziukelis	03 450211x465			
MacquarieFields	Keith Roach	02 618 2858			

Registered by Australia Post —
 Nos. NBG5033, 6279 & 6280X
 Greg. Wilson, PO Box 9, Potts Point NSW 2011

Remember . . .

Address all mail to:

AUSTRALIAN RAINBOW

P.O. Box 1742, SOUTHPORT. 4215.

or PHONE GRAHAM MORPHETT,

on 075-51-0015.

