For your TANDY Computer

$3.25

AUSTRALIAN

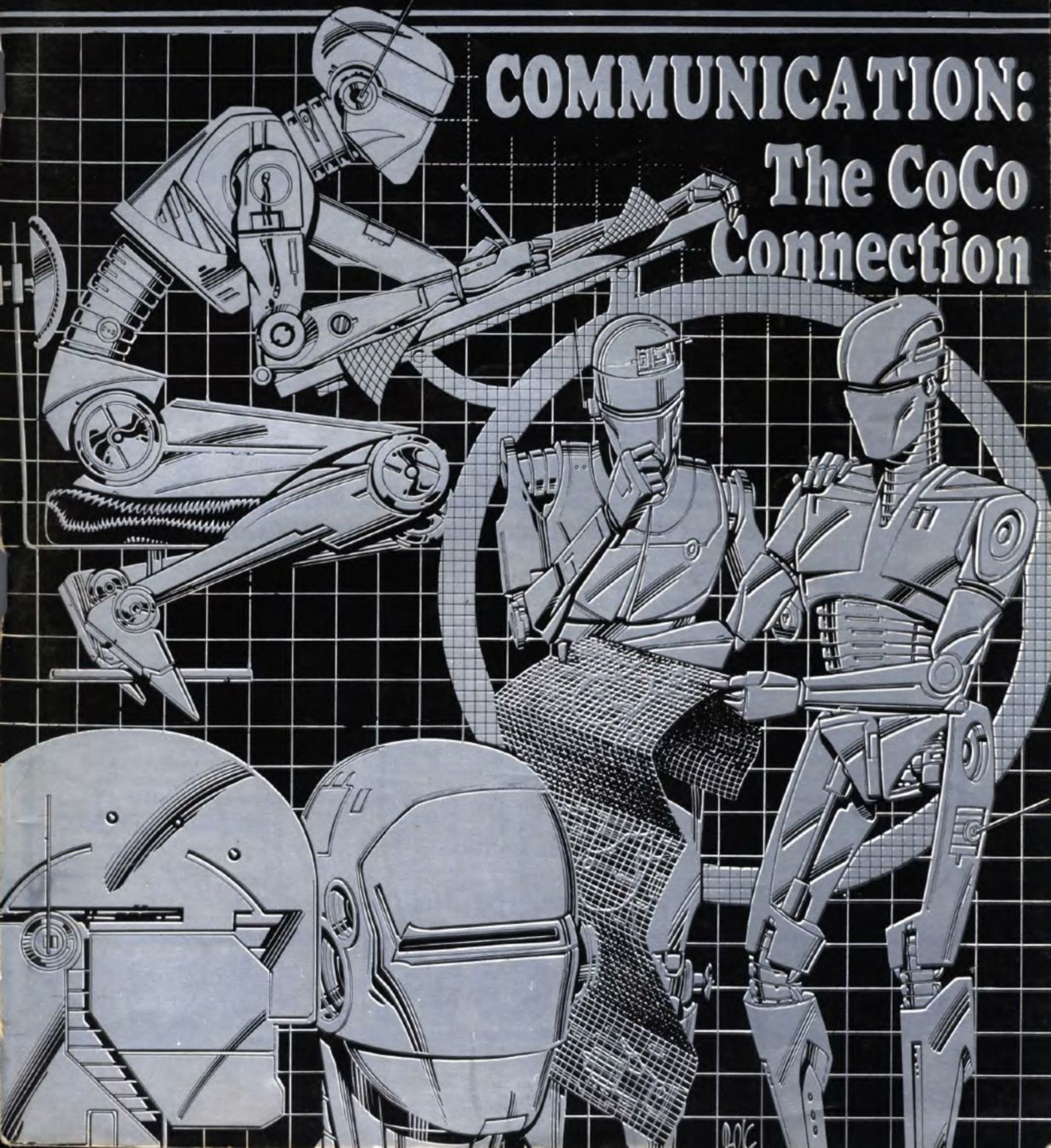
RAINBOW

INCORPORATING

GOCO

December, 1984 January, 1985

No.42 43



COMMUNICATION: The CoCo Connection

128K RAM CARD

Price not available at press time

INCREASE YOUR 64 K Co-Co OR Co-Co II TO 128 K RAM

- FITS COMPLETELY INSIDE YOUR COMPUTER
- SWITCHES TWO NEW 32 K BANKS OF RAM IN AND OUT OF MEMORY.
- BANKS CAN BE MAPPED IN THE UPPER HALF OR LOWER HALF. OR CAN ALSO BE A SECOND COMPLETE 64 K BANK.
- SWITCH TABLES INCLUDED.
- SIMPLE INSTALLATION AND DOCUMENTATION
- A MUST FOR OS-9 USERS.
- COMPLETE WITH 8 (4164) RAM CHIPS.
- PAL CHIP HANDLES ALL BANKING COMMANDS.

AGENTS:
J Fricker Qld 07 262 8869
K Enright 09 38 26472
S Wilson Act 06 25 12339
Bayne & Trembath Vic & Tas
Ian Lee South Australia
L Wrigley 75 Molesworth St
Lower Hutt NZ

OS9 SOFTWARE

Opack — Utilix — Filter Kit — SDisk & Bootfix — OS9 — Basic09 — Dynamite+ — Color Sleuth — RMS (Record Management System) — Stylograph Mailmerge — Spelling Checker Accs Receiveable and Payable Inventory (Control system for retailer and distributor)

PHONE NOW

Blaxland Computer Services

(THE COLOR COMPUTER SOFTWARE SHOP)

P.O. Box 125 BLAXLAND 2774

047 39 3903

RAINBOW BITS

supplier of the latest hardware add-ons for CoCo

CoCo MODEM

AUTOMATIC $295 STANDARD $250

1. Telecom approved to 300 baud bi-directional modem
2. Plugs into Rom port ex expansion pack to work with disk (Specify Tandy controller or other)
3. Inbuilt terminal programme on Eprom (no loading)
4. Automatic model allows CoCo to dial-up remote computer/B B S
5. Requires no power pack. Works with all models of CoCo
6. Leaves printer port free for continuous down load to printer

Y CABLE $30
Permits simultaneous connection of modem and disk.

4-SLOT EXPANSION PACK Kit $65 (requires soldering) Made Up $100
Enables disk controller, modem, 80 column card eprom programmer, rom packs etc to be run/selected at one time

CASE CHANGER
LOWER CASE BOARD made and tested $60 (simple installation)
gives true lower case (with descenders) on screen not reverse video

VIDEO AMP CARD $25 (simple installation) for driving video monitor

all prices include P & P.

Write to 17 Penley St, The Gap. Q. 4061

RAINBOW BITS

AUSTRALIAN RAINBOW GOCO

Founder Greg Wilson

AUSTRALIAN EDITOR AND PUBLISHER
Graham Morphett

CO-EDITOR
Kevin Mischewski

EDITOR'S ASSISTANT
Sonya Young

AND GRATEFUL ASSISTANCE FROM
Brian Dougan
Peggy Annabel
Richard and Judy
Bob Thompson
Paul Humphries
Alex Hartmann
Andrew Simpson
Jim & Sheryl Bentick
Patric Simonis
Annette Morphett
Glen Mischewski

COVER DESIGN
Jim Bentick

All Programs in this issue of RAINBOW are available on cassette tape

DEADLINES

Feb 7th Jan '85
Mar 7th Feb '85
April 7th Mar '85
May 7th April '85
June 7th May '85
July 7th June '85

OS-9

Kevin Holmes is the contact for OS-9 information. He also has access to OS-9 Software from the U.S.
His adress is:—
39 PEARSON ST.,
NARARA, N.S.W. 2250

Printed by:
Australian Rainbow Magazine
P.O. Box 1742
Southport Qld. 4215

Reg'd Publication QBG 4007

PRINT #-2,

All of us at Rainbow hope that this Christmas will be a special one for you and your family.

We make this magazine because we care what happens in the world. We hope that, through the magazine, we can show that the love that is seen more often at Christmas can be a part of our lives right through the year.

Who needs all that weaponry ... the wars ... only the politicians and the generals!

Me, I'd rather sit on the beach with my wife and daughter, dig holes in the sand, and think of dumb things to write about in this magazine!

CoCo takes you wherever you want to be. If you need to be aggressive, you can shoot aliens from the other side of the black hole. If you need to obtain an insight into the mathematics of the solar system, CoCo will assist, if you want to teach your child anything from Bible stories to maths, music to programing (in several different languages), CoCo assists.

A glance at what we have in store for you in this magazine will show that in many ways, we are still only scratching the surface of CoCo's abilities.

In short, there isn't anything that can't be done in time with CoCo, yourself and preferably, a friend.

If you are not reading our magazine for the first time, you should know that this magazine is created by a group of friends, for a group of friends. It's a team effort, and we all take a great deal of personal pride in what we do here.

However, we traditionally get a whole new batch of readers at this time every year and to these folk we say welcome, hope you get a lot of pleasure from your computer and our magazines.

All of the articles are written by subscribers like yourself, who are kind enough to share their many talents with you. The articles are donated, and that makes them very special, particularly when you see the variety and quality each month.

INDEX

uuuuser groupppppp ddCONTACTSSSSS

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

ADELAIDE	JOHN HAINES 08 279 3568	
ADELAIDE NTH	STAN EISENBERG 08 258 6214	
ALBURY	RON DUNCAN 060 43 2031	
BAIRNSDALE	COLIN LEHMANN 051 57 1545	
BALLARAT	MARE BEVELANDER 053 32 6733	
BANKSTOWN	KEN WAYWARD 02 759 2227	
BLACKTOWN	KEITH GALLAGHER 02-627-4627	
BLAXLAND	BRUCE SULLIVAN 047 39 3703	
BOWEN	TONY EVANS 077 86 2328	
BRASSALL	BOB INSWORTH 07 201 8459	
BRIGHTON	GLENN DAVIES 08 296 7427	
BRISBANE EAST	BOB THOMPSON 07 848 5512	
BRISBANE NTH	JACK FRICKER 07 262 8849	
BRISBANE STH	PATRIC SIMONIS 07 209 3177	
BRISBANE SW	GWORN BUTCHER 07 376 2488	
BRISBANE WEST	BRIAN DOUGAN 07 38 2872	
BUNDABERG	JIM McPHERSON 071 72 8329	
CAMBERWELL	TONY BALDWIN 03 728 3476	
CAMPBELLTOWN	LEO GINLEY 02 605 4572	
CANBERRA	SHAUN WILSON 062 51 2329	
CARLISLE McCo	STUART HALL 361 1922	
CAULFIELD*3526	JEFF SHEEN 03 509 3724	
CHATSWOOD	BILL O'DONNELL 02 411 3336	
CHURCHILL	GEOFF SPOWART 051 22 1389	
COLYTON TEENS	DWAYNE MANSON 02 628 5845	
COOMA	SHEILA MAHMILL 064..02..3195	
DANDENONG	BRETT CRUICKSHANK 03 547 6644	
DARWIN	BRENTON PRIOR 089.81.7766	
DENILIQUIN	WAYNE PATTERSON 058 81 3814	
DUBBO	GRAEME CLARKE 068 89 2875	
EMERALD	CAROL CATHCART 059 68 3826	
FORSTER	GARY BAILEY 065 54 5829	
FRANKSTON	BOB WAYTER 03.783.9748	
GIPPSLAND STH	PAT KERMODE 056 74 4583	
GOLD COAST	SHERYL BENTICK 075-39-2093	
GOSFORD	PETER SEIFERT 043 32 7874	
GRAFTON	DAVID HUME 066 42.8627	
GREENACHES	BETTY LITTLE 09 261 4883	
HOBART	BOB DELBOURGO 002 25 3876	
IPSWICH	MILTON ROUE 07 201 4859	
JUNEE	PAUL MALONEY 069 23 3868	
KENMORE	GRAHAM BUTCHER 07 376 2488	
LEETON	CHRIS NAGLE FARM 749, MUSCAT ST AMESBURY, LEETON 2705	
LITHGOW	STUART RAINER 063 51 4214	
LIVERPOOL	LEONIE DUGGAN 02-487-2791	
MACKAY	LEN MALONEY 079511333x292	
MACLEOD	ROBIN ZIUKELIS 03 4582[illegible]x445	
MacQUARIEFIELDS	KIETH ROACH 02 618 2458	
MAFFRA	MAX HUCKERBY 051 45 4315	
MAITLAND	LEN DAWSON 049 49 8144	
MARYBOROUGH	PETER MILLER 071 22 1495	
MELBOURNE	JEFF SHEEN 03 509 3724	
MELTON	MARIO GERADA 03 743 1323	
MILDURA	DOUG MATTHEWS 050 23 5781	
MOE	STEPHEN SEMPLE 051 27 6841	
MORWELL	GEORGE FRANCIS 051 34 5175	
MT ISA	PAUL BOUCKLEY-SIMMONS 077 43 4288	
MUDGEE	BRIAN STONE 063-72-1958	
NAMBUCCA HDS	WENDY PETERSON 065 68 6722	
NEWCASTLE	LEN DAWSON 049 49 8144	
NOWRA	ROY LOPEZ 044 48 7431	
PARKES	DAVID SMALL 068 62 2682	
PENRITH	TOM LEWINS 047-33-5383	
PERTH	JOHN CHRISTOU 09 344 6745	
PORT MacQUARIE	RON LALOR 065 83 8229	
PORT HOPKINGS	ROB DALZELL 08 386 1447	
PORT PIRIE	KEVIN GOWAN 086 32 1568	
RENGALIC	WARREN RAWLINGS 03 776 6521	
ROCKHAMPTON	KEITARH SIMPSON 079 29 [illegible]	
ROCKHAMPTON McCo	TIM SHANK 079 28 [illegible]	
ROSEVILLE	KEN GITZELL 02 467 [illegible]	
SALE	BRIAN MURDEN 051 44 [illegible]	
STANSBURY	DAVID NICHOLS 865 [illegible]	
SPRINGVALE	DAVID SEAMONS 047 51 [illegible]	
STUART	MARY DAVIS 08 276 7427	
SUNBURY	JACK SMIT 03.744.1355	
SUTHERLAND	IAN ANNABEL 02 528 3391	
SWAN HILL	BARRIE GERRAND 050.32.2030	
SYDNEY TEENS	ROD HOSKINSON 02 48 5948	
TAMWORTH	ROBERT WEBB 067 65 7256	
TONGALLA	TONY WILLIS 058 59 2251	
TOOWOOMBA	MIKE CARTER 076 35 6911	
" BEGIN NTH	DAVID PROUT 076.32.7533	
" BEGIN STH	LEV GERSEKOWSKI076 35 8264	
" ADVANCED	GRAHAM BURGESS 076 38 4259	
TOWNSVILLE	JOHN O'CALLAGHAN 077 75 2644	
TRARALGON	MORRIS GRADY 051 66 1331	
UPPER HUNTER	TERRY GRAUDLIN 065 45 1490	
WAGGA WAGGA	BRUCE KING 069 25 3871	
WESTLEIGH	ATHALIE SMART 02 840 5838	
WYALLA NORTH	CHRIS HUNTER 086 45 3395	
WOLLONGONG	BRIAN McCAULEY 042 71 4265	
WONTHAGGI	PAT KERMODE 056 74 4583	

009 Bulletin Board System:
" 9pm to 1am MOUNTAIN, 9am to 4pm VOICE
" SYSOP BOB THOMPSON 047 38 2468

CoCoConf

GOLD COAST QUEENSLAND

JUNE '85.

PLAN TO BE THERE!

And speaking of friends, I know you will all want me to pass on to Johanna Vagg our collective congratulations on the birth of Katrina Lee, an 8 lb bundle of CoCo fodder for whom we've been waiting for what seems like years! Johanna is one of our regular authors, who despite the fact that she was about to have yet another child, still managed to get an article to us this month!

As our friends know, we've been stuffing up since mid year. Late magazines, late tapes and so on. BUT ... ta da ... we now have the Data Base under some form of control; as best as we can tell, we are now up to date with magazine mail out, and we have a big push on to complete all tapes before the end of 1984!

So let me introduce some of the team. I wasn't able to get the team from Brisbane down when I needed them, to get a more comprehensive photo, but then I also missed getting in the folk in Sydney, Melbourne, Hobart, Adelaide, Perth and many, many stops in between too. The photo shows a few of the more usual faces from around here, but as I have said, there are others ...

On the left, Jim and Sheryl; better friends it would be hard to find. Tireless workers, they have two little girls who didn't make it to the photo because they were asleep at the time!

The smoothy with the glasses, standing two feet above the rest of us is Kevin. Kevin, like Jim has a job elsewhere; when he finishes, he goes home to do a further eight hours for us! Kevin is ably assisted by his brother Glen, whose head appears just above Annette's (left) and Sonya's (right).

Annette is the finance wizz around here. She tells us what we can't buy. She is totally to blame for the fact that we are up to date. She works very hard and still manages to be an excellent mother.

Sonya is our most recent sucker friend. She has another job part time but will work with us full time next year. Sonya does good work.

Then there's me. That may look like a shadow from my hat, but it's really just one big black circle under my eyes.

Then there's Katie, the light of my life!

And finally, behind us all, is the Southport monument to better times.

Some interesting things are happening. The BIG news is that you can now obtain upgrades to 128K. As with 64K, this extra memory can not be used by CoCo's ROM Basic (or ECB).

However OS-9 can access the extra memory, as can (I think) Flex and perhaps one or two other systems. (You will recall from the article in last month's Australian CoCo, that OS8 provides it's own 128K upgrade.)

The sad news is that the latest CoCo has been having trouble with some TVs (no color). Rank Arena in particular seems to be a cause of concern. The problem is relatively easy to fix, and will not persist, I'm told....

Gathering strength are the voice packs - whether they come from Tandy, Software Spectrum, or the new REAL TALKER from Blaxland Computer Services, people are snapping them up! They are certainly a lot of fun, and will lead to a new approach to some Educational Software.

I believe that REAL TALKER puts a moving face on the screen as it talks to you - more next issue!

When we took over, one of the initial reasons for spliting the magazines was to be able to hold the price. The last thing I want to do is to raise the price - it hasn't risen in three and a half years. But its gonna have to happen. I looked at various "sweetheart" deals for old readers, the best that I can come up with is this:

If you are a current subscriber, or you can prove that you are a long term purchaser of the magazine, regardless of when you are due to subscribe, if you want to send along $28 per subscription for either RAINBOW or CoCo, or $56 for both, AND the money reaches here by 15th January, then we'll add 12 months to your current sub.

That's about the messiest way I've ever seen a business put its rates up, but then we're more a charity than a business anyhow!

There are obvious reasons this month why both magazines have the same material in the Screen Dump / Print#-2 columns, in fact it may happen every Christmas! Next issue the Magazines will revert to their more usual formats. RAINBOW seems to becoming a magazine for the specialist hobbiest, whilst CoCo reflects what you send!

Everyone who sent for Tee Shirts / Dresses has had their orders filled and posted. I'm sure that you'll like them!

"Faces at the Window" adds an additional dimension to CoCo this month. Fiction is something we get enough of from Tandy at times, but I thought that on this occasion, you might appreciate a little light reading over Christmas!

Finally, in case you didn't realise it, we've doubled the size of this issue. That's so we can have a holiday and just perhaps, not think about computers for a week or so. Annette, Katie and myself have planned a week in Sydney with our families, Sheryl, Jim and Co have a mad friend coming up from Sydney, Kev has to work, and Sonya is also off to Sydney. I'll have a good Christmas if those tapes are finished by then.....

HARDWARE

VIDEO AMPLIFIER

from PARIS RADIO ELECTRONICS

Having recently purchased a monochrome (green screen) video monitor I soon learned that simply taking the composite video signal directly from the Video Display Generator or the Modulator is not really enough. What is needed is some sort of video amplifier.

Paris Radio Electronics in Sydney build and sell one such board and needless to say it works perfectly.

Installation involves the simple soldering of three colour coded wires to the integrated circuit MC1372 and mounting the amplifier atop the Video Modulator. The soldering required is a little delicate (especially in the older "E-Board" machines, such as mine, which have a conglomeration of jumpers to modify the T.V. display to the Australian P.A.L. system.) but after a few test runs on some scrap components should give the average handy-person no problems. Of course, there must be someone at your local meet who is willing and able to carry out the installation if you have any doubts as to your own ability.

The amplifier comes with a one meter cable attached ready for you to plug directly to the socket on your monitor. After a quick adjustment of the contrast and brightness controls, there you have it. The crisp clear image you were dreaming of when you brought home your monitor.

PARIS RADIO will also have availiable shortly a Colour Composite Video amplifier which includes an audio output line.

The MONOCHROME VIDEO BOARD will work on all CoCo models and is availiable for $35.00 from:

PARIS RADIO ELECTRONICS
161 BUNNERONG RD. KINGSFORD 2010
Ph. 02 344 9111

BMC BM-12EA Monochrome Video Monitor

With the prices of video monitors starting to drop to reasonable levels I thought that it was about time I purchased one for my own use. So the shopping started. Most of the lower priced monitors are availiable from the more reputable mail-order electronics suppliers and for the record I purchased this particular monitor from MaGRATHS new Brisbane store. I finally settled on this monitor after seeing one up and running at one of the Brisbane meets (The meet groups are always the best venues to see any developments in hardware or software).

The BM-12EA is a Green Phosphor monochrome monitor and comes in a stylish white case. Some of the more technical specifications are:

CRT Size.....31 cm diag (12 inch)
Video Amp Bandwidth.....18 MHZ
Display Format..1920 Characters Maximum
..80 Char x 24 Lines

I did find it necessary to purchase also a video amplifier. While the monitor will still work by simply taking the composite video signal directly from the Modulator, the picture size was found to be somewhat smaller than it should have been. Naturally enough you cannot plug the monitor directly into the normal T.V. socket on the computer and since it is necessary to make up a cable and socket of some type anyway, why not do it properly and pass the signal through an amplifier such as that supplied by PARIS RADIO.

The result of all this labour has been the achievment of a superb picture at a fraction of the cost of a decent colour T.V. There is one tradeoff you must face when using a monitor, and that is you no longer have sound. I have found there are three ways to overcome this. If overkill is your thing, then why not simply connect your old T.V. into the computers normal T.V. socket. It's quite effective to have two pictures running side by side with one another and besides if your T.V. is colour then you also have the choice of watching either your hires monitor or colour T.V. The second and most sensible option is to install your own audio amp and speaker under the keyboard. When we have more

details we'll tell you how to do this. The third and best temporary option that works on my computer is to simply plug in your cassette recorder, remove the black plug from the recorder, press play and record (you may need to have an old tape in the recorder) and wind up the volume. Any sound output is redirected to the cassette recorder speaker (don't ask me why).

Using the monitor has been a dream. I spend between six and twelve hours a day in front of my computer and I can catagorically say that the green phosphor of the monitor is far easier on the eyes than even a good colour T.V. the clarity of the picture is a vast improvement. There is no perceptible screen movement, but I can detect the slightest trace of background flickering when the brightness is wound up too high. The contrast and brightness controls give ample adjustment for all viewing conditions (bright daylight or dull night-light).

One other thing to bear in mind when purchasing a high resolution monitor. The resolution of your display is not determined by the monitor but by the computers Video Display Generator. You will not get more resolution but what you will get is better clarity at any resolution. To fully utilise your monitors capabilities it is possible to drive the video display through an alternatively VDG such as that used by the WORDPAK 80 card. This will give you (among other things, see PARIS RADIO advert) a true hires 80 X 24 display.

One of the strongest features of the CoCo is the way it lends itself so readily and inexpensivly to upgrading to the level (software and hardware) of the up-market machines.

GAMES

WORLDS OF FLIGHT

Tom Mix Software

For some months now I have been eyeing the adds for WORLDS OF FLIGHT in the U.S. magazines and finally it's availiable locally through SOFTWARE SPECTRUM and their agents.

Flight simulator programs face the problem that they must measure up to the real aeroplane, not an easy task. Aircraft have such a variety of forces acting on them at all stages of the flight. On top of that we all want to be able to see the view out of the window.

With WORLDS OF FLIGHT, Tom Mix Software have produced an excellent compromise between all of these factors. The view you get is that of line graphics scenes in nine different worlds. You may fly from world to world or just play around with aerobatics. The mountains are quite real and if you hit one you certainly crash. In radar mapping mode you may view your location over the ground, the easiest way to navigate. This view may be enlarged or reduced at will.

This is one program where you must read the manual before "taking off". You need to know the layout of the controls and some basics about the performance of the aeroplane. Isn't that just like the real thing?

After setting up your starting world and weather conditions you are parked at the end of the runway, engine stopped heading north. The runway is in view ahead of you with the tower on the left and a hanger a bit further down on the right. All in perspective of course. Start your engine, carry out your pre-takeoff checks. Full power, release brakes, keep straight (crikey, a taildragger is easier to keep straight than this) 60 knots, stick back now we're flying, gear up, watch your airspeed, watch your heading. What 500 feet already, I haven't even got the flaps up, turn onto downwind, NOT TOO STEEP, this is a major airport not your local cabbage patch, watch your airspeed, whoops turned too far, try and kick it back with the rudder, won't come round fast enough. WOW!! A couple of minutes to spare on downwind, lets look at the view WHAT!! How did I get that high? Oh well lets see where we are with the mapping radar. Its nearly time to set up our approach, gear down, bit of flap, power back, airspeed 70 knots turn onto final, what do you know I even managed to line it up, there's the

runway, reckon I'm a bit high, bit more flap, bit more flap, still too high, full flap, what now I'm too low, bit more power, what's that noise Oh No!! too slow, stick forward, too low full power go round, engine over-reving stick back some more, Is that the runway lights?? What do I do?? CRASH!!!!! Lots of chaos and noise. I guess I must be dead, Oh well, I need a coffee, did that take 15 minutes, it seemed like only 2 or 3.

So ends my first flight. The controls do take a bit of getting used to, knowing how much to lead your turns and what speeds etc to fly at. But everything works correctly. Any pilot would find WORLDS OF FLIGHT a good test for his normal flying skills, the thinking pattern is so similar.

I found it was easiest to navigate using the radar map and fly the aircraft using the instruments, primarily the artificial horizon. Transition to the runway view at about 2 to 300 feet seems to be about best, finally landing the plane using the view out the window. If you do well you will hear the comforting squeek of the nosewheel touching down at which time you may taxi to the refuel point to ready for another flight.

So ended my first flight. I'm sure you can have as much fun. WORLDS OF FLIGHT performs well and provides a challenging simulation for all you armchair pilots. Go to it!

WORLDS OF FLIGHT is availiable from:

SOFTWARE SPECTRUM
P.O. Box 2101, Adelaide
SOUTH AUSTRALIA, 5001

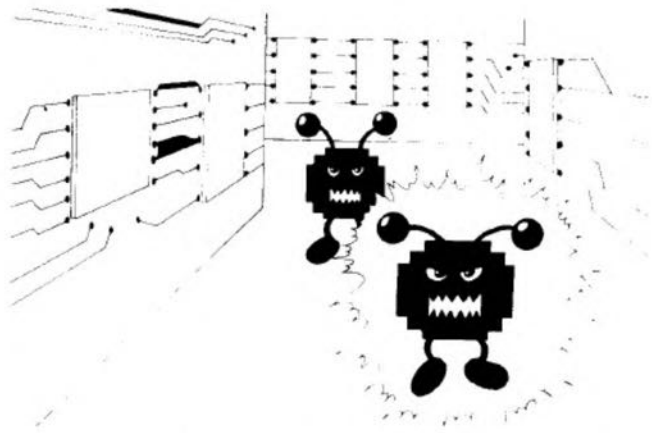
Ph. (08) 211 8763

ANDRONE

I must admit to never having played 3D maze games, indeed never having seen one this glaring lack on my part becomes understandable, but if Androne gives any indication of the standard of this type of game...goodbye Donkey King.

Androne is a remote controlled Robot in search of 'Data Bugs' that fly around the various levels of a 3D maze.

As you wend your way around the corridors you come across the dreaded 'bugs' that hurtle themselves at Androne sapping his energy



After sighting and despatching said bugs, continue on in search of the power units, once found escape via the elevator, to the next level.

There are visuals to help you along your merry way. A direction indicator (N,S,E,W) and a map (without the maze) showing the location of each power unit, elevator and our little friend Androne.

The joystick controls Androne's movement and the firing of his weapon. The score is displayed at the bottom of the screen, you receive points for each bug, power unit etc that you 'take'. Points increase as you proceed through each level, as does the level of difficulty!

You will really enjoyed this one, good graphics, easy to get into and best of all it's for the CoCo.

Androne comes from Tandy in Rom Pack, requires 16K and is priced at $34.95.

LETTERS

Dear Graham,
I am a new subscriber to Australian Rainbow and really enjoy the magazine a lot.

I have a 64K ECB CoCo and a DMP-100. I recently purchased Radio Shack's Hi Res Screen Print Utilities (Cat No 26-3121). I think I may have discovered a bug in the program BUDUMP. When I get a print out of a graphics screen, the right and left edge of the screen are not printed even though all of the picture is displayed on the TV screen. Do you know of a fix, as the local Tandy bloke could not help.

A couple of interesting POKES that I have found are:

POKE 65313,4: MOTOR ON
POKE 65213,52: MOTOR OFF

POKE299,0:POKE300,0. DISABLES ECB
POKE299,25:POKE300,14. RESTARTS ECB
NB, RESET does not restore ECB.

EXEC41175 displays a different computer message.

Keep up the good work,
Michael Robinson.

Dear Michael,
I do not know the program to which you refer, so we will throw this one to our readers - perhaps they will help!

Thanks for the POKES. As you say, they are always interesting.

Graham.

(Michael is one of a number of you who received two lots of magazines last month. We appologise for any inconvenience caused. If you still have magazines in your possession due to a similar error on our part, don't return them. Use them to improve your relationship with Tandy, take them to your nearest Tandy shop and let them know that they can have the magazine/s with our compliments.)

Dear Graham,

I want to thank you and your many helpers and friends for taking over the Australian Rainbow. I find it great reading.

I only purchased my 64K CoCo a few months ago and have since eagerly studied both manuals. I pondered a long time and quite deliberately decided the Tandy CoCo was the best and safest bet in a terribly shaky world of the personal computer market. Everyone who has one 'crows' about them and every day we hear of more and more feats it can perform. I love it.

I'm as green as can be when it comes to programming but as keen as mustard. there simply isn't enough hours in the day to spend on the keyboard.

Joining Australian Rainbow readers was a good move. It filled a gap, - a link with others, that all beginners need. I purchased a few back copies and quickly saw the value of being a subscriber ('its cheaper'-G.), I only wish I'd started long ago.

Though I'm learning as fast as I can, I'm anxious to produce some good examples of BASIC achievements to show off my CoCo and 'prove' it's as good as, if not better than the Commodores and Ataris the kids have been captured by elsewhere.

I am at the Administrative Offices at Kensington Park College of TAFE. Consequently I'm already aware of the Word Processing, Spreadsheet and Data Base packages available for Micros. I purchased 'VIP WRITER' and 'VIP CALC' from Software Spectrum. I use them on my CoCo2 with the CCR-81 tape recorder and the College Microlipe 84 dot matrix printer. My budget is run completely on VIP Calc now. A single sheet (MATRIC M60) does all and provides up-to-the-minute balances and how much to bank each fortnight. It is truly a brilliant package for $63 odd! Multiplan etc. cost hundreds and VIP Calc does just about all they do. (Wasn't so impressed with Tandy's 'Spectaculator' however - too limited in it's uses and costs just as much).
There is a 'bug' in VIP Calc however. I believe 'Hot CoCo' wrote something on this subject (grape vine info only). If you or your readers could produce an article on the VIP Calc, particularly about the 'bug' in it and maybe what to do about it, I would be grateful and maybe others too. My experiance so far is that using '@SUM (L6:L40) to find the total of a column of 36 formula based row totals, produces a 1c error. I can live with that but it is annoying. Just this week another 'bug' cropped up in the same column but was considerably more inaccurate. I can't see anything I've done wrong. The row formulas are simple. The column L6 to L40 is made up of totals of rows 6 to 40, each having the formula (Fn + Gn + Hn + In) - (Jn + Kn). I replicated this formula (with figures of course) at L6 from L7 through L40. Each row total is correct but the column total has been 1 cent out. With a negative balance in L12 I banked some money into that account and entered the figure in I12. L12 went to $.00 which is correct. However the column total at L42 actually reduced in size by several hundred dollars. I was flabber gasted! Everything I've done checked out. I removed the figure from I12 and it reverted to its former 1 cent error. I saved, cleared ended and completely reloaded the program and data tape to find the same occurrence. So far the problem remains unsolved. I will consult Software Spectrum re a new program as I believe there is a software fault. Have you heard of any others with this problem?
Sorry this letter is so verbose. I do intend to join a local users group as well.
Kindest regards,
Allan Thompson.
O'Halloran Hill. S.A.

Allan,
Obviously I agree with all that you say re the computer, thanks for the nice words. However, don't be in too great a hurry to 'prove' your computer (or perhaps your skills); the computer's ability is well documented, particularly in comparison with the computers that you mention. Take time to learn the fundamentals thoroughly, and above all, have fun with what you achieve!

I have not had much experience with the VIP program that you mention, perhaps one of our readers can assist.

Graham.

Dear Graham,

I would like to draw the attention of readers to some errors that I have found in the programs described in PART II of the article "Cooking with CoCo" in Australian Rainbow, August 1984.

The SYSAVE and SYSTEM programs as listed do not in fact save or restore $8000 to $FF00 but only do $8000 to $EF00. This is corrected by the following changes:

Program SYSAVE line 130 and 134 should be:

```
130     CMPA #7
134     CMPA #1
```

Program SYSTEM lines 118 and 122 should be:

```
118     CMPA #7
122     CMPA #1
```

I hope this will be of some help to readers who may have been trying to use these useful utilities to store above $EF00.

Sincerely,
02-WI2,
Brisbane. Qld.

Dear Graham,

I have had a 16KECB for about two years now. I started subscribing to Rainbow in Jan.'84, and was very saddened to hear of Greg's death. Like many people, I had never met Greg, but still I felt I knew him through the magazine.

After receiving my August copy of CoCo magazine, I felt that you and your associates had done an excellent job of putting the magazine together on such short notice, although the magazine wasn't, in my opion, quite up to the standard I had been receiving. I put it down to inexperience.

Today, after receiving my November copy of Australian Rainbow, what I had hoped wouldn't happen has happened. Up until July, on average I would receive my Rainbow and get at least four, usually more, programs out of it. But now, it would appear, the average number of programs is three, sometimes less, and many of these programs are not suitable for an 18yr old person, such as myself.

I will take November's Australian Rainbow as an example. Of the programs that will fit in my CoCo, namely 'CoCoSchool Mark', 'Lo-res Graphics' and 'Create a Calender', only 'Lo-res Graphics' is mildly interesting to me. I am studying for my H.S.C., and so I should, (and do!), know my calendar and I also can spell. I realise that there are getting to be fewer people with anything less than 32K ECB, but Greg always had programs that would fit on both systems, where to me, it appears that you mainly cater for those with at least 32K ECB. The price of your magazine has gone up, yet I am getting less out of it in terms of programs.

However, there are many things with the magazine, that are better, and should be continued with. I

like how you are having a series on "Everything you always wanted to know". This is a very good idea, and has answered many of the queries I have had, but have been unable to find answers for. I also enjoy reading your software reviews.

With studying for my H.S.C. I do not have an income with money to spare on getting a 32 or 64K extension for my computer. As it is, just general items, like pens for my printer, set me back a few weeks.

At this stage, I am considering terminating my membership when it expires in January, and buying up on past issues of Rainbow before I joined up. Could you possibly tell me what issues are available at the moment? I would much appreciate it.

Although it may appear that I am nothing but a grumbler, I do not mean to come across that way. All I am trying to do is get all of my grumbles off my chest, and I believe that you are the person to tell.

I hope that this letter is taken notice of, for I do not really want to cut off my major source of programs for my CoCo. I hope I am not asking too much.

Yours faithfully,
E. Wright
BORONIA. Vic.

Dear Elton,

I'm sorry that you feel the way you do about the magazine. Unless you are buying the magazine at the Newsagent, or getting a Biannual subscription, the price has not, hitherto gone up. In fact considering that you received 'CoCo' free for two months, if anything, the price has been lower. As explained elsewhere, the price is about to rise, but even now, as explained, you can get out of paying the higher price in the short term.

To a certain extent, both CoCo and Rainbow's content is dictated by what we receive from users such as yourself. If we don't get games, we can't print them.

I believe that what you are saying, on the other hand underlines the urgency of us getting to where we want to be with both mags - you see we aren't totally satisfied yet either.

I would suggest however, that it sounds like "CoCo" might be the magazine more suited to your needs.

Australian Rainbow will always be tied to American Rainbow's apron strings. If they have an Educational or Graphics issue, then our's will lean that way too. Rainbow will always tend to be 'state of the art', and fairly heavy - the magazines to which you refer (which, by the way are still available from us) as I recall, were pretty heavy stuff to us in those early days too!

But CoCo depends on you and others like you to submit work for publication. If nothing comes in the mail, we haven't got a magazine.

Conversely, where we have a choice, we try to mix as equally as possible, the different types of programs.

Finally, if all else fails, and the programs in the front half of CoCo dont suit, there's always the MiCo section in the rear from which to convert programs! They should nearly always fit your machine.

Trust this will help you understand the differences.

Best wishes
Grahan.

EDUCATION PAGE

It's been a busy month! Working in several meetings and projects specifically related to Education, proved quite difficult, but as usual, very rewarding.

With Mike Lehmann and Leo Wilson from Tandy, I spent an interesting morning recently talking to the Queensland Schools Computing Services Branch.

This branch publishes "SUGAR", a monthly Educational Software report, which discusses software deemed suitable for Queensland schools. We were somewhat mystified by the lack of space given to CoCo in this publication, and so part of the reason for spending time with these folk was to demonstrate the fine range of Educational Software available for CoCo.

In fact, there is probably as good a range of useful Educational programs available for CoCo, as there is for any computer currently available.

(Linda Wilkinson is the person to contact if you want to see 'Sugar'. Her phone number is 07-224-6254. Linda has a tough job - she has to keep up with six computers and their software, note items of interest, and report in 'Sugar'. And I thought this was hard!)

That CoCo is well serviced with good quality Educational Software is particularly emphasised when one considers that CoCo has the Software Development Council to back up any genuine request for software that an Educator may make. The SDC undertakes to revise, develop or create programs to Educators' specifications. There's not too many other computers around with that sort of back up!

One of the programs being used in schools a lot these days is Logo. At Rainbow, we have not had a lot to do with with this language, so we decided that we'd look at it in a class room situation. Our report will be in March '85 Rainbow, (we hope!).

One of the reasons that schools have historically received bad service over the years from the Computer Industry stems from the fear that the schools would add computers to what was seen as a growing list of items that were 'fair game' for plagerists.

Mainly, of course, the fear is that programs supplied to a school will be copied and passed to student's private collections or to other schools.

To a certain extent this has happened, although the incidence of software theft is lower than first expected. CoCo software in particular seems to enjoy a 'fair go', probably because it is priced reasonably in the first place.

We all depend on each other - the suppliers need the schools, schools need the impetus that a fresh program provides. If you don't BUY software, the guy wont be there when you really need him!

Ross Spicer was kind enough to supply a list of Public Domain software available from his Wide Bay Region. These packs are for the Apple computer, but provide a guide should any of our hackers out there wish to put together something for CoCo.

SECONDARY:

- Language
  (English, German, French)
- Commercial
  (Typing, Bus. Princ., Accounting)
- Mathematics
  (Arithmetic, Graphing, Senior Maths, Algebra, General)
- Science
  (Biology, Chemistry, Physics, Simulations, Misc)
- Social Science
  (Cit Ed, Geog, Hist, Econ)
- Teacher Utilities

PRIMARY:

The Primary Starter Pack. (Simulations, Data Base, Maths, Language Arts, Music, Games, Typing, Puzzle Makers.)

Tandy in Qld once more made it onto the Contracts List for the coming year. (Don't know what's wrong with the other states!)

Leo's new Educational Software Source Book is humming along, and will be ready for the new year.

You teachers are slow to show yourselves, but statistically, we know that there are at least 400 of you reading the magazine.

Mike Hayes in Sydney even had a Teachers' user group going this year, and there are teachers in far off Holbrook, Deniliquin and Western Australia using CoCo in their work.

To all of you we say that you have never had it so good! Right now Tandy will give priority assistance in any Tandy related computer project you may have in mind.

Neil Barton from Chatswood Hills State School recently found out what that means. He has just completed a Word Processing Project there using a bank of CoCos and year 7 students. Over a two week period, the students covered:

1. General introduction to hard and software.
2. Introduction to Word Processors.
3. Activities involving use of the editor.
4. Creating a piece of expression.
5. Hard copy of first draft.
6. Editing and proofreading of hard copy.
7. Screen editing of first draft.
8. Further development to the point where a book was produced which then became a classroom reader.

The children found the exercise stimulating, the final product was of excellent quality, and the exercise was voted a total success by all involved - including Tandy who ended up selling a couple more CoCos!

So if you're looking for a special project for December, grab a Tandy person or phone Carel Davies (02-675-1222) or Leo Wilson (07-52-9000) and ask their advice.

The promised report on "Maze" will have to wait another month. We just couldn't find enough time to do the project justice.

Next issue will review a Classroom Management program, and February's Australian CoCo will be a 'Back to School' edition with lots of programs that will help Teachers.

We still have room for more programs in that issue, so if you have something that you think is worth sharing, then by all means, send it in!

****

Some schools have received $300 to $400 software grants and have been asking as to what types of programs should be purchased for their CoCo. We looked at SOFTWARE SPECTRUM's range as well as TANDY's and came up with the following list.

Title	Source	Cass /Disk	Price
LEARN TO READ MUSIC	SS	C	$41.95
ALICE IN WONDERLAND	SS	C	$35.95
FRACTIONS TUT	SS	C	$28.95
CLASSROOM MANAGEMENT	SS		T.B.A
VOCAB BUILDER	SS	C	$86.85
SPEECH PAK	Tandy	C	$159.95
GRAPHICOM	SS	D	$39.95
SPELLING TEST (Tom Mix)	SS	C	$28.95
GEOGRAPHY PACK All bar Aust.	SS	C	$46.95
AUST QUIZ	Tandy	C	$9.95
VIP WRITER	SS	C/D	

EDUCATION OVERVIEW

The 'System' And Its Important Procedures

By Michael Plog, Ph.D.

In computer jargon, the word "system" probably has more definitions than any other single word. Different people use the same term, yet mean different things. I have heard computer salespeople talk about their system by describing a set of hardware components. Likewise, software vendors will describe a set of programs as a system. The description of an educational computer system needs to be more inclusive than either of these two examples.

To avoid confusion, let's understand the definitions. If you disagree with my use of the term, then simply go through the article and write your preferred term in place of "system." My dictionary defines system as "... an assemblage or combination of things or parts forming a complex or unitary whole." For educational computer use, a system should include three parts: hardware, software, and procedures. All three must fit together to make a single entity, which can properly be called a system.

First, an educational computer system has to have a hardware configuration. That could be as simple as a Color Computer with a single tape drive, or as complex as several computers sharing a hard disk drive. No matter how complex or simple, you cannot have a system without some sort of hardware. Hardware considerations are important when designing a system, and become essential when using the system. The hardware must fit the job to be completed.

Second, an educational system must have software. Without software, the machinery is nothing more than some very complex electronic components. For right now, we will ignore the difficulties caused by any differences between software, firmware, operating system, etc. We can safely lump all items together under some concept of "giving instructions to the machine" and call that software.

Third, a system must have a set of procedures to govern the operation. In my opinion, this is the most important part of any system.

Consider for a moment a computer system developed to keep track of student grades. If the clerical staff never updates the records, the entire venture is worthless. School officials will never be able to get accurate student grades from the computer. Procedures of such a system must include timely updates and things like backups and formats of paper records to accompany electronic records.

The concept of procedures of a computer system go much further than this simple example. A complete computer system can impact on many operations of a school (or business, home, hobby, etc.). For example, teachers will need to have a specific time limit for giving grade updates to the people who operate the computer. If the updates are not delivered on time, the system is still worthless — it will not produce the desired results for school officials.

Someone somewhere has possibly outlined all the necessary criteria for procedures of a computer system. I have not seen such an outline. No such outline will be presented here. Instead, we will concentrate on a few aspects uniquely connected with an educational system.

Training is essential for any successful implementation of a computer system. Training on the equipment is necessary, but not sufficient. Teachers need to know how to operate computers, but beyond that, teachers need to be taught the importance of computers in the classroom.

This is no easy task. We are talking about training in such areas as attitudes, behaviors, and even emotional approaches to dealing with situations. The task of teacher training is the school administrations' responsibility, as well as the responsibility of the unofficial educational leaders (usually from the ranks of teachers) found in any school setting. No matter how it is done, the final outcome must be that teachers understand how a computer can be important in their classrooms. This is a crucial part of the procedures of an educational computer system.

Another important aspect of system procedures is the incorporation of computers in classrooms. Assume that a significant number of teachers have a good understanding of the importance of computers. Assume further that machines and selected software are available to these classrooms. The task (part of the overall procedures of the system) remains as to how to incorporate computers in the educational experiences of students.

I recently heard about a school that implemented computers in classrooms last school year. Each teacher was assigned a computer for two weeks. At the end of that time, the computer moved on to another classroom and teacher. From the report I received, some teachers were elated at the prospect of having a computer, if just for a short time. Some teachers, however, found this to be disruptive to the learning process and generally disappointing. About the time that the students (and the teacher) discovered how to operate the machine, it was removed.

Now, it may be that the school the administration wanted all students to have some exposure to computers, with a more intensive implementation to occur that particular school year. Possibly, the administration received some computers and wanted to distribute time on the machines in a "fair" manner. It is also possible that the administration had no idea what to do with the computers, and were on a "fishing trip" — (Let some teachers figure out what can be done with the machines, then tell everyone else). I do not know what the administration had planned, or what the goals of this example were. Therefore, I cannot pass judgment on the experience. There are, however, a few rules that should be followed in incorporation procedures.

One major rule for incorporation procedures is to have as little disruption of the normal classroom activities as possible. Disruption only causes negative attitudes about educational uses of computers, which must be corrected before successful implementation can occur.

Another "rule of thumb" is that computer incorporation in a classroom can be achieved more smoothly if consideration is given to the material being covered. If teachers can be shown that the same material can be covered more easily using computers, they will be much more likely to assist the incorporation process.

Both of these rules recognize the importance of the teacher in the incorporation process. Without cooperation from teachers, administrators will never be able to fully implement an educational computer system. Likewise, without administrative support, computer use in schools will be only a fragmented approach — a true integrated system of several parts will be impossible.

If you have any reactions you wish to share, please write me at 829 Evergreen, Chatham, IL 62629.

KEVIN'S COLUMN

To my great delight I was set free from the office a couple of times this month to attend two computer displays.

The first of these was a traveling publicity presentation of a new machine the MAI 2000. This machine is really way out of our league but it did highlight one or two features af particular interest to us. Firstly a few details about the machine itself:

CPU	Motorola 68010
Clock Speed	8MHZ
RAM	768K standard
	1500K maximum
Drives	2 x 120 Megabyte
Base Price	$16000 approx

Along with this comes the hardware necessary to support up to 14 terminals and the capability ti connect up to 63 MAI 2000 systems together in a Local Area Network. All this processing power in a box that happily sits on top of a small table.

All this is very impressive but the kick I got from the display was the similarities I could see to my CoCo. I know we do not have the raw processing power that such money can buy but lets start with the operating system. The MAI 2000 runs a UNIX derivative BOSS/IX. I'm sure you will be aware that our own OS9 is essentially an 8-bit derivative of UNIX. Secondly, much emphasis was placed on the fact that the operating system and applications software was written in C. Tandy have on the shelves here now a C-compiler to run under OS9.

So what does that all mean for us? It emphasizes one of the superb features of the Color Computer in all its forms. I believe there is no better machine the average family can buy that will easily grow with them to whatever level they aspire. Essentially home computers are a learning tool, that was made very clear in the survey. Even if that learning is merely to discover what computers are all about. That learning process can be carried right through to the level of a full-blown CoCo system running OS9 (costing a mere 1500 to 2000 dollars). The environment you are then working in is akin to that supported by the commercial market. I am even led to believe that one of the main features of C is its transportability. It is quite a simple matter to write your program using C on the CoCo and cross compile it to another machine.

I also attended the Brisbane Computer Expo. What a dazzling array of machines. Graphics unlike any I have ever seen before, peripherals to make anyone drool and OH so many different computers. Each with their own features. Its enough to dazzle the mind of anyone even remotely interested in computers. If you ever get an opportunity to attend one of these Expo's in your home state it is well worth your effort to attend, even if only to find the cheapest supplier of floppy disks and paper.

I did make a surprise discovery amoungst this maze. In one corner I found the PREST COMPUTERS display. The suprise was they are selling the SMOKE SIGNAL CHIEFTAIN computer. This is a true multiuser system using a Motorola 6809 CPU the same as CoCo and running under Level 1 or 2 OS9. It supports up to two 20 Megabyte hard disks, one floppy disk or a 60 Megabyte tape streamer (for backup of the hard disk) and up to 16 terminals. So if you are in business have committed yourself to OS9 and the CoCo you have here in Australia a simple growth path retaining all your current software.

I am more convinced than ever that the future is bright for the CoCo and this is more than reinforced by Tandy who continue to support our machine.

Kevin.

We mean business...

CoCo is perfectly capable of taking care of business.

New software arriving from overseas each day underlines just how true this is.

Particularly if one employs OS-9 and a disk system, the ready made programs are starting to flood in. Stylograph, a Word Processor for OS-9 is here and from what we've seen of it so far, it promises to be better than Wordstar and every bit as good as the larger Word Processors.

I have never been a fan of the disk system as employed on CoCo, and feel that many small businesses would do well to consider several computers on tape system, dedicated to particular tasks, rather than persist with disk. None the less I am realistic enough to realise that the fast load properties of the disk are very nice to have and can be of definite benefit in schools and business.

There are bigger businesses that just have to have disk, and RAINBOW is one.

When Greg was with us, he developed the following program to manage the Newsagencies. It is disk based, and is a regression of a fuller program which runs our subscriber Data Base.

I have changed the program in a number of ways to suit our changed circumstances. The Data Base as presented, operates from Disk 3 on a four disk system.

This is totally unnecessary for anyone else's use and reflects the constraints made on our system by the Subscriber Data Base, rather than this one.

Points to note:

1. Line 3 is a quick SAVE feature for when you are making changes to a program, that you don't want to lose. You get to see the directory, and the free space left on the disk as well.
2. There is space on the disk for about 600 records.
3. Always FINISH everything you do, to ensure that your data gets onto the disk.

This little Data Base is probably not 'state of the art' any more, but it's adaptable, and perfectly able to handle small to medium quantities of Data.

The Listing:

```
1 'Data Base by Greg WILSON & Gr
aham MORPHETT ****** 25/9/84.
2 GOTO60
3 SAVE"DATABASE:3":DIR3:PRINTFRE
E(3):STOP
60 CLEAR1500:OPEN"D",#2,"DATA/DA
T:3",82
70 FIELD#2,5AS E$,27AS N$,26AS S
$,17AS T$,3AS Z$,4AS P$
80 CLS0:PRINT@32,"P";:PRINT@35,"
 PRINT A/CS FILE ";:PRINT@96,"E"
;:PRINT@99,"  EDIT A/CS FILE ";:
PRINT@160,"L";:PRINT@163,"  LABE
L PRINTER  ";:PRINT@224,"A";:PRI
NT@227," ADD TO ACS FILE ";:PRIN
T@288,"F";:PRINT@291,"   F I N I
 S H   ";
81 PRINT@426,"INKEY CODE";
90 I$=INKEY$:IFI$=""THENPRINT@42
6,"           ";:PRINT@426,"INKEY
 CODE";:GOTO90
100 CLS0:IFI$="P"THEN800 ELSE IF
 I$="E"THEN200 ELSE IF I$="A"THE
N 300 ELSE IF I$="F"THEN885 ELSE
 IF I$="L"THEN GOSUB890
110 GOTO 80
200 PRINT@76,"LINE #";:INPUT L:G
ET#2,L:PRINT@96,"F";:PRINT@98,"
 F I N I S H  ";
210 PRINT@128,"L";:PRINT@130,"
 L A B E L   ";:PRINT@160,"E";:P
RINT@162,E$;:PRINT@192,"N";:PRIN
T@194,N$;:PRINT@224,"S";:PRINT@2
26,S$;:PRINT@256,"T";:PRINT@258,
T$;:PRINT@288,"Z";:PRINT@290,Z$;
:PRINT@320,"P";:PRINT@322,P$;:PR
INT@426,"INKEY CODE";
220 I$=INKEY$:IFI$=""THEN220 ELS
E IFI$="F"THENPUT#2,L:GOTO80 ELS
E IFI$="L"THENGOSUB950 ELSE PRIN
T@426,"           ";
225 IFI$="L"THEN210
230 PRINT@351," ";:INPUTX$
240 IFI$="E"THEN RSETE$=X$
241 IFI$="N"THEN LSETN$=X$
242 IFI$="S"THEN LSETS$=X$
243 IFI$="T"THEN LSETT$=X$
244 IFI$="Z"THEN RSETZ$=X$
245 IFI$="P"THEN LSETP$=X$
250 CLS0:PRINT@32,"LINE # ";L;:G
OTO210
300 CLS0:L=LOF(2)+1:PRINT@32,"LI
NE #";L;
320 PRINT@77," ";:PRINT@64,"EXPI
RY";:INPUTA1$:IFLEN(A1$)>5 THEN3
20
325 IFLEFT$(A1$,3)="///"THENL=LO
F(2):GOTO80
330 PRINT@121," ";:PRINT@96," NA
ME ";:INPUTA2$:IFLEN(A2$)>17 THE
N330
340 PRINT@128,"STREET";:PRINT@18
9," ";:PRINT@159," ";:INPUTA3$:I
FLEN(A3$)>27THEN340
350 PRINT@217," ";:PRINT@192," T
OWN ";:INPUTA4$:IFLEN(A4$)>17THE
N350
360 PRINT@235," ";:PRINT@224,"ST
ATE ";:INPUTA5$:IFLEN(A5$)>3THEN
360
370 PRINT@268," ";:PRINT@256,"PC
ODE ";:INPUTA6$:IFLEN(A6$)>4THEN
370
380 RSETE$=A1$:LSETN$=A2$:LSETS$
=A3$:LSETT$=A4$:RSETZ$=A5$:LSETP
$=A6$:PUT#2,L
390 GOTO80
790 GOTO885
800 CLS0:PRINT@64,"N";:PRINT@67,
"    NUMERICAL ORDER ";:PRINT@12
8,"A";:PRINT@131," ALPHABETICAL
ORDER ";:PRINT@192,"P";:PRINT@19
5,"    POST CODE ORDER ";:PRINT@
426,"INPUT CODE";
810 I$=INKEY$:IFI$=""THEN810 ELS
E IFI$="A"THEN860 ELSE IFI$="P"T
HEN841 ELSE IFI$<>"N"THEN800
820 INPUT"FIRST #";FI:PRINT#-2,"
 AC/S ADDITIONS":FORI=FI TOLOF(2
):GET#2,I:PRINT#-2,I;TAB(5);E$;"
 ";N$;S$;T$;Z$;" ";P$
830 NEXTI:PRINT#-2:PRINT#-2:PRIN
T#-2:GOTO80
841 FOR A=2 TO 7:A$=MID$(STR$(A)
,2,1):FORI=1 TO LOF(2)
842 GET #2,I
843 IF LEFT$(P$,1)=A$ THENPRINT#
-2,I;TAB(5);E$;" ";N$;S$;T$;Z$;"
 ";P$;CHR$(10)
844 NEXT I
845 PRINT#-2:PRINT#-2:PRINT#-2
846 NEXT A
847 GOTO 80
860 FORI=1TO LOF(2):READL:GET#2,
L
870 PRINT#-2,E$;" ";N$;L;TAB(29)
;S$;T$;Z$;" ";P$
880 NEXTI:GOTO80
885 CLOSE#2:END
890 FORI=1TO LOF(2):GET#2,I
900 GOSUB950
910 NEXTI:RETURN
950 PRINT#-2,E$
```

```
960 PRINT#-2
970 PRINT#-2,N$
980 PRINT#-2,S$
990 PRINT#-2,T$;Z$;" ";P$
995 FORTT=1TO4:PRINT#-2:NEXTTT
1000 RETURN
```

Brian BERE-STREETER would like to hear of anyone interested in being part of a business person's user group. He will coordinate a meeting in Brisbane, and report in this column, the activities of the group.

The aim is to help anyone working with CoCo in business

The first meeting is set for the end of February. In the meantime, you can leave phone messages for him with us at RAINBOW, or write to him at:

9 Stanford St
ROBERTSON. QLD.

REVIEWS

Software Review

CoCo Keeps Roll And Grade Books With *Teacher's Pet*

While part of all professions, paper work and record keeping seem to be particulary evident in the teaching profession. Most secondary teachers instruct over 120 students per day and have to maintain a file of attendance, quizzes, tests, and class averages for each pupil daily. The repetitive filing system is well suited for computer operation. *Teacher's Pet* has taken the school teacher's roll book and converted it to use on the computer.

The author, P.T.Jones, includes a four-page reference manual and an eight-page tutorial. Both are well written and concise.

As with any program that develops a filing system, the majority of the user friendly program is devoted to the construction of the file. Once that is completed, continued usage throughout the school year would be quite simple.

The main menu includes the following options:

CREATE, ALPHA ORDER, EDIT/ENTER, NEW STUDENTS, SEARCH, DELETE, YEAR END REPORT, PRINT CLASS LIST, INPUT/OUTPUT, MARKS, LDIR (PRINT DIRECTORY).

Each formatted menu-screen has subsections that are self-explanatory.

Teacher's Pet allows you to enter up to 40 students per class for each of four school terms. Each student can have a maximum of nine tests per term. The *CREATE* screen has you define the parameters of your file and controls the class code, the term you are in and how many tests you want to enter. The screen is formatted with each student's name and the number of tests entered for that term. The program permits editing of names only, names and marks, or marks only with suboptions for term or test. The author developed an interesting editing system that includes hitting the BREAK key to get back to the main menu. It is not difficult to get used to and does the usual things you would expect an editing system to do, i.e., scanning or searching the file forwards or backwards. The program also will automatically alphabetize the class lists. Fail safe systems are incorporated to prevent loss of data.

The *YEAR END REPORT* allows any or all of the terms to be given a weighting factor so that if you want the second term to count twice as much as the other terms, enter the number two. The average of each student is calculated with the list of grades entered and can be sent to the screen or printer. The average is rounded off to the nearest whole percent. A zero entered as a test mark is not averaged with numerical grades. Letter grades will be printed as zeroes and are not averaged.

The program incorporates allowances for different Baud rates for printers and explains how to enter them. The author also includes hexadecimal numbers for the machine language program and several *POKEs* to change the screen color.

If you, as a teacher, have easy access to a computer and are not required to constantly refer to your roll book for student grades, then *Teacher's Pet* would suit you. I would have liked a couple of additional features, such as a flag for students whose average was below a defined standard and a specific progress card printout for those students. Overall, *Teacher's Pet* is a nicely developed filing program.

(Aurora Computing, 49 Brookland Ave., Aurora, Ontario, Canada L4G 2H6, 32K disk, $34.95)

— Michael F. Garozzo

File It With The Message Center

By Jim Schmidt

The Message Center is a program I wrote out of need rather than for fun. Initially, the need was a friend's, not mine. He (a CoCo owner) wanted to be sure that his messages to his teen-age children would be noticed and, hopefully, obeyed. Frequently he could not be home when his offspring arrived from school. He needed a way to leave them chores and also find out from them where they were going to be. Many times notes went unnoticed. Getting them to write him a note was akin to "cruel and unusual punishment" in their eyes.

The Message Center has changed all that for my friend. Because the kids think that using a computer to exchange messages is "tubular," they are a together family once again.

Sometime later, I was recounting the above to yet another friend, and a strange look came over him. He listened patiently, and as the last phoneme passed my lips he asked if the program could be used in a small office.

I thought a bit and replied that it could, but I had better add a few bells and/or whistles. It came to me that essentially the program was able to "broadcast" any visual text to those in sight of it. So, why not add *SAVE/LOAD* capability so that, in addition to messages, it could also handle display chores?

The Message Center was born. Friend number two uses the program every day now. His business is real estate which causes him and his two partners to come and go quite often during the day. Whoever answers the phone will typically do this:

1) Call up *The Message Center* program
2) Load the previous messages file
3) Enter the current message
4) Save the new file with the current message

An inquiry about messages is as follows:

1) Call up the program
2) Load the message file
3) Key @@ to begin the display
4) Watch the display using P to pause it where necessary
5) Use M to add more messages/replies if necessary

When he wishes to leave an urgent message, he simply keys in the message and leaves the program running in display mode. The "warble" gets the attention of the next person in the office.

He also takes his CoCo along when he has an open house. In this instance, he loads a previously saved sales pitch and runs it all day in display mode praising the property he is trying to sell.

Another use has been found for the message file by my real estate friend. He wrote a small print program to provide hardcopy of each day's message file. This printout is his follow-up tickler and allows him to inquire from the others if they have phoned so-and-so and what the results were, etc. The file produced by *The Message Center* is a plain vanilla ASCII text file with 32-byte records (strings). So there is no problem dumping it to a printer, if required. The other two partners like getting a hardcopy list of their message traffic periodically, also.

My kids have their own kids, so I can't use the program that way. I have been provided the services of a secretary whose services I share with the other programmers and analysts on the job, so no need there. But, I *do* have a use for the program! The nature of my job is such that I am perpetually studying something — new software, schedules, evaluations, all manner of text. You guessed it! I have found that by keying into *The Message Center* the key phrases and salient points of material I am studying and then just watching the display a few times, I am better able to absorb it.

Features And Functions

To get started, key in the program and *SAVE* or *CSAVE* it. *RUN* it, and the screen prompts you to press ENTER to begin. This prompt is only to let you know that this is an "empty" program and no text resides in memory. Pressing ENTER buys you a beep and a dark screen with the word READY in the lower right corner. You are into the key entry screen which will become apparent when you key the first letter of the first line of text. Key in up to 32 charac-

ters. If you key in the full 32 characters, the line will be stored automatically. If your line ends short of 32 characters then press ENTER to store the line. Continue keying and storing lines for the duration of your message. To store a blank line (skip a line), key in a space and press ENTER. Whenever you wish to display the keyed text, key @@ in the first two positions of the line entry area. The display will begin. To return to the entry screen from the display press M. After a short time, the beep will be heard and the dark screen with the READY prompt will reappear. You can now append more messages (hence "M").

On the entry screen, note the LEFT and LINE prompts. the former will keep track of the remaining characters in a line and the latter denotes the number of the line you are keying. A short beep will sound when you have only five characters remaining in a line. *The Message Center* can store a maximum of 100 lines of 32 characters of text. You can expand this if you like, but I don't recommend it (more on this later).

If you make an error keying a line, press the left arrow key and the line will erase allowing you to rekey it correctly. Once a line is stored, it is stored. The logic needed to allow change/delete, it turns out, is rarely required if you keep an eye on the keying. Since *The Message Center* is not a word processor, the overhead of this logic is usually extraneous. I do have a version of the program with change/delete logic installed, but it is slower and not necessary. If your particular application needs this logic, you can add it yourself or send me $5 and I'll send you that version along with a formatted print/dump program for the text file.

You touch typists out there, be careful. A lot is going on between characters in this program and the instruction *INKEY$* is used for key entry. So what, you say? Slow, is what! Not too slow for us two-finger types, but a tad slow for you five-finger folks. As you approach the 100 lines mark, the keying will get sluggish. Plus BASIC is doing its string thing. It is possible then to miss a letter. However, in practice, at an average of two lines per message, you would be approaching 50 messages. It would be better then to save the messages and clear the program to start a second message file. In that case, the first new message should be that there is a previous message file, and to save the current one before loading the older one.

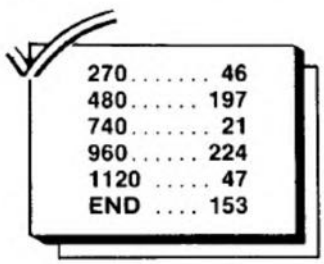
This could be automated also. Logic to prevent a second *LOAD* without an intervening *SAVE* would be easy to implement. This is implemented in the $5 version I mentioned earlier. Usually, it will not be needed. *Please* try to implement these and any other changes you may require yourself. After all, that's what computing is all about.

To clear memory of text, key in five asterisks in the first five positions of the entry line. You are now "empty."

The *SAVE/LOAD* functions are straightforward. While in the entry screen, key two pluses in the first two positions of the line entry area and you will then be prompted further. A *LOAD* will wipe out any text stored so far in the array, so be sure to *SAVE* first if necessary. After a *LOAD*, any text keyed in is added to the end of that which came in from the *LOAD*.

All that remains is to mention the PAUSE/RESUME functions. PAUSE will (surprise!) pause the display. If left paused for a couple of minutes, it will resume the display automatically. Keying 'R' while paused will resume the display without a wait.

I would like to hear from anyone who finds unusual or interesting uses to which they have put *The Message Center*. If you write and wish a response, please include a stamped envelope. My address is 196 Arlene Ct., Wheeling, IL 60090.



The listing:

```
0 '--------------------------
10 '--THE MESSAGE CENTER--
20 '--COPYRIGHT (C) 1983--
30 '--JIM SCHMIDT--
40 '--196A ARLENE CT.--
50 '--WHEELING, IL.  60090--
60 '--------------------------
70 '--EXTENDED COLOR BASIC--
80 '--FOR 16K ==> PCLEAR1--
90 '--------------------------
100 CLEAR 3800,&H3F00
110 DIM B$(100)
120 X=1:CT=0:LO=1
130 GOSUB 550
140 GOSUB780
150 SOUND150,5:GOSUB1240
160 IF X=100THEN380
170 A$=INKEY$:IFA$=""THEN170
180 IFA$<>CHR$(8)THENCT=CT+1
190 IFA$=CHR$(8)THENCT=0:CLS0:L$
=""":GOTO150
200 IFCT=27THENSOUND200,1
210 PRINT@54,"LEFT= ";32-CT;
220 PRINT@86,"LINE= ";X;
230 IFA$=CHR$(13)THENA$="":CLS0:
SOUND237,1:CT=32:GOTO280
240 L$=L$+A$:PRINT@0,L$
250 IF LEFT$(L$,2)="@@"THEN380
260 IFLEFT$(L$,2)="++"THEN1010
270 IFLEFT$(L$,5)="*****"THEN131
0
280 IFCT=32THENCT=0:B$(X)=L$:L$=
"":X=X+1:CLS0:SOUND237,1
290 IFX>99THEN GOSUB1280
300 PRINT@128,"   ENTER '@@' TO
START DISPLAY"
310 PRINT@192,"   ENTER '++' TO
SAVE THE TEXT"
320 PRINT@256,"  PRESS P TO PAUS
E THE DISPLAY"
330 PRINT@320," PRESS R TO RESUM
E THE DISPLAY"
340 PRINT@384," PRESS M TO ADD T
O THE DISPLAY"
350 PRINT@448," PRESS LEFT ARROW
 TO ERASE LINE"
360 GOTO160
370 '--DISPLAY ROUTINE--
```

```
380 CLS0:L$=""
390 FOR LO=1 TO 5:SOUND239,1:SOU
ND240,1:NEXT
400 FOR LO=1TOX-1
410 M$=INKEY$:IF M$="M"THENCLS0:
CT=0:GOTO150
420 IF M$="P"THENGOSUB 1200
430 P$=B$(LO)
440 F=32-LEN(P$):P$=P$+STRING$(F
," ")
450 GOSUB640
460 NEXT
470 P$=STRING$(32," "):GOSUB640
480 FORDE=1TO2000:NEXT
490 M$=INKEY$:IFM$="P"THENGOSUB1
200
500 IFM$="M"THENCLS0:CT=0:GOTO15
0
510 CLS0:P=0
520 GOTO390
530 '--M/L ROUTINE TO DEAL WITH
540 '  BASIC'S NASTY SCROLL--
550 CLS0
560 DEFUSR1=&H3F00
570 FOR P=&H3F00 TO &H3F00+52
580 READ ZZ : POKE P,ZZ
590 NEXT P
600 P=0
610 RETURN
620 '--CONVERT TO GREEN ON
630 '  BLACK AND POKE TEXT--
640 IF P$="" THEN RETURN ELSE FO
RZZ=1TOLEN(P$)
650 ZX=ASC(MID$(P$,ZZ,1))
660 IF ZX>63 AND ZX<128 THEN ZX=
ZX-64
670 IF P<0 OR P>511 THEN GOSUB 7
20
680 POKE P+&H400,ZX
690 P=P+1
700 NEXT ZZ
710 RETURN
720 IF P<0 THEN P=0
730 IF P>511 THEN P=480 : V=USR1
(Y)
740 RETURN
750 DATA 142,4,32,16,142,4,0,166
,0,167,32,48,1,49,33,191,63,253,
204,6,0,16,179,63,253,38,236,142
,5,224,134,32,167,0,48,1,191,63,
253,16,142
760 DATA 6,0,16,188,63,253,38,23
9,57,18,0,0,0,0
770 '--T I T L E--
780 P$="   HELLO...":GOSUB640
790 P=32
800 P$=STRING$(32," "):GOSUB640
810 P=64
820 P$="  THIS IS THE MESSAGE CE
NTER":GOSUB640
830 P=96
840 P$=STRING$(32," "):GOSUB640
850 P=128
860 P$="      COPYRIGHT (C) 1983"
:GOSUB640
870 P=160
880 P$="      JIM SCHMIDT":GOSUB6
40
890 P=192
900 P$="      196A ARLENE CT.":GO
SUB640
910 P=224
920 P$="      WHEELING, IL. 60090
":GOSUB640
930 FORDE=1TO1000:NEXT:CLS0
940 P=288
950 P$="    PRESS <ENTER> TO BEG
IN":GOSUB640
960 A$=INKEY$:IF A$<>CHR$(13)THE
N960
970 P=0:P$=""
980 CLS0
990 '--TAPE/DISK I/O ROUTINE--
1000 RETURN
1010 CLS:PRINT@137,"tAPE OR dISK
??"
1020 SOUND234,1
1030 D$=INKEY$:IFD$<>"T"AND D$<>
"D"THEN1030
1040 IFD$="T" THEN DV=-1 ELSEDV=
1
1050 PRINT@137,"sAVE OR 1OAD??"
1060 SOUND234,1
1070 D$=INKEY$:IFD$<>"S" AND D$<
>"L"THEN1070
1080 IFD$="L"THEN FT$="I" ELSE F
T$="O"
1090 PRINT:PRINT"           READY
 DRIVE"
1100 INPUT"FILENAME IS ";FI$
1110 IFFT$="I"THENX=1
1120 OPEN FT$,#DV,FI$
1130 IF FT$="O"THEN FORLP=1TO X-
1
1140 IF FT$="I"THENIF EOF(DV) TH
EN 1180
1150 IF FT$="O"THEN PRINT #DV,B$
(LP)ELSE LINE INPUT #DV,B$(X)
1160 IFFT$="I"THENX=X+1
1170 IFFT$="O"THEN NEXTLP ELSE G
OTO1140
1180 CLOSE#DV:CLS0:CT=0:L$="":GO
TO150
1190 '--PAUSE ROUTINE--
1200 FORDE=1TO10000
1210 N$=INKEY$:IFN$="R"THENRETUR
N
1220 NEXT:RETURN
1230 '--READY PROMPT--
1240 POKE1531,18:POKE1532,5
```

Continued on P 26..

CC Talk:

A Smart Terminal Package

By Frank Gossette

The capability of any microcomputer for communication with other computers, both large and small, can greatly enhance the power of the machine and its potential utility to the user. By connecting to a larger mainframe computer, the microcomputer user can access databases, electronic mail services, and other sophisticated resources normally beyond the capabilities of the home computer's hardware and software. *CC-Talk* is a terminal software package that can open the door to contemporary telecommunications for users of the Radio Shack TRS-80 Color Computer and the Tandy TDP-100 home computer systems.

The *CC-Talk* package contains all of the functions and features that are necessary to begin exploring telecomputing. Over normal phone lines with a modem, the user can access mainframe computers, bulletin board systems, and information services as a "dumb" terminal. The program can transmit and receive all ASCII characters and control codes in either full or half duplex operation. In addition, *CC-Talk* also provides the capability to download information from the host computer (which can be viewed off-line or saved to tape or disk), and to upload previously prepared ASCII files to the computer on the other end of the line.

The package is comprised of a BASIC language terminal driver program that can be easily modified to suit your own applications and system configuration, and a machine language interface routine that handles the serial input/output functions and hardware interfaces. In the spirit of making telecommunications free and accessible to all, the package is yours to use, enjoy and modify for your personal use.

Using The *CC-Talk* Package

The BASIC language terminal program is shown in Listing 1. It is written in Microsoft's Extended Color BASIC and will operate on any 16K or larger machine. Without modification, the program provides for dumb terminal operation in talk mode; storing of received text to a memory buffer for online or offline viewing in download mode; transmit any ASCII file (text, data, or BASIC program) saved on tape or disk to the host computer in upload mode; and display all previously downloaded text in the memory buffer in save mode. The operating mode is selected by a single-character keypress from the prompted menu of choices displayed at the bottom of the screen.

The first step in installing the terminal package is to type in the BASIC program in Listing 1 and save it to tape or disk as *"CCT.BAS"*. The program, as shown, operates on a 32K system. For users with 16K machines, change all references to hexadecimal address *"&H7xxx"* to *"&H3xxx"*. The same simple modification is required for the BASIC program which loads the machine language I/O routines (coded in *DATA* statements and *POKEd* into memory) shown in Listing 2. Simply change the *START* address from *"&H7D00"* to *"&H3D00"*. After typing in the loading program from Listing 2 and checking your typing carefully, save the program (on a different tape) as *"CCTLOAD"*. It will not be used regularly once it is properly installed.

Next, *RUN* the *"CCTLOAD"* program and, after proper execution, save the machine code to tape or disk using BASIC's *CSAVEM* or *SAVEM* functions under the name of *"CCT.IO"*. This file should immediately follow the BASIC terminal program if you are using a tape system.

To start terminal operations, first make as much memory as possible available to the package and protect the machine-code in high memory by entering:

```
PCLEAR 1:CLEAR 500,&H1FFF
```

Now, *RUN* the *"CCT.BAS"* program. The I/O drivers will be automatically loaded into memory if not already resident, and the menu of choices displayed.

If you are communicating with a remote computer over telephone lines, dial the computer's number and wait for the carrier signal. Enter talk mode by typing a 'T' from the menu. Then, type in whatever is required by the host computer (account number, password, etc.) in order to "sign-on." You are now functioning as a terminal as far as the other computer is concerned. You can exit talk mode by pressing the BREAK key at any point. This will not affect your connection to the host computer, but will return you to the BASIC program's menu. You can return to the talk mode or another mode from the menu as desired.

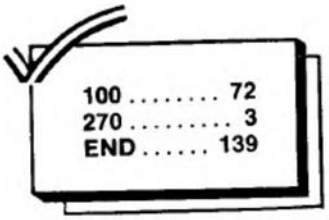
(Note: Text sent from the host while you are at the menu will, however, be lost.)

By typing a 'D' from the menu, you will enter the download mode. In this operating mode, all text sent by the host will be stored in a memory buffer in the Color Computer's memory. A reverse-video asterisk will appear on the screen for each line of text received. The actual text, however, will not appear. If you are downloading a program or data file, it is helpful to know approximately how many lines of text are in the file—as you know, the screen is 32 characters wide and you can count the "dots" to figure out when the downloading is complete. The keyboard functions normally during downloading, so you can send commands to the host to stop the listing just as in talk mode. A message will appear if the memory buffer becomes full, and in this case downloading will cease and you will be returned to the menu. The user can terminate the saving of received text by pressing the BREAK key, which also returns you to the menu.

The save mode (typing 'S' from the menu), as implemented in Listing 1, is actually an "examine buffer contents" routine. Downloaded text is saved in a condensed format, with only valid ASCII characters (both upper- and lowercase) and the CR (Carriage Return) character for each line packed into the buffer. In save mode, the user can step

through the stored text, from beginning to end, one page-full at a time. Pressing the space bar continues display of the next portion of the buffer, while hitting the ENTER key aborts the routine and returns to the menu. Since the entire terminal driver is written in BASIC, you can modify this section to actually save the buffer contents to magnetic tape or disk. A simple example of how this might be accomplished will be presented towards the end of this article.

Typing a 'U' from the menu puts you in the upload mode of the package. You will be prompted for the name of the ASCII file to be transferred. This file could be text, data or a BASIC program saved in ASCII format. The file is then read (from tape or disk) one line at a time and sent to the host as a character string. This routine can also be altered



Listing 1:

```
1 ' CC-TALK (C)1983 FRANK GOSSET
TE
2 ' C/Ø DEPARTMENT OF GEOGRAPHY
3 ' UNIVERSITY OF DELAWARE
4 ' NEWARK, DE 19711
5 '
6 ' enter> PCLEAR 1:CLEAR 5ØØ,&H
1FFF
7 ' before RUNning the program
8 '
1Ø START=&H7DØØ 'memory address
of machine code i/o routines
2Ø TBUFF=&H2ØØØ 'memory address
of download buffer start
3Ø IF PEEK(ST)=134 THEN 4Ø ELSE
CLOADM "CCT.IO":REM load i/o rou
tines if not resident
4Ø DEFUSR1=ST.' talk-mode entry
point
5Ø DEFUSR2=ST+&H158' download-mo
de entry point
6Ø DEFUSR3=ST+&H19B' upload-mode
 entry point
7Ø TECHO=ST+&HD7' full/half dupl
ex switch address
8Ø TFLG=1' Ø=full duplex(default
)/ 1=half duplex
9Ø POKE TE,TF' set duplex switch
1ØØ CLS:PRINT@2ØØ,"C C  - T A L
K":PRINT:PRINTTAB(11)"(C) 1983":
PRINTTAB(9)"DELAGRAPHICS":PRINT
11Ø GOTO 25Ø' jump to main menu
12Ø CLS:PRINT@482," talk mode :"
;TM$;:GOSUB 27Ø:TX=USR1(Ø):RETUR
N
13Ø CLS:PRINT@482,"downloading";
TM$;:GOSUB 27Ø:TX=USR2(TB):RETUR
N' pass download buffer address
14Ø CLS:PRINTTAB(8)"BUFFER CONTE
15Ø ' user may modify to save bu
ffer contents
16Ø ' to tape or disk file
17Ø BB=TB:BE=&H7CFF' buffer star
NTS":PRINTTAB(6)"spaceMORE/enter
EXIT":PRINT
t and end addresses
18Ø FOR I=BB TO BB+255
19Ø TC=PEEK(I):PRINT CHR$(TC);:N
EXT I
2ØØ S1=PEEK(&H88):S2=PEEK(&H89):
PRINT"cont";:POKE &H88,S1:POKE &
H89,S2
21Ø GOSUB 26Ø:IF TK$<>CHR$(32)TH
EN 23Ø
22Ø BB=BB+256:IF BB<BE THEN 18Ø
23Ø ' end view buffer routine -
modify for file save
24Ø RETURN
25Ø TP$=" tALK dOWNLD uPLOAD sAV
E qUIT ":TC$="TDUSQtdusq":TM$="
<break> TO EXIT":GOTO 28Ø
26Ø TK$=INKEY$:IF TK$="" THEN 26
Ø ELSE RETURN
27Ø POKE &H88,4:POKE &H89,Ø:RETU
RN'reset screen position
28Ø PRINT@48Ø, TP$;:GOSUB 26Ø:TC
=INSTR(TC$,TK$):IF TC=Ø THEN 28Ø
 ELSE ON TC GOSUB 12Ø,13Ø,32Ø,14
Ø,285,12Ø,13Ø,32Ø,14Ø,285:GOTO 2
8Ø
285 PRINT:PRINT"   type CONT to
restart":STOP:RETURN
29Ø 'user modifiable upload rout
ine here
3ØØ 'i/o routine requires BASIC
VARPTR
31Ø 'of string to be sent to hos
t computer
32Ø CLS:PRINT@481,"filename>";:I
NPUT TF$:TDEV=-1:OPEN "I",TDEV,T
F$
33Ø PRINT@481," uploading ";TF$;
:GOSUB 27Ø
34Ø IF EOF(TDEV) THEN 37Ø ELSE L
INE INPUT#TDEV,UP$
35Ø IF TFLG=1 THEN PRINT UP$ 'lo
cal echo for half duplex
36Ø TX=USR3(VARPTR(UP$)):GOSUB 3
8Ø:GOTO 34Ø
37Ø CLOSE TDEV:RETURN 'end uploa
d
38Ø FOR TC=Ø TO 255:NEXT TC:RETU
RN 'line turn-around delay
39Ø 'end of listing
```

by the user. The only real requirement is that the BASIC program pass Extended Color BASIC's *VARPTR* (variable pointer) of the string variable to be uploaded to the machine language interface routine.

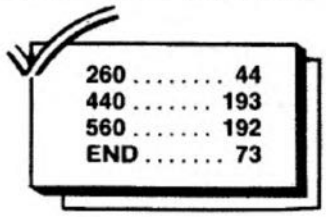
Uploading text files to the host computer, however, is more complicated than the other functions of the package. While downloading can be accomplished with relatively little knowledge of the computer's operating system on the other end of the connection (other than knowing how to print or list the file to the terminal), uploading requires some familiarity with the operating system of the host computer to get it to accept, save, and catalog your uploaded file. You may have to invoke an editor on the host to create a file to accept the incoming text and to save it properly. Such "housekeeping" chores should be done in talk mode, both before transfer and after the transfer is complete.

During upload mode, the keyboard is essentially de-activated. Hitting the BREAK key (perhaps several times) will abort the upload sequence, and might also crash the BASIC program. Simply *RUN* the program again to recover (as Color BASIC does not have an *ON ERROR* function). Text already transferred to the host computer will remain intact and can be saved or deleted from talk mode.

Entering a 'Q' (for Quit) from the menu returns you to Color BASIC's command level, with the OK prompt. While in Color BASIC, you can list the program, a disk directory, or even modify the program without affecting your modem connection (but all incoming text is lost, of course). You may return to terminal operation at any time by entering *CONT* (Color BASIC's "continue" function) or *RUN* (to restart the program).

While operating in any of the terminal modes, several of the keys are redefined to perform the special functions required of data terminals. Both upper- and lowercase letters can be sent to the host computer, with lowercase letters displayed in reverse video on the screen. The program defaults to an ALL CAPS mode which can be switched to upper/lower mode by SHIFT [0], just as in Color BASIC. The down arrow key is redefined as the *CNTRL* (control) key. Special control codes required by many mainframe computers, such as "*CNTRL C*," can be sent to the host by pressing the down arrow key and then the appropriate letter. (Note: This must be capital letter if in upper-/lowercase mode.) The SHIFT-left arrow (backspace) combination sends the ASCII *DEL* (delete) character, which erases the current line on many mainframes. A BREAK signal or *NULL* can be sent by pressing the down arrow then the SHIFT-up arrow combination. A summary of the redefined key functions is found in Figure 1.

The major advantage of writing the main terminal program in BASIC is



Listing 2:

```
10 'CC-TALK machine language i/o
20 'BASIC load routine
30 '
40 CLS:CLEAR 1500,&H1FFF
50 ST=&H7D00:CSUM=0
60 PRINT @40,"L O A D I N G"
70 FOR I=0 TO 468:READ X
80 CSUM=CSUM + X
90 POKE ST+I,X
100 NEXT I
110 IF CSUM <>48157 THEN PRINT @
224,"checksum load error--check
data":PRINT:GOTO 130
120 PRINT @228,"I/O DRIVERS INST
ALLED":PRINT:PRINT "enter:":PRIN
T:PRINT"CSAVEM 'CCT.IO',&H7D00,&
H7ED4,&H7D00":PRINT
130 END
200 DATA  134, 0, 167, 141, 0, 3
0, 141, 60, 38, 4
210 DATA  141, 30, 32, 248, 109,
 141, 0, 18, 38, 12
220 DATA  189, 163, 10, 129, 13,
 38, 235, 23, 0, 252
230 DATA  32, 230, 141, 40, 32,
226, 0, 32, 0, 4
240 DATA  32, 42, 189, 161, 193,
 39, 20, 129, 3, 38
250 DATA  13, 166, 141, 255, 237
, 129, 2, 38, 2, 53
260 DATA  2, 53, 16, 57, 23, 0,
149, 57, 23, 0
270 DATA  100, 129, 0, 57, 52, 8
4, 230, 140, 218, 238
280 DATA  140, 213, 174, 140, 20
8, 129, 32, 36, 9, 129
290 DATA  13, 38, 17, 231, 192,
239, 140, 197, 167, 128
300 DATA  175, 140, 190, 140
310 REM   CHANGE THIS NEXT VALUE
 TO 60 FOR 16K
320 DATA  124
330 DATA  255, 16, 39, 1, 19
340 DATA  53, 212, 142, 1, 104,
48, 31, 38, 252, 57
350 DATA  52, 23, 26, 80, 127, 2
55, 32, 141, 239, 52
360 DATA  2, 198, 8, 100, 228, 7
3, 73, 183, 255, 32
370 DATA  141, 226, 90, 38, 244,
 134, 2, 183, 255, 32
380 DATA  141, 216, 141, 214, 50
, 97, 53, 151, 142, 0
390 DATA  192, 32, 3, 142, 1, 11
3, 48, 31, 38, 252
400 DATA  57, 52, 21, 26, 80, 18
2, 255, 34, 71, 37
410 DATA  31, 141, 231, 182, 255
, 34, 71, 37, 242, 79
420 DATA  52, 2, 198, 7, 141, 22
3, 182, 255, 34, 71
```

to provide the user with a telecommunications environment that can be modified to suit his or her particular needs. As you use and modify the program to your own applications you will, no doubt, discover trade-offs and limitations that result from this decision. However, compared to many communications packages available for home computers, you may find the flexibility inherent in my approach more than makes up for its shortcomings. One can, in fact, through a careful examination of the program in Listing 1, incorporate specific terminal functions into your other programs. The possibilities for innovation are limited only by your imagination.

Technical Considerations

The actual asynchronous communications parameters used by *CC-Talk* are:

Transmission Rate	**300 Baud**
Word Size	8 bits (seven data, space parity)
Stop Bits	Two
Start Bits	One
Parity	No checking (space parity sent)

Figure 1: **SPECIAL FUNCTION KEYS**

KEY PRESSED	FUNCTION	CHARACTER SENT
<BREAK>	BREAK	NONE
<CLEAR>	CLEARS SCREEN	NONE
Left ARROW	BACKSPACE	BACKSPACE
<SHIFT> L. ARROW	ERASE LINE	DELETE
<SHIFT>ZERO	UPPER/lower case	NONE
Down ARROW	<CNTRL>	CONTROL CHARACTER See Note Below
<CNTRL> + <SHIFT> UP ARROW	LONG BREAK	NULL NONE

NOTE: The DOWN ARROW <CNTRL> Key Causes Terminal to Pause and wait for next key pressed. The "control code" for that character is then sent to the host.

Most Host Computers support the use of special "control characters" such as "CNTRL-C." When the keyboard is in lower case mode, the proper control code can be sent by <CNTRL> then <SHIFT> character.

These specifications should be compatible with nearly all timesharing systems you may wish to contact, including business or university mainframes, and microcomputer-based bulletin boards.

Terminal operations can be performed in either full or half duplex modes. In full duplex mode, all characters sent to the host computer are "echoed" by the host and then displayed on the Color Computer's screen. Half duplex or "local echo" mode, used by some systems, does not send back the character received. In this operating mode, characters are displayed on the screen before being sent out. The "duplex switch" is

```
430 DATA  102, 228, 90, 38, 245,
 141, 212, 53, 2, 68
440 DATA  32, 1, 79, 53, 149, 1,
 52, 2, 129, 32
450 DATA  36, 46, 129, 8, 39, 42
, 129, 12, 38, 5
460 DATA  189, 169, 40, 32, 44,
129, 21, 38, 4, 134
470 DATA  127, 32, 33, 129, 10,
38, 15, 189, 161, 193
480 DATA  39, 251, 128, 64, 129,
 31, 38, 18, 141, 63
490 DATA  32, 17, 129, 13, 38, 1
3, 141, 14, 109, 140
500 DATA  200, 39, 3, 189, 163,
10, 23, 255, 97, 53
510 DATA  2, 57, 52, 22, 158, 13
6, 140, 5, 160, 35
520 DATA  30, 48, 136, 192, 52,
16, 142, 4, 0, 236
530 DATA  136, 64, 237, 129, 172
, 228, 35, 247, 204, 96
540 DATA  96, 237, 129, 156, 136
, 35, 247, 53, 16, 159
550 DATA  136, 53, 150, 52, 18,
142, 63, 0, 134, 0
560 DATA  183, 255, 32, 48, 31,
38, 252, 134, 2, 183
570 DATA  255, 32, 53, 146, 189,
 179, 237, 31, 1, 175
580 DATA  141, 254, 196, 142, 4,
 32, 175, 141, 254, 191
590 DATA  134, 1, 167, 141, 254,
 182, 22, 254, 149, 42
600 DATA  42, 66, 85, 70, 70, 69
, 82, 32, 70, 85
610 DATA  76, 76, 42, 42, 13, 48
, 140, 237, 166, 128
620 DATA  189, 163, 10, 129, 13,
 38, 247, 53, 116, 57
630 DATA  23, 255, 24, 129, 0, 3
9, 3, 189, 163, 10
640 DATA  57, 189, 179, 237, 31,
 1, 166, 132, 52, 2
650 DATA  238, 2, 134, 2, 167, 1
41, 254, 120, 141, 226
660 DATA  38, 252, 23, 254, 119,
 141, 219, 166, 192, 23
670 DATA  254, 190, 141, 14, 106
, 228, 38, 236, 134, 13
680 DATA  23, 254, 179, 141, 201
, 53, 2, 57, 109, 141
690 DATA  255, 9, 38, 4, 141, 19
0, 39, 252, 57
```

controlled by the variable *TFLAG* in Line 80 of program Listing 1. If the host computer you are connected to does not echo the characters as received (nothing appears on the screen as you type), change the value of *TFLAG* to one for half-duplex operation.

The Machine Language I/O Routines

Technically, the Tandy Color Computers include, as standard equipment, an RS-232 serial communications interface. In reality, the "serial port" must be directly controlled by the MC6809E central processor (CPU) in software. The only use for the serial interface supported in Microsoft BASIC (in Read Only Memory) is a serial printer, such as Radio Shack's DMP-100. Input from or output to any other peripheral device, such as the modem, must be programmed by the user. Listing 3 provides the assembly language source code for the input/output interfaces and other high-speed terminal functions needed for telecommunications operations. (These operations can only be done in machine language, as BASIC is much too slow.)

The machine code is quite compact, occupying only 468 bytes of memory, and is completely relocatable (meaning it can be placed anywhere in Random Access Memory and it will function properly). The I/O routines employ some of the built-in functions from Color BASIC's Read Only Memory for polling the keyboard, clearing the screen, and displaying a character on the screen thereby reducing the memory requirements of the routines. The assembly language program in Listing 3 was produced on Radio Shack's *EDTASM+* editor assembler.

The assembly language routines which perform the actual serial input and output functions are subroutines *OUTCHR* (Listing 3, Lines 810 through 1030) and *INPUT* (Lines 1070 through 1370). *OUTCHR* sends the character in the 6809 CPU accumulator "A" to the serial output port (bit one of address $FF20) as an eight-bit word with one start bit, seven data bits, space parity, and two stop bits. Subroutine *INPUT* checks the RS-232 input line (bit zero of $FF22) for a start bit and, if found, returns a seven-bit ASCII character in accumulator "A" without checking parity.

While in one of the terminal modes, the I/O interface re-configures the functions of several of the keys, as described above, in subroutine *KEYCOD* (Lines 1420 through 1720). Whenever the down arrow key is pressed (redefined as the *CNTRL* key), the terminal waits on the next key pressed as the control character to be sent. The routine also checks the contents of *ECHO* (Line 1410) to determine if you are operating in full or half-duplex mode — set by the variable *TFLAG* in the BASIC program. If you are operating in half duplex mode, input from the keyboard will be displayed on the screen before transmission.

Since the Color Computer's screen displays only 16 lines of 32 characters each, lines to text sent by the host computer (typically 80 columns in length) can quickly fill the screen. Most mainframe computers provide a "line-turnaround" delay at the end of each line to allow a hard-copy terminal time to reposition the print head back at the left of the page. The I/O interface takes advantage of this delay by scrolling the screen during this pause, if the screen is nearly full. This procedure (Lines 1760-1950 of Listing 3) opens at least two lines (64 characters) at the bottom of the screen for the next incoming line, and eliminates most instances of data loss due to having the screen scroll in the middle of a line being received.

The main terminal loop (Lines 270-390) controls the operations of talk mode and downloading. The routine continuously scans the input port for a character received, and either displays the text on the screen or stores the incoming text in the memory buffer depending on the value of *MODE* (Line 410). When in download mode, subroutine *ALTMOD* does the input buffering and displays the line-received character (asterisk) for each line stored. When the buffer is full, a message is displayed and program control is returned to the BASIC menu.

Downloading is initialized in the routine *DINIT* (Lines 2090 to 1120). This procedure takes the argument *TBUFF* (Line 60 in Listing 1) from the *USR2* call in the BASIC program as the starting point in memory for the downloaded text. Incoming data will be stored sequentially from this address up to *MAXBUF* (Listing 3, Line 200) — $7CFF for a 32K system.

The uploading routine, Lines 2390 through 2590 of Listing 3, begins by retrieving the *VARPTR* argument of the string variable to be sent which was passed by the BASIC program. Extended Color BASIC's variable pointer for string variables defines a memory address which contains the length of the string, and the 16-bit address of the starting character of the string. These values are used by the routine to determine how many characters are to be sent and where to find them in memory. After sending the string, one character at a time, the routine appends a CR (Carriage Return) to the output stream and returns to the BASIC calling program. If you are operating in full duplex, the routine waits for the host's echo before sending the next character.

Enhancements And Other Features

Whether you are a beginning BASIC programmer or an accomplished hacker, there are numerous modifications and enhancements that can be made to the package. One of the simplest, and most useful, would be to add the capability of actually saving downloaded text to tape or disk on your own system. Program Listing 4 provides an example of how this might be accomplished for those of you with tape systems. Lines 150 through 240 of the BASIC terminal driver program in Listing 1 are replaced with a new subroutine. First, you are prompted for the name of the new file. If you respond with a valid character string, a new file is opened to receive the text (an ENTER allows you to look, but not save). The program then *PEEKs* into the memory buffer and creates a string variable until the end-of-line carriage return character is encountered. Then, the line of text is written to the tape. This process continues for each line of text displayed on the screen as you press the space bar. Pressing the ENTER key will terminate saving, close the tape file, and return you to the main menu.

Any type of text, including a BASIC program listing, can be saved in this fashion. Since the file is an ASCII character file, a program can be loaded just as if it had been saved using Color BASIC's (*CSAVE"filename",A*) option. A useful further enhancement of this routine would be the option of choosing to save, delete, or edit a downloaded line as it is displayed. I'll leave that idea for you to experiment with on your own.

Whenever you are connected to a mainframe computer — whether it is the computer at work, school, or one of the subscription services catering to microcomputer users — you need to spend some time to familiarize yourself with the peculiarities and functions of the operating system. In some cases, you can take advantage of the more powerful system software on the host to make your terminal session more enjoyable. For example, many operating systems allow you to specify certain attributes of your terminal — screen size, line width, end-of-line character (Car-

Listing 3:

```
                        00010 *
                        00020 *       (C)1983
                        00030 *    FRANK GOSSETTE
                        00040 * DEPARTMENT OF GEOGRAPHY
                        00050 * UNIVERSITY OF DELAWARE
                        00060 *
7D00                    00100         ORG     $7D00
                        00110 *
                        00120 *  CC-TALK TERMINAL
                        00130 *    I/O ROUTINES
                        00140 *
          A1C1          00150 POLKEY  EQU     $A1C1
          A30A          00160 SCNOUT  EQU     $A30A
          A928          00170 CLRSCN  EQU     $A928
          B3ED          00180 INTCNV  EQU     $B3ED
          0088          00190 CURPOS  EQU     $88
          3CFF          00200 MAXBUF  EQU     $3CFF
                        00210 *
                        00220 *
                        00230 * MAIN TERMINAL LOOP
                        00240 *
7D00 86   00            00250 START   LDA     #$00
7D02 A7   8D 001E       00260         STA     MODE,PCR
7D06 BD   3C            00270 TERM    BSR     INCHEK
7D08 26   04            00280         BNE     TMODE
7D0A 8D   1E            00290         BSR     KEYCHK
7D0C 20   F8            00300         BRA     TERM
7D0E 6D   8D 0012       00310 TMODE   TST     MODE,PCR
7D12 26   0C            00320         BNE     OTHER
7D14 BD   A30A          00330         JSR     SCNOUT
7D17 81   0D            00340         CMPA    #$0D
7D19 26   EB            00350         BNE     TERM
7D1B 17   00FC          00360         LBSR    SCROLL
7D1E 20   E6            00370         BRA     TERM
7D20 8D   28            00380 OTHER   BSR     ALTMOD
7D22 20   E2            00390         BRA     TERM
                        00400 *
7D24                    00410 MODE    RMB     1
7D25                    00420 BUFPOS  RMB     2
7D27                    00430 SCNPOS  RMB     2

7D29      2A            00440 LINCHR  FCB     $2A
                        00450 *
7D2A BD   A1C1          00460 KEYCHK  JSR     POLKEY
7D2D 27   14            00470         BEQ     XKEY
7D2F 81   03            00480         CMPA    #$03
7D31 26   0D            00490         BNE     DECODE
7D33 A6   8D FFED       00500         LDA     MODE,PCR
7D37 81   02            00510         CMPA    #$02
7D39 26   02            00520         BNE     RET1
7D3B 35   02            00530         PULS    A
7D3D 35   10            00540 RET1    PULS    X
7D3F 39                 00550         RTS
7D40 17   0095          00560 DECODE  LBSR    KEYCOD
7D43 39                 00570 XKEY    RTS
                        00580 *
7D44 17   0064          00590 INCHEK  LBSR    INPUT
7D47 81   00            00600         CMPA    #$00
7D49 39                 00610         RTS
                        00620 *
7D4A 34   54            00630 ALTMOD  PSHS    U,X,B
7D4C E6   8C DA         00640         LDB     LINCHR,PCR
7D4F EE   8C D5         00650         LDU     SCNPOS,PCR
7D52 AE   8C D8         00660         LDX     BUFPOS,PCR
7D55 81   20            00670         CMPA    #$20
7D57 24   09            00680         BHS     SAVE
7D59 81   0D            00690         CMPA    #$0D
7D5B 26   11            00700         BNE     XALT
7D5D F7   C0            00710         STB     ,U+
7D5F EF   8C C5         00720         STU     SCNPOS,PCR
7D62 A7   80            00730 SAVE    STA     ,X+
7D64 AF   8C BE         00740         STX     BUFPOS,PCR
7D67 8C   3CFF          00750         CMPX    #MAXBUF
7D6A 1027 0113          00760         LBEQ    MSG
7D6E 35   D4            00770 XALT    PULS    PC,U,X,B
                        00780 *
                        00790 *  RS232 OUTPUT
                        00800 *
7D70 8E   0168          00810 WTOUT   LDX     #$0168
7D73 30   1F            00820 OUT1    LEAX    -1,X
7D75 26   FC            00830         BNE     OUT1
7D77 39                 00840         RTS
7D78 34   17            00850 OUTCHR  PSHS    X,B,A,CC
7D7A 1A   50            00860         ORCC    #$50
7D7C 7F   FF20          00870         CLR     $FF20
7D7F 8D   EF            00880         BSR     WTOUT
7D81 34   02            00890         PSHS    A
7D83 C6   08            00900         LDB     #$08
7D85 64   E4            00910 OUT2    LSR     ,S
7D87 49                 00920         ROLA
7D88 49                 00930         ROLA
7D89 B7   FF20          00940         STA     $FF20
7D8C 8D   E2            00950         BSR     WTOUT
7D8E 5A                 00960         DECB
7D8F 26   F4            00970         BNE     OUT2
7D91 86   02            00980         LDA     #$02
7D93 B7   FF20          00990         STA     $FF20

7D96 8D   D0            01000         BSR     WTOUT
7D98 8D   D6            01010         BSR     WTOUT
7D9A 32   61            01020         LEAS    1,S
7D9C 35   97            01030 XOUT    PULS    PC,X,B,A,CC
                        01040 *
                        01050 * RS232 INPUT
                        01060 *
7D9E 8E   00C0          01070 HALF    LDX     #$00C0
7DA1 20   03            01080         BRA     INWAIT
7DA3 8E   0171          01090 FULL    LDX     #$0171
7DA6 30   1F            01100 INWAIT  LEAX    -1,X
7DA8 26   FC            01110         BNE     INWAIT
7DAA 39                 01120         RTS
                        01130 *
7DAB 34   15            01140 INPUT   PSHS    X,B,CC
7DAD 1A   50            01150         ORCC    #$50
7DAF B6   FF22          01160 IN1     LDA     $FF22
7DB2 47                 01170         ASRA
7DB3 25   1F            01180         BLO     XINP
7DB5 8D   E7            01190         BSR     HALF
7DB7 B6   FF22          01200         LDA     $FF22
7DBA 47                 01210         ASRA
```

```
7DBB 25 F2      01220          BLO    IN1
7DBD 4F         01230          CLRA
7DBE 34 02      01240          PSHS   A
7DC0 C6 07      01250          LDB    #$07
7DC2 8D DF      01260 IN2      BSR    FULL
7DC4 B6 FF22    01270          LDA    $FF22
7DC7 47         01280          ASRA
7DC8 66 E4      01290          ROR    ,S
7DCA 5A         01300          DECB
7DCB 26 F5      01310          BNE    IN2
7DCD 8D D4      01320          BSR    FULL
7DCF 35 02      01330          PULS   A
7DD1 44         01340          LSRA
7DD2 20 01      01350          BRA    INEND
7DD4 4F         01360 XINP     CLRA
7DD5 35 95      01370 INEND    PULS   PC,X,B,CC
                01380 *
                01390 * KEYBOARD DECODER
                01400 *
7DD7    01      01410 ECHO     FCB    $01
7DD8 34 02      01420 KEYCOD   PSHS   A
7DDA 81 20      01430          CMPA   #$20
7DDC 24 2E      01440          BHS    TECHO
7DDE 81 08      01450          CMPA   #$08
7DE0 27 2A      01460          BEQ    TECHO
7DE2 81 0C      01470          CMPA   #$0C
7DE4 26 05      01480          BNE    DEL
7DE6 BD A92B    01490          JSR    >CLRSCN
7DE9 20 2C      01500          BRA    XKEYCD
7DEB 81 15      01510 DEL      CMPA   #$15
7DED 26 04      01520          BNE    CNTRL
7DEF 86 7F      01530          LDA    #$7F
7DF1 20 21      01540          BRA    SNDKEY
7DF3 81 0A      01550 CNTRL    CMPA   #$0A
7DF5 26 0F      01560          BNE    ENTER
7DF7 BD A1C1    01570 K2       JSR    POLKEY
7DFA 27 FB      01580          BEQ    K2
7DFC 80 40      01590          SUBA   #$40
7DFE 81 1F      01600          CMPA   #$1F
7E00 26 12      01610          BNE    SNDKEY
7E02 8D 3F      01620          BSR    NULL
7E04 20 11      01630          BRA    XKEYCD
7E06 81 0D      01640 ENTER    CMPA   #$0D
7E08 26 0D      01650          BNE    XKEYCD
7E0A 8D 0E      01660          BSR    SCROLL
7E0C 6D 8C C8   01670 TECHO    TST    ECHO,PCR
7E0F 27 03      01680          BEQ    SNDKEY
7E11 BD A30A    01690          JSR    SCNOUT
7E14 17 FF61    01700 SNDKEY   LBSR   OUTCHR
7E17 35 02      01710 XKEYCD   PULS   A
7E19 39         01720          RTS
                01730 *
                01740 *  SCREEN SCROLL
                01750 *
7E1A 34 16      01760 SCROLL   PSHS   X,B,A
7E1C 9E 88      01770          LDX    <CURPOS
7E1E 8C 05A0    01780          CMPX   #$05A0
7E21 23 1E      01790          BLS    XSCRL
7E23 30 88 C0   01800          LEAX   -$40,X
7E26 34 10      01810          PSHS   X
7E28 8E 0400    01820          LDX    #$0400
7E2B EC 88 40   01830 MOVE     LDD    $40,X
7E2E ED 81      01840          STD    ,X++
7E30 AC E4      01850          CMPX   ,S
7E32 23 F7      01860          BLS    MOVE
7E34 CC 6060    01870 BLANK    LDD    #$6060
7E37 ED 81      01880          STD    ,X++
7E39 9C 88      01890          CMPX   <CURPOS
7E3B 23 F7      01900          BLS    BLANK
7E3D 35 10      01910          PULS   X
7E3F 9F 88      01920          STX    <CURPOS
7E41 35 96      01930 XSCRL    PULS   PC,X,B,A
                01940 *
                01950 * LONG NULL BREAK
                01960 *
7E43 34 12      01970 NULL     PSHS   X,A
7E45 8E 3F00    01980          LDX    #$3F00
7E48 86 00      01990          LDA    #$00
7E4A B7 FF20    02000          STA    $FF20
7E4D 30 1F      02010 NWAIT    LEAX   -1,X
7E4F 26 FC      02020          BNE    NWAIT
7E51 86 02      02030          LDA    #$02
7E53 B7 FF20    02040          STA    $FF20
7E56 35 92      02050 XNULL    PULS   PC,X,A
                02060 *
                02070 *  INIT DOWNLOAD
                02080 *
7E58 BD B3ED    02090 DINIT    JSR    INTCNV
7E5B 1F 01      02100          TFR    D,X
7E5D AF 8D FEC4 02110          STX    BUFPOS,PCR
7E61 8E 0420    02120          LDX    #$0420
7E64 AF 8D FEBF 02130          STX    SCNPOS.PCR
7E68 86 01      02140          LDA    #$01
7E6A A7 8D FEB6 02150          STA    MODE,PCR
7E6E 16 FE95    02160          LBRA   TERM
                02170 *
                02180 * BUFFER FULL MSG
                02190 *
7E71    2A      02200 MCHAR    FCC    /**BUFFER FULL
7E80    0D      02210 CR       FCB    $0D
                02220 *
7E81 30 8C ED   02230 MSG      LEAX   MCHAR,PCR
7E84 A6 80      02240 SHOW     LDA    ,X+
7E86 BD A30A    02250          JSR    SCNOUT
7E89 81 0D      02260          CMPA   #$0D
7E8B 26 F7      02270          BNE    SHOW
7E8D 35 74      02280 ABORT    PULS   U,Y,X,B
7E8F 39         02290          RTS
                02300 *
                02310 *  UPLOAD ROUTINE
                02320 *
7E90 17 FF18    02330 TSTIN    LBSR   INPUT
7E93 81 00      02340          CMPA   #$00
7E95 27 03      02350          BEQ    XTSTI
7E97 BD A30A    02360          JSR    SCNOUT
7E9A 39         02370 XTSTI    RTS
                02380 *
7E9B BD B3ED    02390 UPLOAD   JSR    INTCNV
7E9E 1F 01      02400          TFR    D,X
7EA0 A6 84      02410          LDA    ,X
```

```
7EA2 34   02       02420          PSHS    A
7EA4 EE   02       02430          LDU     2,X
7EA6 86   02       02440          LDA     #$02
7EA8 A7   8D FE78  02450          STA     MODE,PCR
7EAC 8D   E2       02460 CHEKIN   BSR     TSTIN
7EAE 26   FC       02470          BNE     CHEKIN
7EB0 17   FE77     02480          LBSR    KEYCHK
7EB3 8D   DB       02490          BSR     TSTIN
7EB5 A6   C0       02500          LDA     ,U+
7EB7 17   FEBE     02510          LBSR    OUTCHR
7EBA 8D   0E       02520          BSR     TSTECH
7EBC 6A   E4       02530          DEC     ,S
7EBE 26   EC       02540          BNE     CHEKIN
7EC0 86   0D       02550 UPDONE   LDA     #$0D
7EC2 17   FEB3     02560          LBSR    OUTCHR
7EC5 8D   C9       02570          BSR     TSTIN
7EC7 35   02       02580          PULS    A
7EC9 39            02590          RTS
                   02600 *
                   02610 * TEST FOR FULL DUPLEX
7ECA 6D   8D FF09  02620 TSTECH   TST     ECHO,PCR
7ECE 26   04       02630          BNE     XTEC
7ED0 8D   BE       02640 WECHO    BSR     TSTIN
7ED2 27   FC       02650          BEQ     WECHO
7ED4 39            02660 XTEC     RTS
                   02670 *
                   02680 *
                   02690 *
          0000     02700          END
00000 TOTAL ERRORS
```

Listing 4:

```
140 CLS:PRINTTAB(8)"BUFFER CONTE
NTS":PRINTTAB(6)"spaceMORE/enter
EXIT"
150 INPUT "SAVE FILE NAME";TF$:I
F TF$<>"" THEN TSAVE=1:OPEN "O",
#-1,TF$
160 ' <enter> ( null filename) t
o preview buffer contents only
170 BB=TB:BE=&H7CFF' buffer star
t and end addresses
180 J=0:SV$="":FOR I=BB TO BB+25
0
190 TC=PEEK(I):J=J+1:IF TC=13 TH
EN 195 ELSE SV$=SV$+CHR$(TC):NEX
T I
195 PRINT "no <cr> found" '250 c
haracters without end-of-line
200 PRINT SV$ 'display line on s
creen
210 GOSUB 260:IF TK$<>CHR$(32) T
HEN 230
215 IF TSAVE THN PRINT #-1,SV$ '
PRINT TO TAPE
220 BB=BB+J:IF BB<BE THEN 180
230 CLOSE #-1
240 RETURN 'end of save routine
```

riage Return or Line Feed + Carriage Return) etc. If you can specify line width, set the host to send 32-character lines. This will eliminate the "word-wrap" of the Color Computer's screen which some people find annoying.

CC-Talk uses only the carriage return to terminate a line, and ignores a line feed if sent by the host. If the other computer requires a linefeed as part of the end-of-line signal, see if you can change the "terminal environment" settings on the host, or type *CNTRL J* from the keyboard. When uploading a file to a system that requires a linefeed, you can add a line to the *CCT.BAS* program:

```
255 UP$=UP$ + CHR$(10)
```

This should cure any problems associated with that feature of the program.

When operating in full duplex mode, on some systems, it may be necessary to clear the screen prior to uploading a text string. If you encounter problems, simply modify Line 350 in the BASIC program to read:

```
350 IF TFLG=1 THEN PRINT UP$
ELSE CLS 'print or clear
```

The possibilities for further enhancements to the package are endless — put your ingenuity to good use. I use a highly modified version for turning the Color Computer into a color graphics terminal running mapping and graphics software on the university's mainframes. Perhaps, in a future article, I'll show you how much fun that can be. In the meantime, enjoy this no-cost introduction to telecommunications and share your discoveries and enhancements with others!

... From P 18

```
1250 POKE1533,1:POKE1534,4
1260 POKE1535,25:RETURN
1270 '--100 LINE LIMIT REACHED-
1280 P$="WARNING - MAX LINES IS
100":SOUND 245,10:P=480
1290 GOSUB640:RETURN
1300 '--PURGE ARRAY--
1310 CT=0:L$="":A$="":FORLL=1TOX
-1:B$(LL)="":NEXT:CLS0:X=1:GOTO1
50
```

Adding An Auto Answer

Tony Sharp

After reading Dr. Lane Lester's article in the November '83 issue of RAINBOW ("Rainboard"), I thought, "A bulletin board for CoCo; what a great idea!" My own BBS! Why, I could use only the *Remoterm* program and access my computer from work using my TRS-80 *VIDTEX* terminal.

Ah, but there is a catch; I don't have an auto-answer modem. Hmm, I do have Radio Shack's Modem I. What follows is my attempt to add auto-answer to the TRS-80 Modem I.

This circuit is just that — an auto-answer. As long as it receives the caller's carrier tone it will stay on line. When the caller switches her modem off or hangs up the phone, the circuit disconnects. You cannot hang up from program control.

The complete unit fits inside your Modem I under the main board and draws power from the existing power supply. All of the parts can be obtained from your local Radio Shack. (See parts list.)

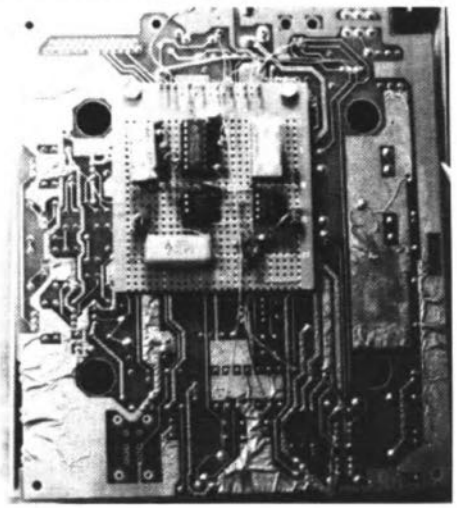
Now, here is the obligatory disclaimer: Warning: The Service Department General has determined that modification is dangerous to your warranty.

With that out of the way, please refer

to the schematic for this discussion. IC1A, IC1B and IC2 detect the ring signal from the phone line and use it to trigger the timer, IC3. R4, C3 and D3 create a delay so that the phone is not answered too quickly. The timer, IC3, is set by R6 and C5 to give the caller about 10 seconds to switch her modem on. The output of IC3 is applied to the OR gate formed by D5, D6, R10, R11 and Q1. This pulls in the relay K1 and answers the phone. The carrier detect signal from the modem is applied to point 'B' and is delayed (about two seconds) and conditioned by IC1C and IC1D. It is then applied to the OR gate at Q1. As long as the carrier detect signal is present, the relay will stay pulled in no matter what the timer does. When the caller hangs up and the carrier detect goes away, the relay drops out and the phone is released, ready for another caller.

If the circuit answers the phone too quickly or slowly for your tastes, you can change the time constant by changing the values of R4 or C3 or both. If you want more rings, increase the values. If you want fewer rings, decrease the values.

Switch S1 turns the power on to the modem and connects the carrier detect line. Mount it on the top panel of the modem under the CD light, where there is a hole. You may have to cut the lugs

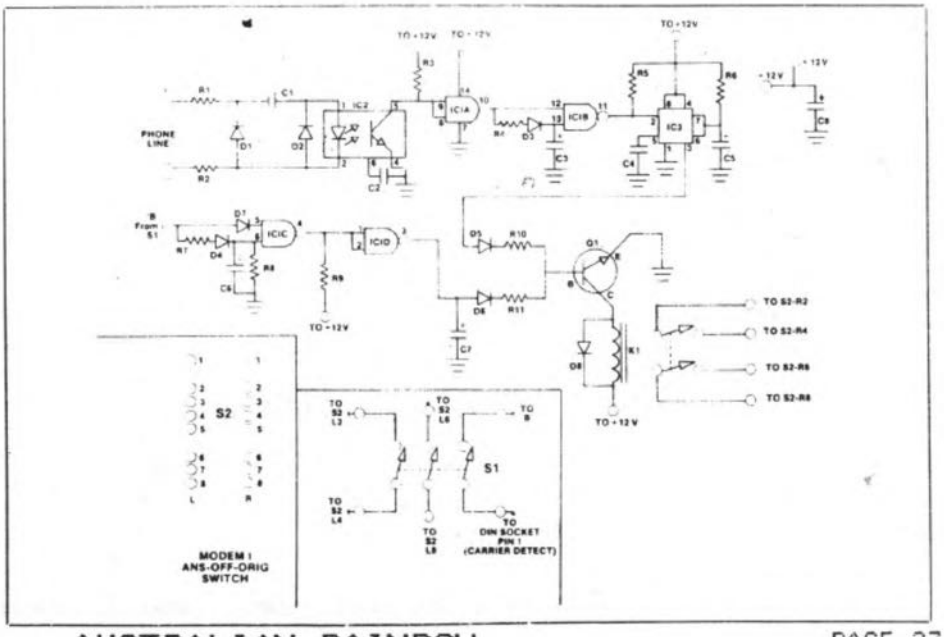


shorter to keep it from hitting the board.

Switch S2 is the existing ANS-OFF-ORIG in the modem. It must be in the "OFF" position when using auto-answer. The view of the switch terminals in the schematic is from the bottom (underside) of the modem circuit board.

The Modem I connects the phone line via a cable that plugs into your modular jack. It is at the modem board end of this cable that you make the connections I call Phone Line '+' and '-'. In my installation the red wire is positive and the green wire is negative. You should check this out with a VOM before making those connections.

The auto-answer board is powered from the modem board; so we need to locate the point I call '+12'. When you



first open the case, look at the lower center of the board and you will see U1. This is the 12-volt regulator. It is a square-looking device with three pins and is secured with a screw. With the board oriented with U1's pins to your left, the pin you want is the one nearest you. Locate this point on the bottom of the board and you have +12. Remember to connect the auto-answer board's ground to the modem board's ground.

I picked up the carrier detect signal from pin-1 of the DIN socket at the rear of the board.

The prototype was built on a small perf board and wired point to point. Parts placement is not critical. If you use IC sockets (you'll be glad you did), be sure to use the low profile type and do not use one for the relay K1; there is not enough room.

It is a good idea to bring all the external connections out to the end of the board to terminals. (I used short, stiff pieces of wire.) This makes it easier to wire it to the modem.

The auto-answer board is mounted on the underside of the modem board. Luckily there are two fairly large ground pads in just the right spot to mount the board. Solder a number six nut on each of the pads to match the spacing of the holes in the board and use two short screws to attach it. Be certain that there is enough spacing between the auto-answer board and the modem board but not too much. It's pretty tight in there.

Please be sure to check and recheck all connections. We don't want to blow up the modem! When you have everything wired up, boxed up and ready, plug the modem into the phone jack and to the computer. (It will not work if you have the printer plugged in instead!) Load in your communications program, set the mode switch to "OFF" and set the auto-answer switch (S1) to "ON." At this time the "ON" LED should be lit. Have a friend call you, but tell her to just listen and not turn her modem on. After your modem answers the phone she should hear your carrier tone for about 10 seconds after which your phone should hang up. This tests the time out timer. Now have her call you again and go on-line just like she would if calling a BBS. When she switches her modem on, your CD light should come on. You can now proceed to communicate with your new auto-answer modem! If you have any problems, turn the modem off and check all connections and wiring.

An entirely new area of data communication is now possible for you without any great expense. So, go ahead and set up your own bulletin board or remotely accessed computer. The possibilities are endless.

I will be glad to answer any questions I can if you send a SASE to Tony Sharp, 118 W. Solomon St., Griffin, GA 30223.

PART	VALUE	R.S. Part Number
R1,R2	39k .5W	271-041
R3,R5& R6,R9	1meg .25W	271-1356
R4,R7&R8	220K .25W	271-1350
R10,R11	1K .25W	271-1321
C1	.47uf 250v	272-1054
C2	470pf Disc	272-125
C3,C5&C6	10uf Tant.	272-1436
C4	.01 Disc	272-131
C7	470uf	272-957
C8	22uf, Tant.	272-1437
D1	1N4005	276-1104
D2 through D8	1N914	276-1620
IC1	CD4011	276-2411
IC2	H11A1	276-1654
IC3	NE555	276-1723
Q1	2N2222	276-2009
K1	DPDT 12v Relay	275-213
S1	3PDT Switch	275-661
Board		276-168
Sockets		Low Profile

JUNK FOOD

D. Taylor

If you have a palate for some mouth-watering fun, you'll love *Junkfood*. The object of this 16K non-Extended BASIC game is to let your hungry mouth, controlled by your right joystick, eat all the "edible" foods to gain as many points as possible — but watch out for those purple pickles, they give you more than heartburn, they're deadly!

Upon execution of *Junkfood*, the title screen will be displayed. Press the right joystick fire button to begin the game. You will have three mouths, or lives, in a game. The food scrolls in rows from left to right and you must maneuver your mouth (up and down only) to chomp as much edible food as you can. Edible food and their points are:

Hotdogs - 10 points
Hamburgers - 10 points
Green Pickles - 100 points

The menu of inedible food consists of purple pickles, which need only be touched to lose one mouth. You will be squirted with mustard and will lose a mouth if you stay between the rows of food too long.

As *Junkfood* progresses, regular food (hamburgers and hotdogs) will be replaced by purple pickles to make the game more difficult. The speed will also increase. When all regular food has been replaced and the maximum speed is reached, the game will stop momentarily and you will be rewarded 1,000 points. The round will then be reset with regular food, including green pickles, and with a few more purple pickles added between spaces of food.

If you lose a mouth in the middle of a round, the round will be reset, maintaining that level of difficulty. After losing a mouth, press the fire button to continue.

After completing *Junkfood*, the high scoreboard will appear. Your score is displayed near the top-left corner and the highest three scores will appear in the middle of the screen. If you have a high score, the new high scoreboard will be displayed. Use the joystick to control the three initial boxes by moving the joystick to the left, to decrease the order of the letters, or to the right, to increase the order. When the correct letter is displayed, press the fire button and continue for the other two boxes. To begin a new game, press the fire button.

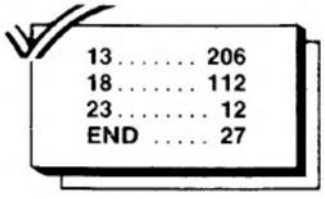
The program is actually in machine language, but you won't need an assembler because the four BASIC listings can by typed in directly.

1) Type in each listing and save it to tape. Don't run them yet. (If you have this month's RAINBOW ON TAPE you can skip this step.)

2) Now *CLOAD* each of the four listings and *RUN* them in turn. Each listing *POKE*s part of the program into memory.

3) Put in a blank tape and enter *CSAVEM "JUNKFOOD",12288,15988,12288*. This will save the machine language program onto your tape. You can now *EXEC* if you would like to play the game.

To load the game tape, just type *CLOADM:EXEC*. The finished game tape will work on a 16K CoCo with or without Extended Color BASIC. Good luck, it takes a big appetite to be a "chompion!"



Listing 1:

```
1 '*****************************
2 '          JUNKFOOD
3 '       COPR. (C) 1984
4 '       BY DAVID TAYLOR
5 '*****************************
6 'PART#1 : RUN AND LOAD PART#2
7 '*****************************
8 FORX=12288TO13295:READ Z:POKEX
,Z:NEXT
9 DATA15,113,204,128,192,221,114
,126,56,183,16,142,128,128,142,1
0,0,16,175,129,140,34,0,38,248,7
9,183,255,199,183,255,195,183,25
5,197,134,5,180,255,34,183,255,3
4,142,10,44,16,142
10 DATA59,92,189,57,74,48,136,24
,140,11,140,38,241,204,60,158,25
3,61,86,134,2,183,61,85,189,52,1
97,124,61,85,189,52,197,142,11,1
60,204,48,48,237,129,237,129,237
,132,48,136,28
11 DATA140,13,0,38,239,142,61,53
,16,142,62,148,16,191,59,201,236
,129,237,161,140,61,85,38,247,20
4,22,62,253,59,210,204,60,28,253
,59,213,204,60,201,253,60,110,20
4,60,209,253,60
12 DATA108,204,60,223,253,60,66,
204,3,33,253,60,104,204,128,128,
253,59,215,253,59,217,253,59,219
,127,59,207,127,59,238,134,48,18
3,59,205,127,61,88,189,53,112,18
9,56,32,126,51,60
13 DATA173,159,160,10,190,59,210
,166,137,1,0,129,202,38,2,134,12
8,230,137,1,160,193,181,38,2,198
,128,193,202,38,2,198,128,16,131
,128,128,39,3,253,59,215,166,137
,1,128,230,137
14 DATA1,96,16,131,128,128,39,3,
253,59,219,166,137,1,32,230,137,
1,64,16,131,128,128,39,3,253,59,
217,182,1,91,129,6,37,50,129,57,
46,92,252,61,91,195,0,1,253,61,9
```

```
1
15 DATA16,131,3,192,16,39,4,176,
16,190,59,210,142,62,117,166,169
,1,32,230,169,1,64,237,132,166,1
69,1,96,230,169,1,128,237,2,32,8
9,204,0,0,253,61,91,252,11,163,2
53,59
16 DATA239,16,190,59,210,16,140,
12,94,39,67,49,168,224,142,62,11
7,204,128,128,237,132,166,169,1,
96,230,169,1,128,237,2,126,49,15
8,204,0,0,253,61,91,252,11,163,2
53,59,239,16
17 DATA190,59,210,16,140,31,222,
39,21,49,168,32,142,62,117,166,1
69,1,32,230,169,1,64,237,132,204
,128,128,237,2,189,49,164,126,49
,216,16,191,59,210,190,59,213,13
4,19,183,59,212
18 DATA236,129,237,164,49,168,32
,122,59,212,46,244,16,190,59,210
,142,62,117,236,132,167,169,1,32
,231,169,1,64,236,2,167,169,1,96
,231,169,1,128,57,182,59,207,129
,21,39,19,124
19 DATA59,207,129,0,39,31,129,10
,16,39,0,12,189,54,166,126,50,22
3,127,59,207,126,50,223,204,60,2
8,253,59,213,16,190,59,210,126,4
9,164,204,60,148,253,59,213,16,1
90,59,210,189
20 DATA49,164,134,1,183,59,238,1
26,53,24,127,59,238,134,3,183,59
,237,16,142,59,215,236,161,132,1
43,129,128,38,14,196,143,193,128
,38,8,122,59,237,46,237,126,50,2
04,189,54,154,134
21 DATA3,183,59,237,16,142,59,21
5,236,161,132,240,129,128,38,14,
196,240,193,128,38,8,122,59,237,
46,237,126,50,237,204,48,48,253,
62,127,253,62,129,204,49,48,253,
62,131,189,50,115
22 DATA126,50,204,142,11,160,16,
142,62,121,189,50,224,127,59,206
,49,37,142,62,132,166,164,171,13
2,187,59,206,127,59,206,128,48,1
29,57,46,14,167,164,16,140,62,12
1,39,17,49,63,48
23 DATA31,32,226,128,10,167,164,
134,1,183,59,206,32,233,142,62,1
21,16,142,11,160,141,41,49,168,3
2,16,140,12,224,38,245,182,11,16
1,177,59,205,39,3,126,55,204,57,
16,190,59,210
24 DATA204,128,128,253,59,215,25
3,59,217,253,59,219,126,49,164,5
7,236,132,237,164,236,2,237,34,2
36,4,237,36,57,204,48,48,253,62.
127,253,62,131,204,48,49,253,62.
129,189,50,115,32
25 DATA203,204,62,139,253,59,233
,134,9,183,59,230,190,59,221,16,
190,59,233,48,30,134,15,183,59,2
29,236,132,237,1,48,30,122,59,22
9,46,245,166,160,167,2,122,59,23
0,46,229,57,204
26 DATA0,0,195,0,1,16,179,60,104
,38,247,57,204,33,158,253,59,221
,189,51,1,189,48,192,246,61,85,1
6,39,1,195,189,53,24,189,52,112,
189,51,243,182,61,88,16,46,3,191
,189
27 DATA51,47,189,48,192,246,61,8
5,16,39,1,166,189,53,24,189,48,1
92,246,61,85,16,39,1,153,189,53,
24,252,59,221,131,6,128,16,131,7
,158,39,8,16,131,10,222,39,174,3
2,175
28 DATA204,30,94,32,170,190,61,5
1,166,128,167,159,60,110,140,62,
212,39,35,191,61,51,129,1,39,53,
129,2,39,54,129,3,39,55,129,4,39
,36,204,60,235,237,159,60,108,19
5,0,3
29 DATA237,159,60,66,32,68,190,6
0,104,140,0,1,39,6,48,136,236,19
1,60,104,189,53,66,189,53,66,142
,62,180,32,196,204,61,15,32,213,
204,60,68,32,208,204,60,112,32,2
03,204,60
```

```
17..... 72
END ..... 29
```

Listing 2:

```
1 '*******************************
2 '            JUNKFOOD
3 '        COPR. (C) 1984
4 '        BY DAVID TAYLOR
5 '*******************************
6 'PART#2 :RUN AND LOAD PART#3
7 '*******************************
8 FORX=13296TO14255:READ Z:POKEX
,Z:NEXT
9 DATA112,32,198,108,159,60,106,
166,159,60,106,129,4,39,150,37,7
2,129,7,37,5,79,167,159,60,106,1
66,159,60,110,129,4,39,75,134,9,
16,190,59,233,49,40,174,159,60,6
6,230,132
10 DATA231,164,48,4,49,63,74,46,
245,236,159,60,66,16,163,159,60,
108,39,9,131,0,1,237,159,60,66,3
2,11,236,159,60,108,195,0,3,237,
159,60,66,57,16,190,59,233,49,40
,198
11 DATA128,231,164,49,63,16,188,
59,233,45,237,32,244,134,9,16,19
```

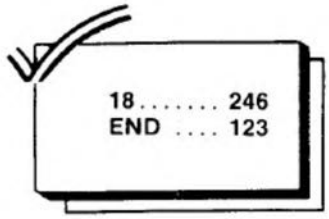
```
0,59,233,49,40,198,128,231,164,4
9,63,74,46,247,32,185,190,60,106
,140,60,199,39,7,48,1,191,60,106
,32,6,142
12 DATA60,193,191,60,106,190,60,
110,140,60,206,39,7,48,1,191,60,
110,32,6,142,60,200,191,60,110,1
90,60,108,140,60,219,39,7,48,2,1
91,60,108,32,6,142,60,207,191,60
,108,190
13 DATA60,66,140,60,233,39,7,48,
2,191,60,66,32,6,142,60,221,191,
60,66,57,246,61,85,90,88,79,195,
10,30,31,2,190,61,86,134,10,183,
59,212,236,129,237,164,49,168,32
,122
14 DATA59,212,46,244,57,204,10,0
,253,61,86,246,61,85,193,1,39,9,
189,52,197,204,60,158,253,61,86,
122,61,85,141,112,246,61,85,39,6
,189,55,152,189,56,32,16,142,12,
94,126
15 DATA50,204,189,59,79,126,57,2
4,134,3,183,59,237,16,142,59,215
,236,161,132,240,129,224,16,39,1
,39,196,240,193,224,16,39,1,31,1
22,59,237,46,233,182,59,238,129,
1,16,39,252
16 DATA217,57,190,61,89,166,132,
129,3,37,14,129,4,38,14,246,61,8
8,193,1,37,7,124,61,88,198,5,231
,132,48,1,140,62,212,39,4,191,61
,89,57,142,62,180,124,61,88,32,2
44
17 DATA16,142,128,128,142,12,254
,16,175,129,140,34,0,38,248,16,1
91,12,158,16,191,12,190,16,191,1
2,222,204,3,33,253,60,104,204,0,
0,253,61,91,127,61,88,142,60,186
,16,142,60
18 DATA193,166,128,167,160,16,14
0,60,200,38,246,142,62,148,16,14
2,62,180,16,191,61,89,236,129,23
7,161,140,62,180,38,247,204,128,
128,253,62,117,253,62,119,253,62
,139,253,62,141,253,62
19 DATA143,253,62,145,183,62,147
,57,252,59,239,16,179,11,163,39,
15,204,0,0,253,61,91,252,11,163,
253,59,239,126,49,40,204,60,28,2
53,59,213,16,190,59,210,189,49,1
64,204,9,192
20 DATA142,59,210,16,163,132,35,
56,131,1,128,31,1,16,131,10,192,
39,54,16,142,61,93,236,164,237,1
32,236,34,237,2,49,36,48,136,32,
16,140,61,121,38,237,48,136,132,
95,134,159
21 DATA167,128,92,193,27,38,249,
189,56,2,189,59,79,126,52,229,19
5,3,64,16,131,36,0,38,186,204,0,
0,253,61,91,126,49,40,142,16,1,1
27,59,209,16,142,61,121,127,59,2
08,236
22 DATA161,195,96,96,237,129,124
,59,208,182,59,208,129,14,38,239
,124,59,209,246,59,209,193,12,38
,16,16,140,62,117,39,17,127,59,2
08,127,59,209,48,4,32,213,49,168
,228,48,4,32
23 DATA203,189,55,243,189,59,79,
126,52,229,204,0,184,253,62,136,
142,59,245,191,59,199,190,59,199
,182,255,35,138,8,183,255,35,230
,128,39,9,247,62,133,189,54,191,
191,59,199,57,52
24 DATA80,206,62,133,142,59,243,
191,59,241,174,67,166,159,59,241
,39,73,198,255,61,132,252,186,25
5,32,183,255,32,230,196,48,31,38
,12,16,190,59,241,49,33,16,191,5
9,241,174,67,90
25 DATA38,237,126,54,245,126,54,
248,126,54,251,18,198,3,244,255,
32,247,255,32,230,196,48,31,38,1
2,16,190,59,241,49,33,16,191,59,
241,174,67,90,38,237,32,177,53,8
0,57,129,3
26 DATA16,38,252,59,204,32,0,253
,62,136,142,60,17,191,59,199,134
,20,183,59,204,142,59,154,16,142
,11,170,189,50,224,48,12,49,38,1
89,50,224,48,20,49,168,26,16,140
,13,10,38
27 DATA235,189,54,166,142,59,160
,16,142,11,170,189,50,224,48,6,4
9,38,189,50,224,48,26,49,168,26,
16,140,13,10,38,235,189,54,166,1
22,59,204,46,189,204,48,48,253,6
2,127,253,62
28 DATA131,204,49,48,253,62,129,
189,50,115,189,53,112,141,36,189
,56,132,189,56,132,126,51,95,142
,60,193,246,255,0,193,126,39,13,
193,254,39,9,48,1,140,60,200,39,
235,32,236,191
```

```
17..... 80
END ..... 29
```

Listing 3:

```
1 '*****************************
2 '          JUNKFOOD
3 '       COPR. (C) 1984
4 '       BY DAVID TAYLOR
5 '*****************************
6 'PART#3 : RUN AND LOAD PART#4
7 '*****************************
8 FORX=14256TO15215:READ Z:POKEX
```

```
,Z:NEXT
9 DATA60,106,57,142,10,0,16,142,
11,170,189,50,224,49,38,189,50,2
24,49,168,26,16,140,13,10,38,239
,57,68,72,177,11,161,39,1,57,182
,61,85,129,7,38,1,57,124,61,85,1
89
10 DATA52,197,182,59,205,139,2,1
29,58,39,4,183,59,205,57,134,48,
32,248,204,160,0,253,62,136,142,
59,251,191,59,199,126,54,166,182
,255,35,138,8,183,255,35,204,2,8
8,253,62,136
11 DATA142,59,253,230,128,39,8,2
47,62,133,189,54,191,32,244,57,1
82,255,35,138,8,183,255,35,204,6
4,0,253,62,136,198,63,247,62,133
,189,54,191,204,104,0,253,62,136
,189,54,191,204
12 DATA64,0,253,62,136,198,67,24
7,62,133,189,54,191,204,84,0,253
,62,136,198,75,247,62,133,189,54
,191,204,104,0,253,62,136,198,85
,247,62,133,189,54,191,204,0,0,1
95,0,1,16
13 DATA131,64,0,38,247,204,32,0,
253,62,136,198,41,247,62,133,189
,54,191,57,134,32,183,59,203,190
,59,201,166,132,129,4,38,15,198,
5,231,132,48,7,140,62,179,34,21,
191,59,201
14 DATA57,122,59,203,39,247,48,1
,140,62,180,38,223,142,62,148,32
,218,48,136,224,32,230,206,37,0,
16,206,38,0,79,183,255,198,183,2
55,200,183,255,203,183,255,204,1
83,255,206,183,255
15 DATA208,183,255,210,183,255,1
92,183,255,194,183,255,196,134,5
,180,255,34,138,8,183,255,34,16,
142,175,175,142,8,0,16,175,129,1
40,10,0,38,248,142,8,12,16,142,5
9,92,189,57,74
16 DATA142,8,75,189,57,74,142,8,
111,189,57,74,142,8,138,189,57,7
4,189,59,27,126,48,10,142,11,160
,16,142,8,32,189,50,224,16,142,1
75,175,142,8,192,16,175,129,140,
10,0,38
17 DATA248,79,183,255,198,183,25
5,194,183,255,196,189,57,87,189,
58,139,189,58,215,189,55,152,126
,48,10,236,161,16,131,128,0,39,4
,237,129,32,244,57,142,8,32,16,1
42,59,172,166,160
18 DATA161,128,34,9,37,51,16,140
,59,178,38,242,57,142,8,32,16,14
2,59,178,166,160,161,128,34,9,37
,39,16,140,59,184,38,242,57,142,
8,32,16,142,59,184,166,160,161,1
28,34,8
19 DATA37,41,16,140,59,190,38,24
2,57,189,57,205,189,58,45,189,58
,71,57,189,57,205,142,59,178,16,
142,59,184,189,58,59,142,59,193,
16,142,59,196,189,58,90,57,189,5
7,205,16,142
20 DATA59,190,189,58,62,16,142,5
9,199,189,58,98,57,142,8,233,16,
142,59,132,189,57,74,16,142,9,46
,134,128,167,164,167,33,167,34,1
34,77,167,164,173,159,160,10,204
,0,0,195,0
21 DATA1,16,131,64,0,38,247,246,
1,90,193,6,37,26,193,57,46,32,18
2,255,0,129,126,39,4,129,254,38,
218,49,33,16,140,9,49,39,23,126,
57,227,166,164,129,65,39,201,74,
126
22 DATA57,229,166,164,129,90,39,
191,76,126,57,229,57,142,59,178,
16,142,59,184,189,50,224,48,26,4
9,58,189,50,224,142,8,32,49,58,1
89,50,224,57,142,59,193,16,142,5
9,196,236,132
23 DATA237,164,166,2,167,34,48,2
9,49,61,236,132,237,164,166,2,16
7,34,142,9,46,49,61,236,132,237,
164,166,2,167,34,16,142,59,148,1
89,57,74,142,8,233,16,142,59,148
,189,57,74
24 DATA142,8,245,16,142,59,148,1
89,57,74,57,142,8,235,16,142,59,
136,189,57,74,142,9,41,134,49,16
7,132,76,167,136,64,76,167,137,0
,128,16,142,59,190,48,3,236,161,
237,129,166
25 DATA160,167,132,48,136,62,16,
140,59,199,38,239,142,59,172,16,
142,9,49,189,50,224,48,6,49,168,
64,189,50,224,48,6,49,168,64,189
,50,224,57,142,8,192,204,246,246
,189,58,245
26 DATA142,8,192,204,249,249,189
,58,245,246,255,0,193,126,39,4,1
93,254,38,227,57,237,132,237,137
,1,32,48,2,140,8,224,38,243,48,3
1,237,132,48,136,32,140,9,255,38
,246,204,0
27 DATA0,195,0,1,16,131,48,0,38,
247,57,16,190,60,106,190,61,51,4
8,1,140,62,212,39,25,49,33,16,14
0,60,200,38,4,16,142,60,193,246,
255,0,193,126,39,11,193,254,39,7
28 DATA32,224,142,62,180,32,238,
191,61,51,16,191,60,106,57,204,0
,0,195,0,1,16,131,255,255,38,247
,57,10,21,14,11,6,15,15,4,128,0,
67,79,80,82,110,96,113,121,120,1
16
```



Listing 4:

```
1 '*******************************
2 '            JUNKFOOD
3 '        COPR. (C) 1984
4 '        BY DAVID TAYLOR
5 '*******************************
6 'PART#4 : RUN AND EXEC12288
7 '*******************************
8 'OR, BEFORE 'EXEC'ING : CSAVEM
"JUNKFOOD",12288,15988,12288
9 '*******************************
10 FORX=15216TO15988:READ Z:POKE
X,Z:NEXT
11 DATA128,0,66,89,128,0,68,65,8
6,73,68,96,84,65,89,76,79,82,128
,0,14,5,23,32,8,9,7,8,32,19,3,15
,18,5,128,0,175,175,175,175,128,
0,66,79,78,85,83,32
12 DATA2,15,14,21,19,32,58,32,49
,48,48,48,48,48,48,48,48,48,48,4
8,48,48,48,48,48,48,48,48,48,48,
32,32,32,32,32,32,32,32,32,0,0,0
,0,0,0,0,0,0,0
13 DATA0,0,0,0,0,0,0,128,128,128
,128,128,128,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,4
2,31,42,28,38,0,255,0,64,58,64
14 DATA58,52,58,52,46,52,46,40,4
6,40,34,40,34,28,34,28,0,64,58,6
4,58,64,58,64,58,64,58,0,128,128
,128,128,186,128,191,128,202,186
,202,186,128,186,128,186,128,186
,128,186
15 DATA128,186,128,186,128,186,1
81,128,181,128,202,128,186,128,1
28,128,128,128,0,0,144,149,154,1
44,144,159,159,144,149,159,159,1
54,159,159,159,159,181,191,191,1
86,191,191,191,191,181,191,191,1
86
16 DATA159,159,159,159,159,159,1
59,159,3,32,0,0,0,0,0,0,128,128,
128,128,128,128,128,128,128,138,
128,138,133,138,128,143,143,133,
133,133,133,138,143,138,128,133,
143,128,128,128,128,128
17 DATA128,128,128,128,128,128,1
28,128,128,128,128,128,128,128,1
86,128,191,128,202,186,202,186,1
28,186,128,186,128,186,128,186,2
02,186,191,128,128,128,128,128,1
28,128,128,128,2,1,2,1,0,3
18 DATA3,38,182,9,9,66,78,69,1,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,224,224,224,224,224
19 DATA224,224,224,224,234,224,2
34,229,234,224,239,239,229,229,2
29,229,234,239,234,224,229,239,2
24,224,224,224,224,224,224,224,2
24,176,176,176,176,176,176,176,1
76,176,154,149,176,181,159,159,1
86,191
20 DATA159,159,191,181,159,159,1
86,176,159,159,176,176,149,154,1
76,176,176,176,176,0,0,4,2,5,4,5
,1,4,2,4,3,5,2,5,1,4,2,1,1,5,4,2
,5,1,5,4,1,1
21 DATA4,3,4,2,2,0,0,0,2,0,0,0,0
,245,255,250,128,255,255,255,240
,255,175,255,255,255,175,255,255
,255,175,255,255,255,255,255,240
,245,255,250,128,129,131,131,131
,131,130,128
22 DATA129,130,128,128,128,129,1
30,129,131,131,131,131,130,128,1
29,131,131,131,131,130,128,133,1
42,140,140,140,143,130,133,138,1
28,128,128,133,138,133,142,140,1
40,140,143,130,133,142,140,140,1
40,143
23 DATA130,133,138,128,128,128,1
33,138,133,138,128,128,128,133,1
38,133,138,128,128,128,133,138,1
33,138,128,128,128,133,138,133,1
38,128,128,128,135,136,133,138,1
28,128,128,133,138,133,138,128,1
28,128
24 DATA135,136,133,138,128,128,1
28,135,136,133,143,143,143,143,1
38,128,133,138,128,128,128,133,1
38,133,143,143,143,143,136,128,1
33,143,143,143,143,136,128,133,1
38,128,128,128,141,130,133,138,1
28,128
25 DATA128,133,138,133,138,128,1
28,133,138,128,133,138,128,128,1
28,128,128,133,138,128,128,128,1
33,138,133,138,128,128,128,133,1
38,133,138,128,128,128,143,128,1
33,138,128,128,128,128,128,133,1
39,131
26 DATA131,131,143,136,128,141,1
31,131,131,142,128,133,138,128,1
28,128,133,138,133,138,128,128,1
28,128,128,132,140,140,140,140,1
36,128,128,128,140,140,140,128,1
28,132,136,128,128,128,132,136,1
32,136
27 DATA128,128,128,128,128
```

RAINBOARD

By Lane Lester

Are you the SYSOP of a RainBoard? Have you called a RainBoard yet? Do you know what a RainBoard is? Last November's issue of THE RAINBOW was also a data communications issue, and I was pleased to present a set of programs to enable you to run your own bulletin board system. The RainBoard provides not only the usual functions such as message exchange, text files to read, and programs to download; it also, in keeping with its name, provides color graphics and begins each session with a picture of a rainbow ending in a pot of gold with the message, "WELCOME TO THE RAINBOARD, WITH A RAINBOW OF COLOR AND A POT OF GOLD IN GOOD TIMES!" Also included in that same issue was Dan Downard's machine language program that interfaced my BASIC programs to the CoCo's RS-232 port.

Almost as an afterthought, I placed at the end of the article an offer of a disk with all of the programs and files needed to run your own RainBoard. The price of $20 was what I figured would take care of the nuisance and expense of printing a cover letter, copying the RainBoard disk, and mailing it. In addition to the hundreds of copies of RAINBOW ON TAPE that were sold for that issue, I have now sent out over 80 copies of the RainBoard disk, and the orders continue to come in. I have also been besieged by telephone calls from all over the United States and Canada from folks who have typed in the programs and either had problems or just want to chat about running a bulletin board. Evidently BBSing is one of the hottest new uses for personal computers.

In addition to the RainBoards scattered all over the United States, including one in Hawaii being SYSOPed by a retired longshoreman, the most colorful BBS in the world has now gone international. Somewhere in the Pacific, a U.S. Navy ship's computer users are communicating colorfully. Known RainBoards are in Canada and Australia, and not only in English-speaking countries. In August, I received a disk from Dr. Joao Araujo, Rio de Janeiro, Brazil, containing a Portuguese version. They have a 200-member CoCo club and have translated the software (the text portions — the programs are still in BASIC) for a BBS to support their club.

I think one of the major attractions of the RainBoard was that it provided a breakthrough in the cost of starting up one's own board. In the past, prospective SYSOPs had to plan on spending several hundred dollars for an auto-answer modem, about a thousand on two or more disk drives, and over a hundred on BBS software. The RainBoard software is inexpensive (only the back-issue price of THE RAINBOW, if you feel like doing some typing), it only requires one drive, and auto-answer modems are now available for about $100. [Look elsewhere in this issue for a hardware project to convert your Modem I to auto-answer.]

Has a program ever been written without bugs? Not any of mine, and *RAINBORD/BAS* was no exception. Most of the bugs were cornered before the November 1983 issue was published, but one particularly troublesome one got through. Another RainBoard SYSOP had to call it to my attention, and it was the result of my fondness for eight-character filenames. The routines that search the disks for files use DSKI$ to look at the directory track, and would not find any files with shorter names. So at the end of lines 1030 and 1160 one needs to add:

```
FILE$ = LEFT$(FILE$+
STRING$(7,32),8)
```

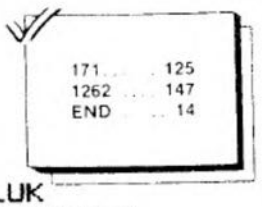
to pad out the filename with as many blank spaces as needed. There are plenty of ways this could be accomplished, but I'm indebted to Mel Hefter, the genius behind Custom Software Engineering, for this elegant approach. Another bug that was corrected in a later RAINBOW was a single byte in Dan Downard's *REMOTE/BIN*, which set the Baud rate incorrectly in the machine language driver. The correct value at $3F01 is $B8, and the easiest way to handle this if you get it on a back issue of RAINBOW ON TAPE is to *LOADM* the program, enter *POKE &H3F01,&HB8*, and then *SAVEM* the corrected program.

Although I did, at one time, operate a RainBoard I had to do it with my only CoCo and on a shared telephone line. So, when our town got a 24-hour BBS, I closed the RainBoard down, and have continued to enjoy BBSing as a caller of that board and others around the country. I would love to give you the phone number of our local board in hopes of communicating with some of you, but we have had a phenomenal run of bad luck with the equipment. In addition to the usual kinds of glitches, the equipment was once totally destroyed by lightning. We are also plagued by a "cracker," one of those perverts who derive pleasure from destroying other people's systems. Anyway, I'm afraid that any number I might give you would no longer be in operation. If you would like to see a RainBoard in action, you may call either (813) 321-0397 or (412) 654-0445.

For jaded RainBoard SYSOPs, or for anyone who needs a little more encouragement, here's a new feature that can be added to the RainBoard. The original version kept the entire membership list in RAM, limiting the size. The patch (called *PATCH/BAS*) which follows provides for a direct access member file of up to 300 members. *INITMEMB/BAS* initiates the file, *EDITOR/BAS* edits the file, and *SYSOP2/BAS* replaces the original *SYSOP/BAS*. These new routines are the products of my computing colleague, Erik Gavriluk, who helped me immensely in getting the original RainBoard in shape.

In case you've decided you'd like to operate your own RainBoard, you can send $20 to Lane Lester, 413 Woodland Circle, Lynchburg, VA 24502 for a disk containing all the programs you need, plus documentation and text files to show the kinds of things that are usually included. Alternately, if you weren't a RAINBOW subscriber last year, back issues of both the November 1983 magazine and the companion RAINBOW ON TAPE are available from THE RAINBOW and, together, these contain the bare necessities you need to set up your own bulletin board.

Either way, you can quickly and easily become the SYSOP of your own system.



Listing 1

```
Ø ' PATCH, BY ERIK GAVRILUK
1 ' ENABLES RANDOM ACCESS MEMBER
SHIP FILE FOR THE RAINBOARD.
2 ' REMARK FOLLOWING THE LINE EX
PLAINS WHAT SHOULD BE DONE,
3 '     ,E.G., CHANGED = CHANGE T
HIS LINE TO READ...ETC.
4 ' IF NOTHING ELSE IS LISTED AF
TER A LINE, INSERT THAT LINE.
5 ' THIS FILE CAN BE MERGED WITH
 THE EXISTING RAINBORD/BAS.
6 ' RUN INITMEMB/BAS BEFORE USIN
G THIS NEW MODIFIED RAINBOARD.
20 GOTO 1700'CHANGED
70 CLEAR5000:DIMDISPLAY$(46),TEX
T$(70)'Modified line
100 CLS:PRINT"RAINBOARD IS READY
 TO RECEIVE!"'Replace
131 LINE INPUT"ARE YOU USING A T
RS-80 COCO (Y/N)?";CC$:IF CC$="N
" OR CC$="n" THEN BITS=7:GOTO 15
0'Insert line
132 IF CC$="Y" OR CC$="y" THEN 1
40 ELSE 131'Insert line
170 CLOSE:PRINT"PRESS ENTER TO B
ECOME A MEMBER":LINE INPUT"OR TY
PE YOUR LOGON NUMBER:";LN$:IF LN
$="" THEN 1500 ELSE V=VAL(LN$):I
F V<1 THEN 170 ELSE IF V>300 THE
N 170'Replace
171 OPEN"D",#1,"MEMBERS/TXT",25:
FIELD 1,16 AS NM$,6 AS PW$,3 AS
IN$:GET #1,V'INSERT
172 I$=IN$:LINE INPUT"ENTER YOUR
 PASSWORD:";P$:IF P$=PW$ THEN NA
ME$=NM$:PRINT"HELLO, "NAME$" ("I
$")":GOTO 210 ELSE 170'INSERT
210 INIT$=I$:CLOSE:PRINT"CHECKIN
G FOR MESSAGES.":C=1:GOSUB650:GO
TO360'Modified line
700 MSG$="WOULD YOU LIKE"+CHR$(1
3)+"TO REPLY TO THIS (Y/N)? ":GO
SUB40:IFC$="Y"THENGOSUB740'MODIF
IED
710 NEXTL,K:IFCD THENRETURNELSEP
RINT"SORRY, NO MESSAGES FOUND.":
RETURN'CHANGED
770 PRINT"WE NEED 3 LETTERS.":GO
TO 760'CHANGED
810 IFLEN(S$)>8THENPRINT"8 LETTE
R MAXIMUM, "NAME$:GOTO 800'CHANG
ED
820 IFINSTR(S$,":")ORINSTR(S$,"Ø
")ORINSTR(S$,"/")ORINSTR(S$,".")
THENPRINT"PLEASE DO NOT USE: Ø .
 : /":GOTO 800'CHANGED
900 PRINTSTRING$(3,7)"THAT LINE'
S TRUNCATED TO:"'CHANGE
1260 PRINTTAB(6)"***THE RAINBIRD
S***"'CHANGED
1261 OPEN"D",1,"MEMBERS/TXT",25:
FIELD 1,16 AS NM$,6 AS PW$,3 AS
I$'INSERT
1262 FOR X=1 TO 300:GET #1,X:IF
PW$=STRING$(6,32) THEN CLOSE:RET
URN ELSE PRINTNM$;"(";I$;")":NEX
T X:CLOSE:RETURN:'INSERT
1410 MBR=Ø:OPEN"D",1,"MEMBERS/TX
T:Ø",25:FIELD 1,16 AS NM$,6 AS P
W$,3 AS IN$:FOR X=1 TO 300:GET #
1,X:IF PW$=STRING$(6,32) THEN CL
OSE:RETURN ELSE IF I$=IN$ THEN M
BR=-1:CLOSE:RETURN ELSE NEXT:CLO
SE:RETURN
1420 NEXT:CLOSE:RETURN
1450 PCLEAR1:GOTO 70'DELETE
1500 OPEN"D",1,"MEMBERS/TXT",25:
FIELD 1,16 AS NM$,6 AS PW$,3 AS
IN$
1501 LINE INPUT"ENTER YOUR FULL
NAME:";N$
1502 LINE INPUT"ENTER 3 INITIALS
 WE SHOULD ADDRESS MAIL TO:";I$:
IF LEN(I$)<>3 THEN PRINT"THREE L
ETTERS":GOTO 1502
1503 PRINT"WORKING .. PLEASE WAI
T"
1504 FOR X=1 TO 300:GET #1,X:IF
LEFT$(NM$,LEN(N$))=N$ THEN 1509
ELSE IF IN$=I$ THEN 1510 ELSE IF
 PW$=STRING$(6,32) THEN 1505 ELS
E NEXT X
1505 LINE INPUT"ENTER SIX LETTER
S FOR A PASSWORD:";P$
1506 IF LEN(P$)<>6 THEN PRINT"SI
X LETTERS PLEASE":GOTO 1505
1507 LSET NM$=N$:LSET PW$=P$:LSE
T IN$=I$:PUT #1,X:CLOSE:PRINT"YO
UR LOGON NUMBER IS";X
1508 NAME$=N$:GOTO 210
1509 PRINT"SOMEONE ALREADY HAS T
HIS NAME":GOTO 1511
1510 PRINT"SOMEONE CURRENTLY HAS
 THESE INITIALS"
1511 PRINT"PLEASE TRY AGAIN":GOT
O 1501
1700 PCLEAR 1:GOTO 70
```

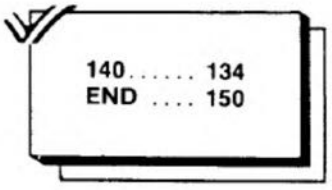
Listing 2

```
10 ' Program to initialize rando
m access
20 ' Membership file for RAINBOR
D/BAS
30 ' BY ERIK GAVRILUK
40 CLS:PRINT"INITIALIZING MEMBER
```

```
S/TXT FILE."
50 OPEN"D",#1,"MEMBERS/TXT:0",25
60 FIELD 1,16 AS NM$,6 AS PW$,3
AS IN$
70 FOR X=1 TO 300:LSET NM$=STRIN
G$(16,32):LSET PW$=STRING$(6,32)
:PUT #1,X:NEXT X
80 CLOSE #1
90 PRINT"FILE INITIALIZED.. YOU
HAVE ROOM FOR 300 USERS"
```

Listing 3

```
5 'EDITOR - BY ERIK GAVRILUK
10 CLS:PRINT"USERSLOG EDITOR FOR
 RAINBOARD"
20 OPEN"D",#1,"MEMBERS/TXT",25:F
IELD 1,16 AS NM$,6 AS PW$,3 AS I
N$
30 REC=1
40 CLS:PRINT"RECORD #"REC:GET #1
,REC
50 PRINT"NAME:"NM$:PRINT"PASSWOR
D:"PW$:PRINT"INITIALS:"IN$
60 PRINT:PRINT"1.NAME,2.PW,3.INI
T,4.NEXT,5.END"
70 LINE INPUT A$:ON VAL(A$) GOTO
 100,200,300,400,500
80 GOTO 60
100 LINE INPUT"NAME:";N$:LSET NM
$=N$:PUT #1,REC:GOTO 40
200 LINE INPUT"PASSWORD:";P$:LSE
T PW$=P$:PUT #1,REC:GOTO 40
300 LINE INPUT"INITIALS:";I$:LSE
T IN$=I$:PUT #1,REC:GOTO 40
400 REC=REC+1:GOTO 40
500 CLOSE:END
```



Listing 4

```
10 'SYSOP PROGRAM
20 CLEAR5000:DIMNAME$(50)
30 CLS:PRINT"ENTER NUMBER OF FUN
CTION:","1. DISPLAY ACTIVITY FIL
E","2. PRINT ACTIVITY FILE","3.
INITIATE ACTIVITY FILE"
40 PRINT"4. DISPLAY MEMBER FILE"
,"5. PRINT MEMBER FILE","6. STAR
T MEMBER FILE","7. REMOVE MEMBER
S","8. KILL OLD MESSAGES","9. EN
D
50 INPUTK:IFK<1ORK>9THEN30ELSEON
K GOSUB60,60,100,110,110,140,160
,210,250:GOTO30
60 IFK=1THENDV=0ELSEDV=-2
70 OPEN"D",1,"ACTIVITY":L=LOF(1)
:PRINT#DV,"CALLERS ="L
80 FORI=1TOL:GET#1,I:INPUT#1,ACT
$:FORJ=255TO0STEP-1:IFMID$(ACT$,
J,1)=" "THENNEXTJ ELSEACT$=LEFT$
(ACT$,J)
90 PRINT#DV,ACT$:NEXTI:CLOSE:LIN
EINPUT"PRESS ENTER TO CONTINUE";
K$:RETURN
100 KILL"ACTIVITY/DAT":RETURN
110 IFK=4THENDV=0ELSEDV=-2
120 OPEN"D",#1,"MEMBERS/TXT",25:
FIELD 1,16 AS NM$,6 AS PW$,3 AS
IN$
121 FOR I=1 TO 300:GET #1,I:IF P
W$=STRING$(6,32) THEN 130 ELSE P
RINT#DV,NM$;"(";IN$;")";" ==>";P
W$:NEXT I
130 CLOSE:LINEINPUT"PRESS ENTER
TO CONTINUE";K$:RETURN
140 CLS:LINE INPUT"SYSOP NAME:";
NA$:LINE INPUT"INITIALS:";I$:LIN
E INPUT"PASSWORD:";P$:OPEN"D",#1
,"MEMBERS/TXT",25:FIELD 1,16 AS
N$,6 AS PW$,3 AS IN$
141 LSET N$=NA$:LSET PW$=P$:LSET
 IN$=I$:PUT #1,1:LSET PW$=STRING
$(6,32):PUT #1,2:CLOSE
150 RETURN
160 K=4:GOSUB110
170 LINE INPUT"ENTER INITIALS OF
 MEMBER TO     REMOVE, X TO RETU
RN:";I$:OPEN"D",#1,"MEMBERS/TXT"
,25:FIELD 1,16 AS NM$,6 AS PW$,3
 AS IN$
171 FOR X=1 TO 300:GET #1,X:IF I
N$=I$ THEN 175 ELSE NEXT X:CLOSE
:PRINT"NOT FOUND":RETURN
175 LSET PW$="....."+CHR$(255):P
UT #1,X:CLOSE:RETURN
180 WRITE#1,NUM:FORI=1TONUM:WRIT
E#1,NAME$(I):NEXT:CLOSE:RETURN
190 J=1:FORI=1TONUM:IFLEFT$(NAME
$(I),3)=INITS$THENI=I+1:NUM=NUM-
1
200 NAME$(J)=NAME$(I):J=J+1:NEXT
:GOTO170
210 'Kill Old Messages
220 CLS:PRINT"ENTER NUMBER OF CU
RRENT MONTH:":INPUTMONTH
225 FORI=3TO11:DSKI$1,17,I,A$,B$
:A$=A$+LEFT$(B$,120):FORJ=0TO7:S
BJECT$=MID$(A$,J*32+1,8):EXT$=MI
D$(A$,J*32+9,3)
230 A=ASC(SBJECT$):IFA=255THENJ=
7:I=11:GOTO240ELSEIFA=0THEN240
235 IFMID$(EXT$,2,1)="*"OR(LEFT$
(EXT$,1)="A"ANDVAL(RIGHT$(EXT$,2
))<MONTH-1)THENKILLSBJECT$+"/"+E
XT$+":1"
240 NEXTJ,I:RETURN
250 END
```

ROAD RACE

By Shane Franklin

(Shane Franklin is a 15-year-old sophomore at Marshall Sr. High School, Marshall, Texas, who became interested in computers about two years ago. After receiving his computer, he has become a "computer addict" and plans a career in this field.)

Here's a game that all racing fans might enjoy. All it takes is a 16K Extended Color Computer, a joystick and a little time. The rules are as follows:

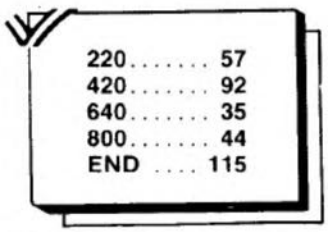
Limits of Gears

Maximum speed in first gear is 25 MPH.
Maximum speed in second gear is 50 MPH.
Maximum speed in third gear is 75 MPH.

Accelerate — push stick forward (up)
Decelerate — pull back (down)
Gear Up — push stick up and press button
Gear down — push stick down and press button
Note: If HP gets over 8000, you will blow your engine.

When you load the program you will see the track and a lot of numbers. The numbers are the maximum speed for that turn. The program is a little slow, but, it takes a while for the computer to show the gear, speed and horse power. If you want the game to go faster you will have to take out the part of Line 670, which makes the sound.

Have fun, but don't blow your engine!



The listing:

```
10 POKE65495,0
20 PMODE1,1:SCREEN1,0:PCLS:RESTO
RE
30 DRAW"S8"
40 N$(1)="BR2D4"
50 N$(2)="R2D2L2D2R2"
60 N$(3)="R2D2L2R2D2L2"
70 N$(4)="D2R2U2D4"
80 N$(5)="R2L2D2R2D2L2"
90 N$(6)="R2L2D4R2U2L1"
100 N$(7)="R2D4"
110 N$(8)="R2D2L2U2D4R2U2"
120 N$(9)="R2D2L2U1BD3R2U1"
130 N$(0)="R2D4L2U4"
140 G$="R2BD2L1F1G1L1H1U2"
150 S$="R2L2D2R2D2L2"
160 P$="R2D2L2U1D3"
170 H$="D4BR2U2L1R1U2"
180 COLOR2,1
190 DRAW"BM0,2R1D4L1BM4,4R19F3D1
2R26E8R59F2D2G4L22D1L2D11R1D1R30
"
200 DRAW"F2D53G2L4H2U36H6G6D29G4
L87H1U36E2R3F2D23F4R64E2R1E2R1E2
"
210 DRAW"U1H2L1H2L1H1L57H2U2E2R8
E1R3E1R3E1R3E1R3E1R3E1R21E3H12L7
0"
220 DRAW"BM4,8R17F3D11F1R29E8R56
D1G3L22G2D13F2R29F2D49G2H2"
230 DRAW"U36H6L4G6D29G4L83H1U32E
1R1F1D24F3R69E2R1E2R1E3U3H3L1H2"
240 DRAW"L1H1L57H1E1R7E1R4E1R3E1
R3E1R3E1R3E1R21E3U4H12L72"
250 DRAW"BM0,60R1D4L1"
260 PAINT(4,6),2,2
270 DRAW"C4"
280 READ A,B,C
290 DRAW"BM"+STR$(A)+","+STR$(B)
+N$(C)
300 IF B=50 AND C=5 THEN 320
310 GOTO 280
320 DRAW"C4BM46,190U7R75D7L1U6L7
3D6"
330 DRAW"U6R15D6R1U6R26D6R1U6"
340 DRAW"C4BM54,182; XG$;BM62,18
6R1"
350 DRAW"BM84,182; XS$;BM92,182;
 XP$;BM100,186R1"
360 DRAW"BM138,182; XH$;BM146,18
2; XP$;BM154,186R1"
370 DRAW"C3BM16,14R1F2D1G2L1H2U1
E2"
380 DRAW"BM+0,+8R1F2D1G2L1H2U1E2
"
390 DRAW"BM+0,+8R1F2D1G2L1H2U1E2
"
400 PAINT(16,18),2,3:PAINT(16,34
),2,3:PAINT(16,50),2,3
410 PAINT(50,180),2,4:PAINT(82,1
80),2,4:PAINT(138,180),2,4
420 FORN=1TO3000:NEXT
```

```
430 PSET(4,6,3):COLOR3,2:S=4:X=7
5
440 FORSS=1TO3:ZZ=JOYSTK(Ø)
450 PAINT(16,16*SS),S,3
460 SOUNDX,15
470 J=JOYSTK(1):IF J<15 THEN 4ØØ
480 IF SS=2 THEN S=1:X=1Ø
490 NEXTSS
500 G=1:SP=Ø:HP=Ø:TI=Ø
510 COLOR3,2
520 H=4:V=6
530 READ A,B,N,ST:FORX=1TON
540 P=PEEK(65280):IFP=126 OR P=2
54 THEN 550 ELSE 580
550 XX=JOYSTK(Ø):J=JOYSTK(1):IFJ
<25 THEN G=G+1 ELSE G=G-1
560 IF G<1 THEN G=1 ELSE IFG>4 T
HEN G=4
570 SOUND2Ø,1:IF SP<(G-1)*25 THE
N 840
580 XX=JOYSTK(Ø):J=JOYSTK(1):IFJ
<15 THEN SP=SP+G:GOTO6ØØ
590 IFJ>53 THEN SP=SP-(5-G)*4
600 HP=(4.5-G)*50*SP:IF HP>8000
THEN 840
610 IF HP<Ø THEN HP=Ø
620 IF SP<Ø THEN SP=Ø
630 TI=TI+100-SP
640 PAINT(50,180),2,4:PAINT(82,1
80),2,4:PAINT(138,180),2,4:DRAW"
BM68,182; XN$(G);"
650 SA=INT(SP/100):SB=INT(SP/10-
SA*10):SC=INT(SP/1-SB*10-SA*100)
:HA=INT(HP/1000):HB=INT(HP/100-H
A*10):HC=INT(HP/10-HB*10-HA*100)
:HD=INT(HP/1-HC*10-HB*100-HA*100
0)
660 DRAW"BM106,182; XN$(SA);BM11
4,182; XN$(SB);BM122,182; XN$(SC
);BM160,182; XN$(HA);BM168,182;
XN$(HB);BM176,182; XN$(HC); BM18
4,182; XN$(HD);"
670 SOUNDHP/40+1.1:IF ST<>Ø AND
SP>ST THEN 870
680 PRESET(H,V):H=H+A*2:V=V+B*2:
PSET(H,V,3):NEXTX:IFH=4 AND V=64
 THEN 740 ELSE 530
690 RETURN
700 PRINT:PRINT"WANT TO PLAY AGA
IN?"
710 A$=INKEY$:IF A$=""THEN 710
720 IF A$="Y" THEN 20
730 END
740 CLS:PRINT:PRINT"C O N G R A
D U L A T I O N S !"
750 PRINT:PRINT"YOU FINISHED THE
 COURSE."
760 PRINT"YOUR TIME WAS:";TI*.10
Ø
770 GOTO700
780 DATA 88,18,7,96,18,Ø,236,8,4
,244,8,5,156,28,6,164,28,5,184,4
8,6,192,48,5,238,48,6,246,48,5
790 DATA 232,178,4,240,178,5,212
,70,6,220,70,Ø,210,154,7,218,154
,Ø,18,168,6,26,168,Ø,Ø,88,4,8,88
,5,46,132,7,54,132,Ø
800 DATA 160,128,6,168,128,5,48,
96,4,56,96,Ø,84,86,9,92,86,Ø,176
,86,5,184,86,Ø,140,50,7,148,50,5
810 DATA1,Ø,18,Ø,1,1,3,65,Ø,1,11
,Ø,1,1,1,60,1,Ø,27,Ø,1,-1,8,70,1
,Ø,57,Ø,1,1,2,45,-1,1,4,45,-1,Ø,
21,Ø,-1,1,1,65,-1,Ø,1,65,-1,1,1,
65,Ø,1,11,Ø,1,1,2,65,1,Ø,29,Ø,1,
1,2,65,Ø,1,51,Ø,-1,1,2,45,-1,Ø,2
,45,-1,-1,2,45,Ø,-1,36,Ø
820 DATA-1,-1,6,60,-1,Ø,2,60,-1,
1,6,60,Ø,1,29,Ø,-1,1,4,70,-1,Ø,8
5,Ø,-1,-1,1,60,Ø,-1,34,Ø,1,-1,2,
45,1,Ø,1,45,1,1,2,45,Ø,1,23,Ø,1,
1,4,70,1,Ø,65,Ø,1,-1,2,65,1,Ø,1,
65,1,-1,2,65,1,Ø,1,65,1,Ø,1,65,1
,-1,3,65,Ø,-1,1,65,-1,-1,3,65,-1
,Ø,1,65,-1,-1,2,65,-1,Ø,1
830 DATA65,-1,-1,1,65,-1,Ø,57,Ø,
-1,-1,2,40,1,-1,2,40,1,Ø,8,Ø,1,-
1,1,90,1,Ø,3,90,1,-1,1,90,1,Ø,3,
90,1,-1,1,90,1,Ø,3,90,1,-1,1,90,
1,Ø,3,90,1,-1,1,90,1,Ø,3,90,1,-1
,1,90,1,Ø,21,Ø,1,-1,3,50,Ø,-1,2,
50,-1,-1,12,75,-1,Ø,71,Ø
840 CLS:PRINT:PRINT"B O O O O M
M ! !"
850 PRINT:PRINT"YOU BLEW YOUR EN
GINE."
860 GOTO700
870 CLS:PRINT:PRINT"C R A S H !
! !"
880 PRINT:PRINT"YOU WERE GOING T
O FAST AROUND   THAT TURN."
890 GOTO700
```

Everything You Always Wanted To Know About The Color Computer

By Andy Kluck

This month we will examine some of the peculiarities of Disk BASIC. The use of the *&H* prefix for Hex constants and *&O* or just & for Octal constants is described in the Extended BASIC manual, but it doesn't tell you what you have to do to convert Hex or Octal values contained in strings to numeric variables. This can be done by adding the prefix to the string and taking the value of the result:

```
H=VAL("&H"+H$)
O=VAL("&"+O$)
```

Concurrent Files

The Disk BASIC *OPEN* statement gives an AO Error if the specified device number is already open. In most cases it also gives this error if the same file is already open on another device number. However, an exception is made if the new *OPEN* statement uses the same access mode, either input or random (direct), as the file was previously opened with. This apparently intentional loophole allows a file to be opened and accessed through two or more device numbers at the same time as long as all *OPEN* statements use the same mode, input or random. This could be useful if you need to access two parts of a file at the same time.

COPY

For some strange reason, *COPY* is listed in a section of the Disk manual entitled "Special Multi-Drive Commands." This seems unfortunate, since such a command is the only efficient way to duplicate binary and data files. However, the *COPY* routine actually has a special provision for copying with one drive. If *COPY* is used with only one filename: *COPY "FILENAME/EXT"* instead of the syntax suggested in the Disk manual, the computer will request the user to switch disks as many times as necessary to copy the file. According to the "Disk BASIC Summary" in the back of the manual, executing *COPY* will "erase memory." This is misleading. Unlike *BACKUP* and *DSKINI*, *COPY* uses only the area of free memory between the array variables and the stack for its data storage. *COPY* does not clear variables, and it may be used within a program. There is one problem with *COPY*, either with one or two drives; because of a bug in *CLOSE*, with either Disk BASIC 1.0 or 1.1, an I/O Error in *COPY* can cause a system crash.

DIR

Disk BASIC does not support listing of a diskette's directory to the printer; however, this can be done by setting the device number at $6F to -2 and then calling the directory routine.

```
POKE 111,254:DIR
```

The two instructions should not be separated, and if executed within a program should be followed by:

```
POKE 111,0
```

to redirect output to device 0

RUN

The statement *100 RUN A$(X)* might be useful in a menu program. However, it doesn't work because Disk BASIC assumes that if the first character of whatever follows *RUN* isn't a quote, then it is a line number. The problem may be solved by adding an empty string: *100 RUN ""+A$(X)*

VERIFY

According to the Disk BASIC manual, *VERIFY ON* causes a VF Error if a sector is written incorrectly. Actually, the routine that handles the verifying does more than this. When it detects that a sector has been written in error, it does not simply give a VF Error and give up; instead it attempts to write the sector up to four more times before it gives the error. So disk BASIC with *VERIFY ON* does not just tell you when it has glitched your directory; it often can repair it immediately without you or your application program knowing what happened. Incidentally, if you get I/O Errors reading disks made without *VERIFY ON*, try: *? PEEK(&HF0)* after the errors. If you get an eight or 16, there's a good chance that *VERIFY ON* would have prevented the error. Unless your drive has a better than average reliability record, I recommend *VERIFY ON* at all times except for *BACKUP*s, which take a long, long time with verify. Instead, a fast disk scanner may be used to test the destination disk after *BACKUP*. For example, this BASIC program can usually test a disk in about 15 seconds if there are no errors.

The program assumes that the disk was formatted with a skip factor of four and attempts to read each track in one revolution of the disk. In some cases, however, depending on the current drive speed, the speed at which the disk was formatted, and the speed at which the sectors were written, the end of one sector comes too close to the beginning of the next sector. This results in much slower operation. If the program runs slower than about 2.5 tracks per second on some disks, replace Line 30 with:

```
30 S=2
```

With this setting the program will read every other sector and take two revolutions to read each track by using the sector order of Lines 320 and 330 instead of Lines 300 and 310. Finally, in order to implement *VERIFY ON* in some software packages it may be necessary to insert the *VERIFY* statement into the program; for example, with the *Telewriter*, add the line:

```
0 VERIFY ON
```

to the program: *S/XXX*.

CLOSE

There are several problems in the part of the *CLOSE* routine in the Disk BASIC 1.0 ROM used with random (direct) access. A misdirected branch at *$CACC* causes some strings fielded in buffers other than the one being closed to be deleted. Also, because of a stack mess up, any string array elements fielded in any buffer can cause unexpected results. Most importantly, whenever a random file other than the last one opened is closed, the system crashes. Therefore, if you must have two or more random files open at once, open them in ascending order and close them in descending order by device number. Remember

The listing:

```
10 FOR A=&H1DA TO &H1F8: READ I$
: V=VAL("&H"+I$):S=S+V: POKE A,V
: NEXT
20 IF S<>4040 THEN PRINT"DATA ER
ROR": STOP
30 S=1
40 FOR N=1 TO S
50 FOR A=&H1FA TO &H20B
60 READ I
70 POKE A,I
80 NEXT A
90 NEXT N
100 POKE &HEA,2 ' READ SECTORS
110 POKE &HEB,0 ' DRIVE 0
120 POKE &HEE,4: POKE &HEF,0
130 FOR TR=0 TO 34
140 POKE &HEC,TR
150 EXEC &H1DA ' READ TRACK
160 IF PEEK(&H1F9)=0 THEN 240
170 FOR A=&H1FA TO &H20B
180 SE=PEEK(A)
190 E=PEEK(A+18)
200 IF E=0 THEN 230
210 ER=ER+1
220 PRINT "ERROR";E;"ON TRACK";T
R;"SECTOR";SE
230 NEXT A
240 NEXT TR
250 PRINT ER;"TOTAL ERRORS."
260 DATA 7F,01,F9,8E,01,FA,A6,80
270 DATA 97,ED,AD,9F,C0,04,96,F0
280 DATA A7,88,11,BA,01,F9,B7,01
290 DATA F9,8C,02,0C,26,E8,39
300 DATA 1,12,5,16,9,2,13,6,17
310 DATA 10,3,14,7,18,11,4,15,8
320 DATA 1,5,9,13,17,3,7,11,15
330 DATA 12,16,2,6,10,14,18,4,8
```

that whenever an error occurs, files are closed in descending order. A similar problem may occur in *COPY* — if an I/O Error occurs while one of the files is open, *CLOSE* gets confused and causes the same crash. While the first three problems are fixed in the 1.1 Disk ROM, the last one is not.

The Case Of The Garbled Up Disk

Radio Shack devotes a whole chapter to the garbled up disks, but they omit some of the common causes of disk garbling. Starting at address *$800* there are four areas used to store the file allocation table of each drive with open files. Each area also has one byte used to count the number of open files on that drive, and one byte that counts changes in the *FAT* and indicates whether the table should be written out when a file on that disk is closed. Any loading of garbage into this area is likely to mess up these bytes such that the next time a file on the disk is closed, the garbage is written out to the disk's *FAT* sector, which is essential to accessing the files. For example, many programs designed for cassette systems load starting at *$600*. If one of these were converted to a disk file with a tape to disk program, a likely result of attempting to load it would be not just a crashed BASIC but also an unusable disk. A similar problem could occur if the program was loaded from a cassette and an attempt was made to save it on a disk. This is also what can happen if you use a certain often-published *PCLEAR 0* routine and try to *LOAD* a BASIC program. While these situations all arise out of user error in messing with BASIC's reserved areas, a similar problem can be caused by another of the bugs in Disk BASIC 1.0. Whenever the *File Allocation Table* is written out on the disk (during *CLOSE, KILL, WRITE*, etc.) the "number of open files" counter for the next higher numbered drive is set to zero by a botched instruction at *$C70C*. Because of this, any changes in the *FAT* made on a disk in one drive (due to *CLOSE, KILL*, etc.) while files are open on the next higher drive, can result in the file counter of the higher drive being decremented to a non-zero number when the files on it are closed. If this happens, any disk later placed in the higher numbered drive may be wrecked by having its *File Allocation Table* overwritten by a copy of the *FAT* from another disk. For example, if you open a file on drive 1, *SAVE* or *KILL* on drive 0, *CLOSE* all files or *UNLOAD 1*, switch disks in drive 1, and *SAVE* on drive 1, the new disk may get the *FAT* copied from the old one and require reconstruction efforts to recover files. This does not apply in a one-drive system, but if you're going to *OPEN* any files on any drive except 0, and work with other drives while they're open, I recommend using Disk BASIC 1.1, in RAM if necessary, to avoid this problem.

UNLOAD And END

The Disk BASIC manual cautions that you should use *UNLOAD* before switching disks whenever there may be files open. But all *UNLOAD* does is close all files on the specified or default drive. So unless you want to leave files on some drives open, you can save keystrokes by using *END*, which closes all disk and cassette files, or any syntax error, as the Disk BASIC error routine closes all disk files. Note also that *UNLOAD* closes files from low to high device numbers, the opposite order from *CLOSE*, so *UNLOAD* can trigger the random files crash in the 1.0 Disk ROM.

64K

As you probably know, the Radio Shack 32K CoCo contains 64K RAM chips, and with a hardware modification first published by Frank Hogg in February of '82 and eventually adopted by Radio Shack for use in the current revision circuit boards, the full 64K can be accessed through memory paging. One way to use this extra memory is to run a program like:

```
10 FOR I=0 TO 22
20 READ X: POKE 950+I,X
30 NEXT I
40 EXEC 950
50 DATA 26,80,142,128,0,183
60 DATA 255,222,236,132,183,255
70 DATA 223,237,129,140,255,0
80 DATA 38,241,28,175,57
```

to copy the BASIC interpreters into the upper half of RAM and leave the SAM in map type one so that BASIC may be modified and the area above BASIC may be used as extra RAM. Many people have assumed that without the modification, which prevents a bus conflict problem during write operations, or the new board, none of the extra memory may be accessed. It appears, however,

Continued on P 45..

CLOAD Command Fixer

By Curt Chadwick

This article is dedicated to all of you out there with 64K, no disk, and a desire to learn assembly language programming. I don't have a disk either, and I spend a great deal of my computer time waiting for my worn but trusty cassette recorder to save and load programs. One thing about BASIC that has always bothered me is the *CLOAD* command. How annoying to get an "I/O Error" and then have to keep typing *CLOAD*, or whatever method you might have devised to get to the beginning of the next program.

I've seen some rather unusual methods used to find the beginning of the next program. There's "Fast Fingers Fred" who manipulates the cassette buttons so fast as to find the break. Or . . . you could use the audio and motor commands to find the end. I leave the play button down and pull the earphone and remote plugs to find the end of the program which drives my wife crazy with the computer squawk coming from the recorder.

Wouldn't it be nice to just type *CLOAD* and have the computer search for the end of the file for you? Now that you have 64K it's time to put it to work. With 64K, BASIC is now in RAM! What that means is that if you would like to change BASIC you can! What is it about BASIC that you would like to change? I've already told you what I would like to change the *CLOAD* command.

I've had my CoCo now for two years and have had as one of my objectives to learn at least some assembly language programming. To accomplish that objective I decided to try and understand the *CLOAD* command and learn enough assembly language to perform the patch. When I started looking in BASIC to find the subroutine that handles the *CLOAD* command there wasn't much help available except from members of my local computer club, which I would like to take this opportunity to thank. Now, there are magazines which publish or advertise a disassemble of the BASIC ROM. I've found that, for the beginner, they may lack many details and leave gaps in the explanation of what goes on in BASIC. If you compare comments by different authors, you might even think they were talking about separate subroutines. However, they are the best place to start.

There are also books which can help. Lance Leventhal's book *6809 Assembly Language Programming* and *TRS-80 Color Computer Assembly Language Programming* by William Barden, Jr. The subroutines for the cassette I/O start around *&HA6F3* and go all the way to *&HA880*. By the way the "*&H*" means Hex numbers. They may be foreign for awhile, but you do get used to them. Those subroutines are called by many different BASIC routines and the problem is to find the *CLOAD* command routine and then look for a way to "fix" it.

First, let's take a look at the tape format. Check the back of *Going Ahead With Extended Color BASIC* for the ROM subroutines. The WRTLDR turns on the cassette and writes a leader. BLKOUT writes a block to cassette and there are other names like BLKTYP, CBUFAD, and several others. If those names mean anything to you, or you understand the ROM subroutine section of the manual, then you probably won't need to read this article. When I started my goal to learn more about the computer and learn some assembly language, that ROM subroutine section was a mystery.

The listing below is a less mysterious representation of the tape format. The terms such as leader, sync byte, and so on should begin to mean something. The tone you hear is generated by *128-&H55's*. A *&H55* in binary is alternating ones and zeros which generates a tone to get the computer in sync with the tape.

Note the block type byte.

EXAMPLE:

55------55 55 3C 0 F I L E N A M E 2 0 0 2000 2400 2000 A2

128 - &H55's	Leader, the familiar tone at the beginning
Leader byte	1 Byte &H55 Make sure the tape is up to speed
Sync Byte	1 Byte &H3C Signals the start of a block
Block type	1 Byte 0=Title Block
File name	8 Bytes Padded with &H20 if needed
File Type	1 Byte: 0=Basic 1=Data 2=ML
Data type	1 Byte: 0=Binary &HFF=ASCII
Gap	1 Byte flag 0=Continuous flow &HFF=Gaps(Data)
MSB LSB	Starting address(ML)
MSB LSB	Load address(ML)
Check Sum	Number of bytes

55----55 55 3C 1 FF DATA..(0 to 255 BYTES)... FF

128-&H55's	Second leader tone
Data Block	
&H55	Leader Byte
&H3C	Sync Byte
Block Type	1=Data &HFF=EOF
Block Length	0to&HFF
Data	0 to 255 Bytes
Check sum	Sum of data+block type&block length

ADDITIONAL DATA BLOCKS DO NOT HAVE A LEADER TONE (128-&H55)

EOF BLOCK (End of File)

55 3C FF 00

Gap byte	1 Byte &H55
Sync byte	1 Byte &H3C
Block type	1 Byte &HFF (End of file)
Block Length	00

When you type *CLOAD*, BASIC recognizes it as a reserved word and goes to a table of locations to get the address for that routine in ROM. That ROM address is *&HA498*, which is partially disassembled in Listing 1. Don't forget, the disassembly is done by a beginner and lines are documented, which may be obvious to more experienced programmers. The routine checks to see if there is an *M* after *CLOAD* because there is no reserved *CLOADM* command. The *CLOADM* is actually a subroutine of the *CLOAD* command check, *&HA4FE*. Notice the subroutine at *&HA648* called "go search for file" shown in the disassembly "Listing 2." That subroutine is also called in the *CLOADM* subroutine.

The secret to fixing the *CLOAD* command is in the block type. The search for file routine (Listings 2 and 3) reads in a leader and block of data by calling *&HA701*. The *&HA701* subroutine reads a block and puts the block type in *&H7C*. The ORB checks to see if *&H7C* is a title block and/or if there are any errors. If it is not a title, then it does an RTS (return from subroutine) and prints an I/O Error. If it is a title block, it compares the title found to the title requested. If it doesn't match, it skips the rest of the file and looks for another title block.

If you start the tape in the middle of a file, of course, the first block read won't be a title block so the routine returns an I/O Error. If we start in the middle of a file, we don't really care if there is an I/O Error until after we find a title block. Also, we should ignore all blocks that aren't title blocks. Sound simple? Well, it is. Look at *&HA698*. If the program said to ignore I/O Errors and all blocks which are not titles and keep reading blocks until it finds a header, then it would be "fixed."

To fix the routine, you must first move the ROM to RAM with your "move ROM" program that boots 64K. The source code for the "fix" which modifies BASIC is in Listing 4. The program puts a jump in the "go search for file" routine at *&HA698* to point to the fix. At Line 180, the fix starts by storing the registers to make sure nothing is disturbed. The rest is fairly obvious. Lines 220-240 are the code which was replaced by the jump. The program is written in position independent code which means it can be put at any memory location and still run. I would suggest adding it on to your move ROM program so that when you boot the 64K the *CLOAD* is patched at the same time. For those of you without an editor assembler, boot 64K and type in the BASIC program in Listing 5. After you have checked for errors, *RUN* the program. Then save the machine language program by typing *CSAVEM "CLOADMOD", &HFD00, &HFD25, &HFD00*. As I said, you could append the program onto the move ROM program or run it separately.

The program runs without any problems except once in awhile I have gotten an I/O Error by starting in the middle of a file. I speculate that in the data there must be read a *&H55* and a *&H3C*. That triggers a read block which returns a check sum error. Should that ever occur, just type *CLOAD* again. I have never had it happen twice in a row. The fix is designed for 1.1 BASIC and I haven't checked to see if later versions are the same. If they are different, there should be enough discussion and listings so you can figure it out. I hope that by studying how to fix that pesky *CLOAD* I/O Error you have become interested in assembly language and maybe even learned some. I know I have.

Listing 1: "CLOAD routine"

```
A498    CLR 78      CLEAR FILE STATUS
A49A    CMPA #4A    IS THERE A"M" AFTER
                    "CLOAD"?
A49C    BEQ A4FE    IF SO GOTO CLOADM
                    ROUTINE
A49E    LEAS S+2    RESTORE STACK AND
                    RETURN
00200           BNE     REREAD  IF NOT TITLE
                        REREAD
00210           PULS    A,B,X,Y,CC  PUT
                        EVERYTHING BACK
00220           ORB     $7C     FILL IN BYTES
                        REPLACED W/ PATCH
00230           LBNE    $A6D0
00240           JMP     $A69C
00250   REREAD  PULS    A,B,X,Y,CC  REPLACE
                        STACK
00260           JMP     $A696   GOREAD
                        ANOTHER BLOCK
00270           END
```

```
A6C9    BRA A686    IF NOT RIGHT FILE START
                    OVER
A6CB    LDA #46     "F" FOR FOUND
A6CD    BSR A6F8    PUT "F" ON SCREEN
A6CF    CLRA
A6D0    RTS
```

Listing 5:

```
1       'LISTING 5
2       'MODIFY CLOAD COMMAND
5       READ A$,B$
6       'GET STARTING AND ENDING ADDRESS-
        TO CHANGE ADDRESS CHG THE TWO
        NUMBERS IN LINE 100
7       H$="&H"
8       A=VAL(H$+A$): B=VAL(H$+B$)
9       'CONVERT TO HEX
10      C=B-A+1
15      'DETERMINE NUMBER OF BYTES
20      FOR D=1 TO C
25      READ A1$ 'READ BYTES OF DATA
30      POKE A, VAL(H$+A1$)
33      'POKE PROGRAM INTO MEMORY
35      A=A+1: NEXT D
100     DATA FD00,FD24
102     DATA 8E,A6,98
104     DATA 86,7E
106     DATA A7,80
108     DATA 31,8D,00,04
110     DATA 10,AF,84
112     DATA 39
114     DATA 34,37
116     DATA 96,7C
118     DATA 26,0B
120     DATA 35,37
122     DATA DA,7C
124     DATA 10,26,6A,C6
126     DATA 7E,A6,9C
128     DATA 35,37
130     DATA 7E,A6,96
```

COMMLINK

Modeming Across America

Wayne Day

No matter how much RAM you have in your computer, or how much you spend on the latest in "super-techno" autodialing, coffee-making modems, you can't do a thing with them without some sort of terminal program which lets you communicate with the remote information system you're "talking" to, be it CompuServe, a bulletin board system (BBS) or another Coco.

Two functions *must* be accomplished by the terminal program:

1) Each time you press a key on your computer, the terminal program must convert the data generated into ASCII data and send that information out the serial RS-232 port to the modem; and
2) It must convert the data received from the modem (ASCII) into the appropriate signal that's used to generate a character on your screen.

Those are the absolutes — what we might call a "dumb" terminal because it can only do the very simple things.

A good example of a dumb terminal program is the Radio Shack *Videotex* program, which has been available in both tape and ROM cartridge versions. *Videotex* is usually the first communications program to be acquired by most CoCo users, since it is available in every Radio Shack store in the country.

Operation of *Videotex* is simple, merely a matter of plugging the cartridge in, turning the computer on, dialing up the BBS, and away you go. After your online session is over, though, and you've hung up the phone, about the only thing you can do with *Videotex* is review the last few pages of information that *Videotex* has received.

As a dumb terminal, *Videotex* doesn't have any built-in way to send any of the data you received to a printer and you can't store any of the information on disk or tape.

So what can a "smart" terminal program do for you? Let's go back to the very beginning of our telecommunications session, and see how a terminal program with "smarts" could help us out.

Since many of the modems being sold today have the capability of "picking up" the phone and dialing a telephone number, we could ask the terminal to remember our most frequently called numbers for us. Additionally, why would you want to type in the correct login sequence every time you call your favorite BBS or CompuServe? The sequence rarely changes, and it's a time waster for you, right? Let's combine those two features and call them "AUTODIAL and AUTOLOGON."

How does the terminal program remember the numbers and your logon sequence? Our "smart" terminal program lets you build a text file that contains all the information needed, and then recalls that information when you tell it to.

For example, let's assume you're calling CompuServe's Consumer Information Service (CIS), and want to build a file that works with your Hayes Smartmodem (a very programmable modem).

First, we've got to get the modem's attention with the "AT" command, followed by the instruction to dial a number, using touch-tones.

Then, when our terminal senses that a connection has been made, we'll tell it to send a CONTROL-C to get CIS's attention, and then to answer the User ID: and Password: prompts, then return control to us.

Thus, our command file might look something like this:

```
CIS             (what we call the file)
AT DT8702461    (Dial the number)
WAIT            (Wait for connect)
$03             (Send a CNTRL-C)
>ID:            (When CIS sends "ID:"...)
71234,5678      (send our User ID)
>word:          (When CIS sends
                 Pass(word:)...)
DARING#BIRD     (send our password)
/TERM           (return control to
                 operator)
```

The "smart" terminal program could read this file, and execute those commands just exactly as we had entered them ourselves, saving us some time, and more importantly, saving us from having to enter the same thing every day when we log on to CIS.

Other Features

While we're dreaming about what we'd like to put in our smart terminal program, let's think about what we do with all the information that scrolls off our screen.

In our dumb terminal, the data we got from the BBS was lost forever since there was no way to save it.

Aha! Let's make it possible to save anything we get in RAM, so we can look at it later. This feature of a smart terminal program is called a buffer, a temporary storage area.

But, wouldn't the data in the buffer be lost if we turned off the program, or powered-down the computer? Yep, it would, so we'll also include a method by which we can save all or a portion of the buffer to disk or tape, OK?

Receiving data from some other source, such as transferring a program from CompuServe to your own system, is known as "downloading." You can download to a printer, too. So that gives us "DOWNLOAD TO TAPE, DISK OR PRINTER."

What happens, though, if you're on a BBS for an hour, reading messages and looking through the available information, and all you really want to print out is one or two messages?

Ideally, our buffer should be able to be opened and closed two ways. The first method would be under manual control — you decide what you want to save, and what you don't want to save.

Additionally, there are times when the computer should know that you want to save what it's going to send in the next little bit, so there should also be "AUTOMATIC BUFFER CONTROL."

In the world of telecommunications, there have been some unofficial standards set, and one set of those standards says that whenever a terminal program "sees" a CONTROL-R (Hex value $12 — or CHR$(18)), it should OPEN the receive buffer. Conversely, when a CONTROL-T character (Hex $14 - CHR$ (20)) comes down the line, the terminal program should CLOSE the buffer.

If the BBS you are using supports the CONTROL-R / CONTROL-T method of buffer control, you wouldn't have to

open your buffer yourself when you want to download a program — just let the terminal program do it for you.

By the way, the CONTROL-R/ CONTROL-T characters are also known as DEVICE CONTROL-2 and DEVICE CONTROL-4, depending on whose list of control codes you're looking at. The important thing to remember, though, is that they are the same thing, no matter what they are called.

Do all BBS and information services use CONTROL-R / CONTROL-T? No, they don't, so in our "smart" terminal that we're working on, we would also like the ability to define just which particular character will be recognized as the "OPEN BUFFER" character, and which one will work as the "CLOSE BUFFER" control code.

So, in this case, we'll also include "DEFINABLE CONTROL CHARACTERS" in our list of desired features.

Let's Send It The Other Way

When you send pre-stored information to another computer, be it a BBS or a consumer-oriented information service, you "UPLOAD" the file, the opposite of "DOWNLOAD."

This can be extremely cost-effective if, for example, you are using a service where time is at a premium, or where you are charged by the minute of connect time.

Using your favorite word processor or a home-brewed message generator, you can compose messages before you connect your modem and have everything ready to go at the touch of a single key.

Where are you going to get the info? Again, it would be nice if you had the option of reading in a text file from either the cassette or disk, so we'll include "UPLOAD FROM TAPE OR DISK" in our list of things to have.

Are We Still Talking ASCII?

So far, we've assumed that all of our communications will be taking place using ASCII, those first 128 characters of the possible 255 that the CoCo can generate.

Is there anything besides ASCII?

Yes, and it's called binary.

Let's assume you have built a BASIC program that you want to save to disk.

Normally, you would enter:

```
SAVE "PROGRAM.BAS"
ENTER
```

But, if you entered:

```
SAVE "PROGRAM.BAS",A
ENTER
```

you would save the program on disk in ASCII format.

What's the difference, since both would load into the computer and run?

BASIC uses "tokens," a one-character or two-character symbol for certain words in the BASIC command library. Thus, instead of writing "RESTORE" on a disk, BASIC normally just writes a CHR$(143), saving six bytes on the disk. Follow that through with the whole program and you can see that tokenizing commands and keywords can save quite a bit of space in the long run.

What's that got to do with our "smart terminal program"? Look at the value of RESTORE. It's 143, above what is normally recognized in the ASCII "language."

An Apple computer, for example, wouldn't recognize that character as "RESTORE."

That's why ASCII was created, so all computers would have a common language that they could all recognize. And that's fine if we're only sending and receiving text or ASCII programs, but what happens when we want to receive a machine language program? ML programs need the whole range of values from 0 to 255, unlike an ASCII BASIC program.

We'll have to include non-ASCII uploading and downloading in our program then, and that will require the ability to send and receive eight data bits, since seven data bits are normally used on most BBSs and information services.

```
binary   1 1 1 1 1 1 1 = 128
         1 1 1 1 1 1 1 1 = 255
```

That means we need to be able to set "COMMUNICATIONS PARAMETERS." Normally, besides the data length of a "word," most full-featured terminal programs also allow you to specify the speed at which the data will be sent (300 and 1200 Baud are the most commonly used), the number of "stop bits" in a data word, as well as parity.

Parity is used to help insure a good transmission of data, and is used to verify that the proper data was sent.

Even parity means that the sum of all the bits in the "word" being sent will be equal to an even number. If the result of just the data is an odd number, an extra "1" value will be added to the word to bring the total value up to an even number.

For example, in a seven-bit word:

```
1 0 0 1 1 0 0
(1) (2) (3) (4) (5) (6) (7)
```

is an odd number. Since there are only three ones, even parity would make the eighth bit of the data word become another "1" and the result would be sent as:

```
1 0 0 1 1 0 0 1
(1) (2) (3) (4) (5) (6) (7) (8)
```

On the other hand, the first seven bits in the next word, the data itself, adds up to an even number, so the eighth bit, the parity bit, is set to a zero, changing nothing.

```
0 1 1 0 1 1 0 0
(1) (2) (3) (4) (5) (6) (7) (8)
```

Since it depends on which parity the host computer is expecting, our terminal program needs to be able to send either even or odd parity, ignore parity entirely, or always set the parity bit to a "1", or always set it to a "0".

Is Parity Foolproof?

Unfortunately, parity is not foolproof, and a noisy telephone line can do horrendous things to the 32K worth of BASIC program you just downloaded from your favorite BBS.

With that in mind, let's add an "ERROR-CHECKING and ERROR-CORRECTING PROTOCOL" to our terminal program, so we can be sure to get the most out of our online time.

A protocol is merely a set of rules: in this case, the rules by which an accurate transfer of data will take place from one system to another. Several such protocols exist for the Color Computer today.

In the general world of microcomputers, perhaps the most popular error-checking protocol is known as "XMODEM."

XMODEM was created in 1982 by Ward Christiansen, founder of the first BBS system (Ward and Randy's CBBS, Chicago, 1977 — see the list of BBSs in this issue of THE RAINBOW) and was originally written for the CP/M operating system.

It works like this:

The sending computer loads in the file, be it a BASIC program, a machine language program, or a text file (it doesn't matter to XMODEM), and looks at the first 128 bytes of the file.

It adds up all the values in the first 128 bytes, and remembers that number, called a checksum, just like *Rainbow Check Plus* used here in THE RAINBOW to make sure you typed the correct

information into your computer

When the receiving computer is ready, it sends a signal to the sender which starts throwing the data out, one byte at a time. Following the last byte of data, the sender adds the checksum it computed earlier.

The receiving computer, while all this is going on, is also keeping track of what it has received, and computes its own version of the checksum.

If the two checksums agree, the receiver signals the sender that all is well, and to continue.

If the checksums are not equal, though, the entire block of 128 bytes of data is re-sent, and the process is repeated.

This way, you're sure that what you sent is what the other end received, and vice versa — error-checking and error-correcting.

The popularity of XMODEM comes into play when you consider that it is the standard file transfer method on the majority of BBS systems that offer any sort of error-checking and error-correcting protocol.

For the TRS-80, the popular TBBS Bulletin Board program supports XMODEM, as does a recently announced BBS program for the CoCo, *COBBS*.

Is XMODEM Standard?

There are many other error-checking protocols in use, and unfortunately, most of them are not compatible with each other.

DFT (Direct File Transfer) for the TRS-80 series of computers (Model I, Model III/4 and CoCo), for example, uses a 256-byte block of data, and a different series of commands between the sender and receiver.

For CompuServe users, CIS offers not one, but two error-checking protocols of its own design, the CompuServe "A" protocol, and the CIS "B" protocol that's used in CIS's CoCo *VIDTEX* (not *Videotex*, which is sold by Radio Shack) terminal program.

Each of the protocols has its advantages, and each has its own disadvantage, the biggest of which may be that not enough systems support that particular protocol.

What it all boils down to is that the particular protocol you will need will depend upon which protocol is in use by the host computer you call.

CompuServe's CIS, for example, has recently begun to support XMODEM in addition to its own protocols, due to the large number of terminal programs for all computers that support XMODEM.

In our "smart" terminal program wish list, then, let's assume we'll add the XMODEM protocol to the program, because of the popularity of the protocol on many BBS systems around the country.

However, if we also wanted to exchange programs with another Color Computer user, we could use almost any of the protocols available. So again, let me emphasize that the particular protocol you "need" will be dependent on what you're going to do with the terminal program. In this case, it's best to investigate all of the possibilities.

Is That About It?

We could also add the ability to send some pre-programmed but standard sentences, display the characters on a high resolution 51x24 or 64x24 screen, instead of the 32x16 screen normally seen on the Coco, and a few of the other "bells and whistles" that make each individual terminal program different, but I think you might have a good idea of what's really needed.

And so, the bottom line: Is there any terminal program available that does everything that we could possibly want it to do?

I haven't been able to find one for the CoCo, nor for any other computer, for that matter. And, if *you* find one, I wish you'd let me know.

Each of the terminal programs available for the CoCo today has, in my personal opinion, its own pluses and minuses.

There *are* programs available that do a large majority of the things on our wish list, the major thing lacking in most terminal programs being protocol uploading and downloading.

The idea here is that you should carefully read the advertisements, write for literature, and investigate your purchase before you commit yourself to just one terminal program.

Or, you can work another strategy, and do as I do, and use several different terminal programs, each one working well for a particular application.

For example, when I'm performing my SYSOP duties on The Color SIG (Special Interest Group) on CompuServe, I may be using one of the programs that works well at 1200 Baud (not all of them do), so I can rapidly read and reply to messages, work on my system files, and maintain the SIG's database.

When I want to upload or download an ASCII text file, I may choose a different program, one that only runs at 300 Baud, but is easy to use to upload and download files. And, when I'm working with a binary file, like a machine language program or a graphics screen, I probably will use a third terminal program.

So, as you can see, the terminal program, or programs, you choose are vitally important, and you should make your purchase decisions wisely.

... From P 40

that even without the modification there should be no problem writing at addresses that would be unoccupied by ROM in map type zero, or even at addresses that would contain ROM as long as the data to be written is copied directly from the ROM, avoiding the bus conflict. This means that even in the older, unmodified 32K units, about 16K of free memory in the range of *$C000-$FEFF* may be easily accessed if a disk controller or ROM cartridge is not in the system by simply running the above program. After running this program, the system is actually in the 64K mode, although without the modification the 16K from *$8000-$BFFF* is unable to reliably store anything other than an exact copy of the ROM, so it's really more of a 48K computer. I have seen several articles regarding the process of relocating ROM packs to run in low RAM, but this seems hardly worth the effort when you can load the programs in RAM from tape and run them at the address they were designed for without breaking your warranty seal to effect a true 64K computer. There is also another use for this mode; if even Extended BASIC is not in the system, it can be loaded from tape and *EXECed*. To make the tape:

```
CSAVEM "XBASIC", &H8000,
&H9FFF, &H8002
```

Remember that pressing Reset sets the SAM back to map type zero, the ROM mode, and makes the RAM above 32K disappear; also, the high speed (address dependent) mode will not work in map type one.

'SOPWITH CoCo' FLIES AGAIN

William G. Franklin

For all you Sopwith flyers who have stuck with it, we are going to clean up a little and then really get serious about navigational aids.

If you had trouble in line 170 of the INSTRUMENT FLIGHT SIMULATOR in the June 1983 issue, check the data statements carefully in the 7000 series lines to see that you have not inserted a semicolon in place of a comma. These are DRAW strings and the letters must be correct or you will get a Function Call Error when the program attempts to draw the character in error. Disk system users please note thar the variable AS in lines 390,400,405,410,1190,1510, and 8030 should be changed to some other designation, i.e., AV or VQ. Finally, if you remove the semicolons altogether in the 7000 series data statements you should be able to run the 16K version without having to CLEAR 100 prior to running.

This update will allow 26 stations to be input; install a course deviation indicator (CDI) feature coupling the path deviation indicator to TACAN and ADF stations; require a field elevation input when entering ILS location data (this means that you will no longer landing at 0 on the altimeter, but the barometric altitude, or field elevation, you set in) It will also give you some instrument labels if you wish, and clean up some of techniques.

The increase to 26 stations now uses the alphabet to identify the stations: A through F for ILS, G through P for TACAN, and Q through Z for ADF. A significant difference is that when entering station data, the letter identifying the station need only be pressed. The rest of the information must still be entered. However, if you make a mistake, the information still may be re-entered the next time as only the last data entered for a

particular station is valid. You will also be entering a field elevation - more on this later.

The CDI feature includes a read out in the window just above the DME window, a dot indication on the outside of the D/F dial, and an indication by the path deviation pointer if you are within +/-15 degrees of the selected course.

Course selection is made with the '1','2' and '3' keys. The '1' key increments the hundreds digit, the '2' key increments the tens digit and the '3' key increments the ones digit. Values above 360 degrees are not allowed For example, if the current selected course is 276 degrees and the hundreds digit is incremented, the tens digit will reset to 0. This should not present any difficulty so long as the operation is understood. As a new course is selected, the dot on the outside of the D/F dial will be placed accordingly , on the course selected.

If you are 'inbound' to station and want to fly a particular radial, the course selected must be the reciprocal, 180 degrees opposite, of the radial you wish to fly . Example 1: You are flying inbound on the 44 degree radial, the selected course should be 224 degrees. Example 2. You are inbound on the 286 degree radial, the course selected should be 106 degrees. If you are outbound, the selected course should be that of the radial. Example: You are flying away from a station on the 318 degree radial, select course 318. In all of these cases the path deviation pointer will indicate where the correct path is and you must fly toward that path - operation of this indicator is the same as in the ILS mode. As a course deviation indicator the calibration is 5 degrees per division up to +/- 15 degrees. Greater than 15 degree deviation from the selected course disconnects the CDI and removes the pointer.

The CDI indicates the selected path, left or right of your airplane, regardless of the airplan heading. Therefore it is important that you understand the inbound / outbound and course select relationship so that you can properly interpret the CDI movement when moving acroos the radial and using that station to cross check your position.

When an ILS station, for which you have input a position, is selected, the runway heading will be displayed in the course set window and the indicator on the D/F dial will be set accordingly. Course select can not be changed while an ILS station is selected.

The field elevation feature allows you to insert a 0 to 2000 foot field elevation and, when landing, the ILS glidescope will indicate properly so that you will touch down with an altimeter reading of the field elavtion. If you are not using an ILS, the terrain elavation reverts to 0. IMPORTANT: If you enter an ILS airspace at below the field elavation, then punch up the ILS station, the program will pause with a reminder of your error. you may continue by pressing ENTER. The ILS station will be turned off and you should climb above the field elavation before reselecting that station. The "aircraft elavation" input allows you to place the airplane "on the field" at the beginning. However , as soon as you are airborne and select a valid nav-aids station, terrain elavation reverts to 0 or to that input for an ILS if you are in range.

The 3 features just described should enable you to use the FAA low-level charts and let-down plates to realistically simulate instrument flying techniques. These charts are available at FAA flight stations. Correspondence I have received from pilots have been enthusiastic about the Sopwith CoCo and these improvements are the ones most frequently requested.

Finally, the change in step 5670 puts the turn rate indicators close to the 4 minute and 2 minute turn rates, and line 5810 will label

the top two instruments on each side. More efficient methods are used in lines 50,60 and 1835; some interesting input techniques in lines 56 and 1610-1625; and the course set control in lines 1660-1700.

If you do not wish to key in the program yourself, you may send $7 to William G. Franklin, 31 Preston Ct. Jacksonville, AR 72076 for a tape of the completed updated Sopwith CoCo. Likewise, if anyone wants the original version for 16K, the same offer applies and please specify you want the 16K version.

START OF THE LISTING:

```
 1 PRINT "COPYRIGHT WILLIAM G. FRANKLIN 1983"
 2 'SIM 16A1    JUL 1983
 20 DIMN$(26):DIMN(26):DIMSX(26):
DIMSY(26)
30 FOR X=0 TO 26:READ N$(X):N(X)
=0:NEXT
40 GOTO 55
50 CLS:PRINT"YOU HAVE ENTERED CO
ORDINATES FORTHE FOLLOWING STATI
ONS:"
53 FOR S=1 TO 26:IF N(S)=1 THEN
PRINTCHR$(S+64);CHR$(32);
54 NEXT
55 PRINT:PRINT"WHAT STATION NUMB
ER (A-Z)? ";
56 S$=INKEY$:IF S$=""THEN 56 ELS
E S=ASC(S$)-64:IF S=-16 THEN 110
58 IF S<1 OR S>26 THEN 56
59 PRINTS$
60 IF S>16 THEN 90 ELSE IF S>6 T
HEN 80
70 PRINT:PRINT"STATION ";CHR$(S+
64);" IS AN ILS":INPUT"STATION B
EARING (0-360)";A:A=A/57.29:INPU
T"STATION DISTANCE (MILES)";D:D=
D*5280:SX(S)=D*COS(A):SY(S)=D*SI
N(A):INPUT"RUNWAY HEADING (0-360
)";X:RB(S)=INT(X)/57.29:N(S)=1
72 INPUT"FIELD ELEV. (0-2000 FEE
T)";X:IF X<0 OR X>2000 THEN 72 E
LSE FE(S)=X
74 GOTO 50
80 PRINT:PRINT"STATION ";CHR$(S+
64);" IS A TACAN":GOTO 100
90 PRINT:PRINT"STATION ";CHR$(S+
64);" IS AN ADF"
100 INPUT"STATION BEARING (0-360
)";A:A=A/57.29:INPUT"STATION DIS
TANCE (MILES)";D:D=D*5280:SX(S)=
D*COS(A):SY(S)=D*SIN(A):N(S)=1:G
OTO 50
110 CLS:PRINT:INPUT"AIRCRAFT BEA
RING (0-360)";A:A=A/57.29:INPUT"
AIRCRAFT DISTANCE (MILES)";D:D=D
*5280:TX=D*COS(A):TY=D*SIN(A):IN
PUT"AIRCRAFT HEADING (0-360)";X:
CS=INT(X)
112 INPUT"AIRCRAFT ELEV. (0-2000
FT)";X:IF X<0 OR X>2000 THEN 112
 ELSE AL=X
120 PRINT:INPUT"GALLONS OF FUEL
(MAX 24,USES 6  GPH @ 2000 RPM)"
;X:IF X>24 THEN FR=12 ELSE IF X<
0 THEN FR=-12 ELSE FR=X-12
130 PRINT:INPUT"WIND DIRECTION (
0-360)";X:WA=INT(X)+180:INPUT"WI
ND VELOCITY (MPH)";X:WS=INT(X)
135 CLS:PRINT:PRINT
140 Z=JOYSTK(0):X=JOYSTK(1):IF X
<63 THEN PRINT@64,"PLEASE PULL T
HROTTLE BACK":GOTO 140
145 CLS
150 PRINT:PRINT:PRINT"PLEASE STA
ND BY. AIRCRAFT IS    BEING SERV
ICED."
170 S=0:DIMA$(10):FOR X=0 TO 10:
READ A$(X):NEXT X
210 PMODE4,1:PCLS0:GOSUB5000
290 RX=30:RY=100:AX=30:AY=40:VX=
224:VY=100:L1=224:L2=40:LX=224:L
Y=40:FX=224:FY=180:SX=30:SY=160:
IX=128:IY=40
300 Q7=1:Q6=1:Q5=1:Q8=.01:H=0:AZ
=AL:C7=9:C6=9:C5=9:LR=40:P1=1:TI
MER=0
320 H=TIMER:TIMER=0:TH=TH+H
330 Z=JOYSTK(0):R=63-JOYSTK(1):B
=JOYSTK(2)-31:EP=(JOYSTK(3)-31)/
188
340 IF B<>B1 OR EP<>PE THEN GOSU
B 8000
360 IF AL>AZ OR R>12 THEN R=1260
+R*20 ELSE R=R*116
365 IF FR=-12 AND AL>AZ THEN R=7
00 ELSE IF FR=-12 AND AL=AZ THEN
 R=0
367 IF R1<30 THEN R1=0
370 R=R1+((R-R1)*.6):R1=R:X=30+(
SIN(R/636.6)*15):Y=100-(COS(R/63
6.6)*15):LINE(30,100)-(RX,RY),PR
ESET:LINE(30,100)-(X,Y),PSET:RX=
X:RY=Y
380 AP=(SIN(6*((R-2000)/4488)+.5
7)-.54)/6
390 AS=(R*(1-AP)*(1-EP))/16.66:A
S=A1+((AS-A1)*.1):IF AS<10 THEN
AS=0
400 X=30+(SIN(AS/39.46)*15):Y=40
-(COS(AS/39.46)*15):LINE(30,40)-
(AX,AY),PRESET:LINE(30,40)-(X,Y)
,PSET:AX=X:AY=Y:A1=AS
405 IF AS<50 THEN EP=-1
410 IF SGN(VV)=-1 THEN VV=((AP+E
P)*AS)/15 ELSE VV=(1-AL/10000)*(
(AP+EP)*AS)/15
420 IF AL=AZ AND SGN(VV)=-1 THEN
 VV=0
```

```
430 V1=V5+((VV-V5)*.3):X=224-(CO
S(V1)*15):Y=100-(SIN(V1)*15):LIN
E(224,100)-(VX,VY),PRESET:LINE(2
24,100)-(X,Y),PSET:VX=X:VY=Y:V5=
V1
440 Z=JOYSTK(0):B=JOYSTK(2)-31:E
P=(JOYSTK(3)-31)/188:P=INT(100*(
EP+(SIN(6*AP-.57)+.54)/6)):IF B<
>B1 OR EP<>PE OR P<>P1 OR AS<40
THEN GOSUB 8000
450 AL=AL+(VV*H)/4.6:IF AL<=0 TH
EN AL=0 ELSE IF AL<=AZ THEN AL=A
Z
460 X=224+(SIN(AL/159)*15):Y=40-
(COS(AL/159)*15):LINE(224,40)-(L
X,LY),PRESET:LINE(224,40)-(X,Y),
PSET:LX=X:LY=Y
470 X=224+(SIN(AL/1592)*8):Y=40-
(COS(AL/1592)*8):LINE(224,40)-(L
1,L2),PRESET:LINE(224,40)-(X,Y),
PSET:L1=X:L2=Y
560 FR=FR-((R*H)/72000000):IF FR
=<-12 THEN FR=-12
570 X=224+SIN(FR/25.2)*26:Y=180-
COS(FR/25.2)*26:LINE(224,180)-(F
X,FY),PRESET:LINE(224,180)-(X,Y)
,PSET:FX=X:FY=Y
580 Z=JOYSTK(0):B=JOYSTK(2)-31:E
P=(JOYSTK(3)-31)/188:IF B<>B1 OR
 EP<>PE THEN GOSUB 8000
1190 IF AS<=0 THEN 1310
1200 BC=-10*(COS((B/31)+1.570796
3)):CS=CS+BC:IF CS>360 THEN CS=C
S-360 ELSE IF CS<=0 THEN CS=CS+3
60
1310 C=INT(CS+.5):IF C=C1 THEN 1
350 ELSE C2=INT(C/100):C3=INT(C/
10)-(10*C2):C4=C-(100*C2)-(10*C3
):C1=C:DRAW"C0;BM134,23;XA$(C7);
C1;XA$(C4);":C7=C4
1320 IF C3=C6 THEN 1350 ELSE DRA
W"C0;BM126,23;XA$(C6);C1;XA$(C3)
;":C6=C3
1330 IF C2=C5 THEN 1350 ELSE DRA
W"C0;BM118,23;XA$(C5);C1;XA$(C2)
;":C5=C2
1350 X=30+SIN(C/57.2)*10:Y=160-C
OS(C/57.2)*10:CIRCLE(CX,CY),1,0:
CIRCLE(X,Y),1,1:CX=X:CY=Y
1360 Z=JOYSTK(0):B=JOYSTK(2)-31:
EP=(JOYSTK(3)-31)/188:IF B<>B1 O
R EP<>PE THEN GOSUB 8000
1510 IF AS=0 THEN 1610 ELSE CD=(
AS*5.28*H)/216:A=CS/57.29:X=CD*C
OS(A):Y=CD*SIN(A):TX=TX+X:TY=TY+
Y
1520 IF AL=AZ OR WS=0 THEN 1610
ELSE WD=(WS*5.28*H)/216:A=WA/57.
29:X=WD*COS(A):Y=WD*SIN(A):TX=TX
+X:TY=TY+Y
1610 S$=INKEY$:IF S$="" THEN 164
0
1615 X=VAL(S$):IF X>0 AND X<4 AN
D S>6 OR S<1 THEN ON X GOSUB 166
0,1670,1680
1620 X=ASC(S$)-64:IF X=-16 THEN
X=0
1625 IF X<0 OR X>26 THEN 1640 EL
SE GOSUB 1710
1630 DRAW"C0;BM126,170;XN$(S);C1
;XN$(X);":S=X:GOSUB 1710
1635 IF S<7 AND N(S)=1 GOSUB 174
0
1640 IF S=0 OR N(S)=0 THEN GOSUB
 1710 ELSE GOSUB 1810
1645 Z=JOYSTK(0):B=JOYSTK(2)-31:
EP=(JOYSTK(3)-31)/188:IF B<>B1 O
R EP<>PE THEN GOSUB 8000
1650 GOTO 2100
1660 F=F+1:IF F>3 THEN F=0
1665 DRAW"C0;BM66,151;XA$(FS);C1
;XA$(F);":FS=F:GOTO 1690
1670 G=G+1:IF G>9 THEN G=0
1675 DRAW"C0;BM73,151;XA$(GS);C1
;XA$(G);":GS=G:GOTO 1690
1680 I=I+1:IF I>9 THEN I=0
1685 DRAW"C0;BM80,151;XA$(IS);C1
;XA$(I);":IS=I
1690 J=(F*100)+(G*10)+I:IF J>360
 THEN G=-1:GOTO 1670
1695 JB=J/57.2
1700 X=30+SIN(JB)*28:Y=160-COS(J
B)*28:CIRCLE(FA,FB),1,0:CIRCLE(X
,Y),1,1:FA=X:FB=Y:RETURN
1710 IF D7=10 AND N(S)=0 THEN RE
TURN ELSE LINE(30,160)-(SX,SY),P
RESET:DRAW"C0;BM83,170;XA$(D7);B
M-10,0;XA$(D6);BM-7,0;XA$(D5);C1
;XA$(10);BM+7,0;XA$(10);BM+10,0;
XA$(10);":LINE(128,40)-(IX,IY),P
RESET:CIRCLE(162,92+GX),1,0,.1:D
7=10:D6=10:D5=10
1712 IF AZ<AL THEN AZ=0
1730 SCREEN1,0:RETURN
1740 F=INT(RB(S)*.5729):G=INT(RB
(S)*5.729)-(10*F):I=INT(RB(S)*57
.29)-(100*F)-(10*G):DRAW"C0;BM66
,151;XA$(FS);BM+7,0;XA$(GS);BM+7
,0;XA$(IS);C1;XA$(I);BM-7,0;XA$(
G);BM-7,0;XA$(F);":FS=F:GS=G:IS=
I:JB=RB(S):GOTO 1700
1810 BX=SX(S)-TX:BY=SY(S)-TY:SD=
SQR(BX^2+BY^2)
1811 IF BX=0 THEN BX=.001
1812 IF BY=0 THEN BY=.001
1820 X=BY/BX:IF BX<0 THEN 1830 E
LSE SB=ATN(X):GOTO 1835
1830 SB=ATN(X)-3.1416
1835 IF S>16 THEN 2039 ELSE IF S
>6 THEN 1939
```

```
LSE IF DB<-.0523 THEN DB=-.0523
1850 X=128-SIN(DB*20)*15:Y=40+CO
S(DB*20)*15:LINE(128,40)-(IX,IY)
,PRESET:LINE(128,40)-(X,Y),PSET:
IX=X:IY=Y
1855 AZ=FE(S):AR=AL-AZ:IF AR<0 T
HEN 3000
1860 X=((AR/(SD-1000))-.04366)*1
000:IF X>18 THEN X=18 ELSE IF X<
-18 THEN X=-18
1870 CIRCLE(162,92+GX),1,0,.1:CI
RCLE(162,92+X),1,1,.1:GX=X:GOTO
1940
1939 IF SD/528>999 THEN 1710
1940 X=30+SIN(SB)*15:Y=160-COS(S
B)*15:LINE(30,160)-(SX,SY),PRESE
T:LINE(30,160)-(X,Y),PSET:SX=X:S
Y=Y:IF S>6 GOSUB 2050
1950 SD=INT(SD/528):IF SD=D1 THE
N RETURN ELSE D2=INT(SD/100):D3=
INT(SD/10)-(10*D2):D4=SD-(100*D2
)-(10*D3):D1=SD:DRAW"C0;BM83,170
;XA$(D7);C1;XA$(D4);":D7=D4
1960 IF D3=D6 THEN RETURN ELSE D
RAW"C0;BM-10,0;XA$(D6);C1;XA$(D3
);":D6=D3
1970 IF D2=D5 THEN RETURN ELSE D
RAW"C0;BM-7,0;XA$(D5);C1;XA$(D2)
;":D5=D2:RETURN
2039 IF SD/528>1999 THEN 1710
2040 X=30+SIN(SB)*15:Y=160-COS(S
B)*15:LINE(30,160)-(SX,SY),PRESE
T:LINE(30,160)-(X,Y),PSET:SX=X:S
Y=Y
2050 CA=SB-(JB-1.5708):DB=COS(CA
):X=128-SIN(DB*4)*15:Y=40+COS(DB
*4)*15:LINE(128,40)-(IX,IY),PRES
ET:IF ABS(DB)>.2588 THEN RETURN
ELSE LINE(128,40)-(X,Y),PSET:IX=
X:IY=Y:RETURN
2100 Q1=INT(TH/3600):IF Q1=Q8 TH
EN 320 ELSE Q2=INT(TH/216000):Q3
=INT(TH/36000)-(6*Q2):Q4=Q1-(10*
Q3)-(60*Q2):Q8=Q1:DRAW"C0;BM184,
170;XA$(Q7);C1;XA$(Q4);":Q7=Q4
2130 IF Q3=Q6 THEN 320 ELSE DRAW
"C0;BM-7,0;XA$(Q6);C1;XA$(Q3);":
Q6=Q3
2140 IF Q2=Q5 THEN 320 ELSE DRAW
"C0;BM-10,0;XA$(Q5);C1;XA$(Q2);"
:Q5=Q2:GOTO 320
3000 CLS:PRINT:PRINT"YOUR ALTITU
DE IS LESS THAN       TERRAIN ELE
VATION.  YOU MAY      CONTINUE BY
 PRESSING <ENTER>.    THE ILS STA
TION SELECTED WILL    BE TURNED O
FF AND YOU SHOULD BE ABOVE";FE(S
);"FT. BEFORE SELECTING"
3005 PRINT"STATION '";CHR$(S+64)
;"' AGAIN."
3010 INPUT"";X:SCREEN1,0
3020 DRAW"C0;BM126,170;XN$(S);C1
;XN$(0);":S=0:D7=0:AZ=0:TIMER=0:
GOTO 1710
5000 FOR X=30 TO 224 STEP 194:FO
R Y=40 TO 160 STEP 60:CIRCLE(X,Y
),25,1:NEXT Y:NEXT X
5010 CIRCLE(128,92),55,1
5020 LINE(84,92)-(88,92),PSET:FO
R X=74 TO 110 STEP 6:LINE(86,X)-
(88,X),PSET:NEXT X
5030 LINE(168,92)-(172,92),PSET:
FOR X=74 TO 110 STEP 9:LINE(168,
X)-(170,X),PSET:NEXT X
5040 FOR Z=1 TO 3:GOSUB 5060:NEX
T Z
5050 FOR Z=5 TO 7:GOSUB 5060:NEX
T Z:GOTO 5100
5060 X=SIN((6.2832/10)*Z):Y=COS(
(6.2832/10)*Z)
5070 XA=30+(20*X):XB=30+(24*X):Y
A=40-(20*Y):YB=40-(24*Y)
5080 LINE(XA,YA)-(XB,YB),PSET:RE
TURN
5100 DRAW"BM28,24;XA$(0);BM39,58
;XA$(1);BM10,36;XA$(2);"
5200 FOR Z=1 TO 9 STEP 2
5210 X=SIN((6.2832/10)*Z):Y=COS(
(6.2832/10)*Z)
5220 XA=224+(20*X):XB=224+(24*X)
:YA=40-(20*Y):YB=40-(24*Y)
5230 LINE(XA,YA)-(XB,YB),PSET:NE
XT Z
5250 DRAW"BM222,24;XA$(0);BM239,
36;XA$(2);BM233,58;XA$(4);BM210,
58;XA$(6);BM204,36;XA$(8);"
5300 FOR Z=2 TO 8 STEP 2:X=COS((
6.2832/10)*Z):Y=SIN((6.2832/10)*
Z)
5320 XA=224+(20*X):XB=224+(24*X)
:YA=100+(20*Y):YB=100+(24*Y)
5330 LINE(XA,YA)-(XB,YB),PSET:NE
XT Z
5340 DRAW"BM237,113;XA$(2);BM215
,121;XA$(1);BM203,102;XA$(0);BM2
15,85;XA$(1);BM237,92;XA$(2);"
5350 LINE(114,15)-(142,25),PSET,
B:CIRCLE(128,19),18,1
5400 FOR Z=1 TO 5 STEP 2
5410 X=SIN((6.2832/8)*Z):Y=COS((
6.2832/8)*Z)
5420 XA=30+(20*X):XB=30+(24*X):Y
A=100-(20*Y):YB=100-(24*Y)
5430 LINE(XA,YA)-(XB,YB),PSET:NE
XT Z
5450 DRAW"BM28,84;XA$(0);BM47,10
```

```
3;XA$(1);BM28,122;XA$(2);BM9,103
;XA$(3);"
5500 FOR Z=2 TO 11 STEP 3:GOSUB
5510:NEXT
5505 FOR Z=1 TO 10 STEP 3:GOSUB
5510:NEXT:GOTO 5540
5510 X=SIN((6.2832/12)*Z):Y=COS(
(6.2832/12)*Z)
5520 XA=30+(20*X):XB=30+(24*X):Y
A=160-(20*Y):YB=160-(24*Y)
5530 LINE(XA,YA)-(XB,YB),PSET:RE
TURN
5540 DRAW"BM28,144;XN$(14);BM47,
163;XN$(5);BM28,182;XN$(19);BM9,
163;XN$(23);"
5600 FOR Z=-1 TO 1
5610 X=SIN((6.2862/24)*Z):Y=COS(
(6.2862/24)*Z)
5620 XA=224+(30*X):XB=224+(33*X)
:YA=180-(30*Y):YB=180-(33*Y)
5630 LINE(XA,YA)-(XB,YB),PSET:NE
XT
5650 DRAW"BM206,154;XN$(5);BM238
,154;XN$(6);"
5660 CIRCLE(128,0),140,1,1,.23,.
29
5670 DRAW"BM128,135;ND5;BM+11,0;
ND3;BM+9,0;ND2;BM-31,0;ND3;BM-9,
0;D2;"
5680 LINE(62,142)-(88,153),PSET,
B:LINE(62,161)-(91,172),PSET,B:L
INE(163,161)-(192,172),PSET,B:LI
NE(122,161)-(134,172),PSET,B:DRA
W"BM80,169;U1;BM+94,0;N;D1;BU2;U
1;"
5705 FOR Z=-3 TO 3:A=Z/57.29:X=S
IN(A*20):Y=COS(A*20):XA=128-(19*
X):XB=128-(21*X):YA=40+(19*Y):YB
=40+(21*Y)
5710 LINE(XA,YA)-(XB,YB),PSET:NE
XT
5800 DRAW"BM126,170;XN$(0);BM66,
151;XA$(0);BM+7,0;XA$(0);BM+7,0;
XA$(0);"
5810 DRAW"BM50,65;XN$(13);BM+7,0
;XN$(16);BM+7,0;XN$(8);BM+0,+60;
XN$(13);BM-7,0;XN$(16);BM-7,0;XN
$(18);BM+151,0;XN$(9);BM-7,0;XN$
(22);BM-7,0;XN$(22);BM+0,-60;XN$
(1);BM+7,0;XN$(12);BM+7,0;XN$(20
);"
5900 RETURN
6000 DATA BU1U4E1R2F1NG4D4G1L2H1
BD1,U5E1R2F1D2NL4D3BL4,U6R3F1D1G
1NL3F1D1G1L3,BU1U4E1R2F1BD4G1L2H
1BD1,U6R2F2D2G2L2,U6R4BD3BL1L3D3
NR4
6010 DATA U6R4BD3BL1L3D3,BU1U4E1
R3BD4NL1D2L3H1BD1,U6BR4D3NL4D3BL
4,BU6BR1R2L1D6NR1L1BL1,BU6BR4D5G
1L2H1BD1,U6D3R1NE3NF3BD3BL1
6020 DATA NU6NR4,U6F2ND1E2D6BL4,
U6D1F4NU5D1BL4,BU1U4E1R2F1D4G1L2
H1BD1,U6R3F1D1G1L3D3,BU1U4E1R2F1
D3G1NH1NF1G1L1H1BD1
6030 DATA U6R3F1D1G1L3R1F3BL4,BU
5BR4H1L2G1D1F1R2F1D1G1L2H1BD1,BU
6R4L2D6BL2,BU1U5BR4D5G1L2H1BD1,B
U6BR4D2G1D1G1ND1H1U1H1NU2BD4,NU6
E2NU1F2NU6BL4
6040 DATA BU6D1F2E2NU1BD4ND1H2G2
D1,BU6BR4D1G2ND3H2NU1BD5,BU6R4D1
G4D1NR4
7000 DATA BU1U4E1R2F1NG4D4G1L2H1
BD1,BU6BR2NG1D6R1L2BL1,BU5E1R2F1
D1G1L1G2D1NR4,BU5E1R2F1D1G1NL1F1
D1G1L2H1BD1,BU2NR4U1E3D6BL3
7010 DATA BU6NR4D3E1R2F1D2G1L2H1
BD1,BU6BR2NR2G2D3U2R3F1D1G1L2H1B
D1,BU6R4D1G3D2BL1,BU1U1E1NR1H1U1
E1R2F1D1G1NL1F1D1G1L2H1BD1,BU4NF
1U1E1R2F1D2NL3D1G2L2
7020 DATA BU6D1F2E2NU1BD4ND1H2G2
D1
8000 TR=INT(B*COS(B/41)+.5):IF T
R=LR THEN 8020
8010 LINE(126+LR,130)-(130+LR,13
0),PRESET:LINE(128+LR,130)-(128+
LR,133),PRESET:LINE(126+TR,130)-
(130+TR,130),PSET:LINE(128+TR,13
0)-(128+TR,133),PSET:LR=TR
8020 P=INT(100*(EP+(SIN(6*AP-.57
)+.54)/6)):IF P>20 THEN P=20 ELS
E IF P<-20 THEN P=-20
8025 IF AL<5+AZ THEN AL=AZ
8030 IF AL=AZ AND AS<50 OR AL=AZ
 AND SGN(P)=-1 THEN P=0
8040 IF P=P1 AND B=B1 THEN RETUR
N
8050 BB=B/93:IF AL=AZ THEN BB=0
8060 X=INT(COS(BB)*25):Y=SIN(BB)
*25
8070 LINE(128+PX,92-P1+PY)-(128-
PX,92-P1-PY),PRESET:CIRCLE(128,9
2-P1),5,0
8080 LINE(128+X,92-P+Y)-(128-X,9
2-P-Y),PSET:CIRCLE(128,92-P),5,1
:P1=P:PX=X:PY=Y:B1=B:PE=EP:RETUR
N
```

RAINBOW WARE

This range of clothing is made here on the Gold Coast.

DESCRIPTION	SIZES	PRICE
V-Neck Crew Neck	12. 14. S. M. L. XL	14.95
Tank Tops	10. 12. 14. S. M. L. XL	13.85
Tennis Dress	S. M. L.	26.20
Hooded Zip-up Jacket	S. M. L.	31.95
Ladies Hooded Pullover Top	S. M. L.	28.50
Ladies Shorts	S. M. L.	17.95
Children's Tank Tops	6. 8. 10. 12. 14	12.80
Children's T-Shirts	8. 10. 12. 14	14.20

COLOURS AVAILABLE white, pinik, aqua, sky blue, yellow, hot pink, black

PLEASE SEND ORDER FORM

_____ $ _____

_____ $ _____

_____ $ _____

_____ $ _____ AMOUNT _____

Mr / Mrs / Miss / Initials / Surname — Complete the section below with one letter, figure or space per square.

Subscription No

Address

PC

☐ CASH
☐ CHEQUE
☐ POSTAL ORDER

☐ VISA ☐ BANKCARD

_____ Cardholder

UNDERSTAND THAT I CAN RETURN THESE GOODS WITHIN 7 DAYS FOR A FULL REFUND FOR ANY REASON

Now is the time to subscribe to Australian Rainbow/GoCo

Copies of back issues can be obtained, subject to the availability of stocks, by using this order form and marking clearly which issues you require to be sent to you. Each issue costs $3.25 including postage and packing. Please enclose your cheque/postal order made payable to: Graham Morphett, PO Box 1742, Southport, 4215.

Send off the slip below to ensure that you get on the mailing list for JANUARY

CUT OUT AND MAIL or PHONE BANKCARD NO. (075 51 0015)

Subscription Rates

AUSTRALIAN CoCo/MiCo			AUSTRALIAN RAINBOW/GoCo
☐	Latest	3.25 per copy	☐
☐	6 months	$18	☐
☐	12 months	$29	☐

Back Issues

$3 per copy

☐ 1st Issue '83 ☐ 1st Issue '81

DISKS & TAPES

Rainbow on Tape (program listings) $12 for month of ☐
Please note that RAINBOW on TAPE is issued irregularly

Tape Monthly CoCoOz		MiCoOz		
☐	Latest $6	☐	Blank tapes 12 for $18 or $1.70 ea.	☐
☐	6 months $32	☐	Cassette Cases 10 for $5	☐
☐	12 months $58	☐	Disks — $3.50 ea. ☐	10 for $28.99 ☐

BOOKS

Byte Elementary	$5.95	☐
Help Medium	$9.95	☐
Facts Advanced	$11.95	☐
MiCoHelp Medium	$9.95	☐
MiCo Exposed V. Advanced	$11.50	☐

BULLETIN BOARD

CoCo Link — Annual Sub. $29 ☐

CUT OUT AND MAIL or PHONE BANKCARD NO. (075 51 0015)

☐ VISA ☐ BANKCARD ☐

_____ Cardholder

☐ CASH
☐ CHEQUE
☐ POSTAL ORDER

New Subscription ☐ Renewal ☐

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

BLOCK CAPITALS PLEASE

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

Mr ☐ Mrs ☐ Miss ☐ Initials Surname

Address

PC

Telephone Number STD Code Local Number

CoCoOz and MiCoOz this month will be $10.00 each unless you subscribe, in which case they wont cost you any more.

Holding the equivalent of two months of programs, MiCoOz this month has a whole heap of games, but also has a very handy assembler included.

CoCoOz had 17 programs on it when I last saw it, and we hope to be able to drop a couple of extras onto it (if there's room), that weren't refered to in Australian CoCo.

CoCoOz has a good mixture of advanced utilities, easy educational games, and even an adventure, as well as TWO accounts packages!

The Brother Connection

- Interface your CoCo to a Brother electronic typewriter.

by Brian Bere—Streeter

When it comes to creating hardcopy from your computer, the most common method is to use a Dot-matrix printer or a Daisy-wheel printer. However, whilst the Top-of-the-line Dot-matrix printers can approach letter quality, for presentation material the Daisy-wheel is without equal. (I don't agree - G). Many Daisy-wheel printers are quite expensive and when all accessories are added, represent a considerable investment. The logical alternative is to interface a current model electronic typewriter to the computer, and the Brother range is ideal for this purpose. Brother commenced the range with the CE-60 and CE-50, but now have a wider range including the budget line CE-25. All of these use the Brother IF-50 interface to connect to computers. In my case I had a CE-60 before I bought my CoCo, and naturally wanted to interface this to my CoCo. The following is an attempt to help smooth the way for those who wish to follow and avoid some of the pitfalls.

HARDWARE CONSIDERATIONS.

Obviously you require a CoCo, a Brother CE Series typewriter and a Brother IF-50 interface. Additionally you will require to make a special cable to connect the CoCo to the IF-50. The standard Tandy 26-3014 4-pin Din to DB-25 is NOT suitable, as they have removed all of the DB-25 pins except the four connected to the four cable cores coming from the 4-pin Din plug. The method Tandy uses to output the RS-232 signal to the printer is a simplified form of the standard connections, and we need more than four pins for the additional connections in the DB-25 plug needed to "fool" the IF-50 into believing it is connected to a standard RS-232 system.

Obtain a standard Tandy 4-pin Din to 4-pin Din cable 26-3020 and a Dick Smith P2692 DB-25 male plug and backshell. Cut one of the 4-pin Din plugs off and strip the cable cores ready for soldering to the DB-25 pins.

Connect the Green core from pin 2 of the 4-pin Din plug to pin 20 of the DB-25 plug. Connect the Red core from pin 3 of the 4-pin Din plug to pin 7 of the DB-25 plug. Connect the White core from pin 4 of the 4-pin Din plug to pin 3 of the DB-25 plug. Note that the yellow core from the 4 pin Din plug is NOT USED. The basic connections are now done, but 4 additional DB-25 pins need to be bridged for the interface to operate. Using small insulated bridge wires, connect pins 4,5,6 and 8 of the DB-25 plug together. This operation picks up a "high" signal output by the interface on pin 4 and returns it on pins 5,6, and 8, thereby "fooling" the interface into believing it is connected to a standard RS-232 system.

Fit the backshell to the DB-25 plug and check the pin to pin connections with a Multimeter for correct continuity, and the hardware considerations are now met.

SOFTWARE CONSIDERATIONS

Most Daisy-wheel printers have a hardware feature which signals the computer to stop outputting text to the printer, when the end of the single sheet page is reached. Typewriters, however, do not have this feature, and therefore the software must perform this function, Tandy's Scripsit word processors output the text to the printer in one continuous document, requiring continuous paper feed and are therefore NOT suitable for this role.

Additionally Disk Scripsit uses a unique printer driver, and only checks for a change in state of the Stop bit in the RS-232 output. The IF-50 requires that the Stop bit to be accurately timed, and so gets out of synchronism with the Scripsit output, resulting in printed "garbage".

(Brian is a recent import to Queensland from Sydney and is to blame, er, creator of, the OS8 mouse seen in last month's Australian CoCo.)

Current word processors that have the necessary PAGE PAUSE function include V.I.P.WRITER, TELEWRITER 64, ELITE WORD and TEXTPRO III. Personally I have read the manuals for Telewriter and Elite Word, but have not seen that for Textpro.

V.I.P. Writer has an additional feature refered to as PRINT PAUSE, and whilst the manual refers to a probable incompatibility with printers with buffers (as has the IF-50), I have found this works fine.

The big advantage of this feature is that it enables you to stop the printer anywhere on a page and change daisy-wheels or printer ribbons (including other colours), thereby getting the most out of your Daisy-wheel typewriter. To me, V.I.P.Writer is the only word processor to use with a CoCo and Daisy-wheel typewriter. (No taste - G.) Also I have found that most spreadsheets (Spectacular, Dynacalc-RS Dos version) and data bases (Profile) work satisfactorily, provided the print run does not exceed a single page, as page pause is generally not provided in this type of program.

I hope the above notes have been of some use to you and encourage you to use your Mighty Little CoCo for serious business use.

Concordance — An Aid To Programming Development

Are you a frustrated programmer? Do you have trouble debugging someone else's program or worse yet, your own? Did you finally convince the "real" head of the household that you needed a printer only to go crazy trying to read an *LLISTed* program? Have you ever made a brilliant modification to a magazine program only to discover that the variable you used was already used somewhere else? Well, cheer up Bunky, Mr. Bill Wasson of Echo Soft has released a new utility that should make your life considerably easier.

The program requires a minimum of 32K and Extended BASIC and is entitled *Concordance*. Say what? Yes, I confess, I had to consult Webster's on this one myself. Definition: "an alphabetical index of the principal words in a book." BASIC translation: "A nicely formatted listing, a cross-reference of all referenced line numbers and an alphabetical listing of all variables and the line numbers in which they appear." Interested? Read on and I will go into a more detailed explanation.

The actual program that does all the work is written in machine language. This program is preceded by a BASIC front end program that allows you to select from many options and actually customizes the machine language code for your individual needs. Upon loading, *Concordance* asks you if you wish to make a backup copy, which it does automatically. For a utility, especially a modifiable one, this is a real benefit and shows a great deal of professionalism on the author's part. Before I forget, included with the program is a well-written, seven-page instruction booklet which covers in detail all the various options as well as how to get the most benefit from the results of this program.

Among the many options included, are the loading of *Concordance* into the upper 32K of a 64K machine, setting the printer Baud rate, selecting the type of format, i.e., 'pretty print' or 'standard' and setting the printer margins which include top, bottom and left side as well as number of lines per page and number of characters per line. Of the two types of format the standard is most similar to *LLIST* except that the line numbers are offset to make the listing more readable. The only drawback to this is, if you wish to duplicate listings in a magazine such as RAINBOW by setting the printer width to 32, you still will not get an exact match because of the offset line numbers. The other, and more impressive, format is the 'pretty print.' 'Pretty print' places one statement per line, indents both *IF. . . THEN* and *FOR . . . NEXT* statements and adds spaces wherever necessary to make everything more readable. Also included in both formats are automatic page numbering and the ability to enter a heading to be printed at the top of each page (I use program name and date). All in all, once you use the 'pretty print' listing, going back to the normal *LLIST* is like a return to the stone age.

Although the 'pretty print' option is impressive it's only the tip of the iceberg. Next comes the line number cross reference listing. This is a list of all line numbers referenced by *GOTO* or *GOSUB* as well as all *PEEKs* and *POKEs* and their addresses. This makes unraveling "spaghetti" code much less of a nightmare, and the ability to easily isolate all *PEEKs* and *POKEs* makes debugging a pleasure. (Well, almost!)

Finally, *Concordance* produces a variable cross reference listing showing in alphabetical sequence all variables and the line numbers in which they are used. It even highlights any variables used in *PEEK* or *POKE* statements. In addition to all of this, *Concordance* does this very quickly. If you have ever used a BASIC 'pretty print' or cross reference program, you'll really appreciate *Concordance*. The only reason this program runs longer than a straight *LLIST* seems to be that it uses more paper. When the program listing finishes the cross reference listing it prints it immediately, there's no hesitation.

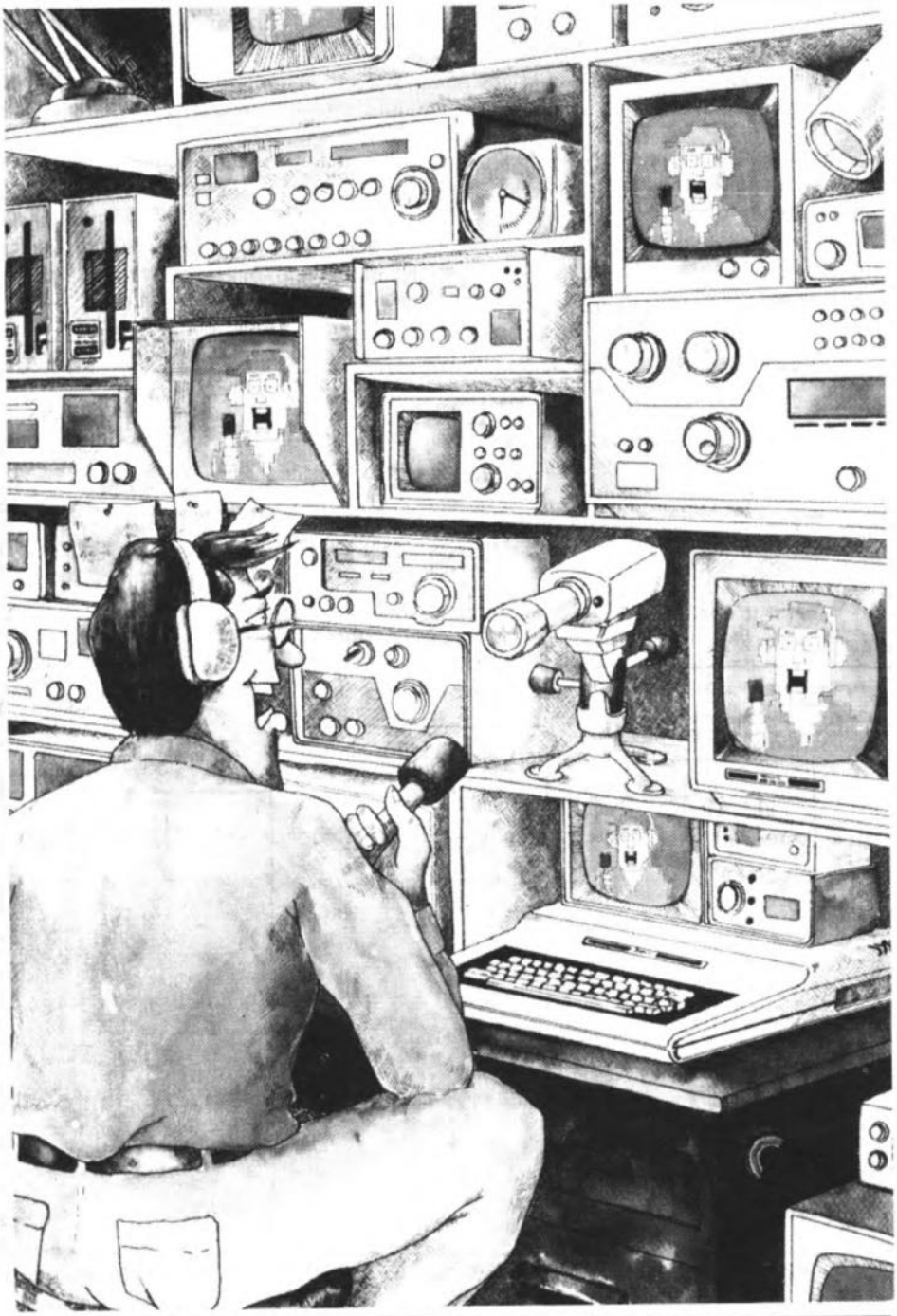
The next best thing to having a printer is this program. Enclosed with each program is a personal note from Mr. Wasson providing you with his home phone number should you have any trouble or questions.

(Echo Soft, 17 Skyline Dr., Chalfont, PA 18914, cassette $21.95, disk $24.95)

— Ken Boyle

Ham Radios And CoCos — A Great Combination

By Dan Downard — Rainbow Technical Editor



This month's theme is communications as you can see by our cover. I believe that a standard accessory in any ham shack today is a computer, and from the letters we receive quite a few are CoCos. As a ham operator (K4KWT), I think it would be appropriate to mention some of the CoCos uses in communications. I feel we owe the non-hams an explanation of what they are missing.

Computers and Ham Radio

I, for one, was introduced to computers via ham radio. If I may be a bit nostalgic, the days are not long past when I made the statement "Copy code with a machine? Impossible!" The first computer I had was an "Explorer 85" single-board system that had the total sum of 1K of RAM. The first thing I programmed it to do was send Morse Code. I could only send 0 through 9 and A through F as it only had a Hex keyboard, but what an accomplishment. Moving up to a 4K system with a 6800 microprocessor allowed me the luxury of receiving code. The 6800 system was purchased at a local "hamfest" and a homemade terminal was remarkedly similar to the modern Color Computer, 6847 VDG and all.

Communications

You can not only send and receive Morse code, Slow-Scan TV, RTTY, ASCII and Facsimile with a CoCo, but perform a myriad of technical tasks. My transceiver has the facility of remote tuning and operation from a computer via a serial interface. Another dream I used to have was operation of the ham

(Dan Downard is an electrical engineer and has been involved in electronics for 24 years through ham radio (K4KWT). His interest in computers began about five years ago and he has built several 68XX systems.)

station from a remote location, such as at work during my lunch break. Just call home, punch a couple of tones on the telephone Touch-Tone pad that would activate the receiver, and see what new DX (foreign) countries were on the air today. It's no longer a dream, and the systems get more sophisticated with each new generation of radios. By the way, almost all ham transceivers use microprocessors for control these days. The possibilities are unlimited.

The CoCo is a natural for communications. Not only do we have one of the most advanced microprocessors on the market, the 6809, but built-in serial I/O, A/D, D/A and expansion capabilities.

Programs are available that send and receive CW, RTTY, ASCII and Slow-Scan TV just by inputting audio from your receiver to the cassette (A/D) port on your CoCo. They lack the selectivity of the elaborate filtering networks used in some commercial communications equipment, but they work. If you have a friend who is a ham, ask him to give you a demonstration of Slow-Scan TV. Imagine receiving a picture from as far away as Japan with a radio receiver and a CoCo, and talking to the operator on the other end at the same time. The technology for this communications mode has been around for awhile, but computers are making it affordable for anyone.

A new horizon of computer applications in ham radio is called "packet radio." High speed messages are being relayed from station to station using CoCos. The 6809 is the standard micro for several packet schemes. In the future, an entire system of repeaters will allow any ham to send a message across the country activating each repeater automatically until the message reaches its destination.

On a local scale, repeaters have controllers using CoCos, allowing phone calls using a hand-held walkie-talkie. These frequencies are also used to exchange information nightly on CoCos. Ham bulletin boards, or mail boxes, differ only due to the fact that radios are used instead of modems and telephone lines.

Operator Aids

One of the things I enjoy is contests. The object of contests is to see how many other stations you can contact in a specified amount of time, whether it be in foreign countries, or maybe different states. What did I do without my CoCo? Elaborate cross-reference (dupe) sheets were designed so that you would not talk to the same station twice, wasting valuable time and points. Maybe in one weekend you would talk to as many as 1,000 other hams. Now it is as simple as typing in the other station's call letters. Not only will your CoCo check for dupes, but print an entire log of your contacts in a few minutes, a chore that used to take days.

For the everyday operator, CoCo databases allow instant retrieval of information on the ham you are talking to, whether the DX station is a new country, and what your antenna heading should be for a particular country. Your CoCo will even design your next antenna for you.

What's Next?

I guess you can sum the computerists and ham radio operators in the same quote, whose author I don't recall. "The difference between men and boys is the price of their toys." We're boys, and girls, with expensive toys. I would also like to think of us as young Thomas Edisons at the threshold of a new discovery or invention.

Creating Moire Patterns in PMODE 4

Anyone who has worked with the TRS-80's Hi-Res graphics knows that they are far from perfect. When you put various lines near each other, they might be in different colors. This is known as the moire (mor-AY) effect. Using this fault of the computer system, and optical illusions created by crowded lines, I discovered that it is possible to get eight colors on the *PMODE 4* screen, not including the background color! This creates incredible possibilities. Apparently, Radio Shack and Motorola realized this when creating the 6809 — the special effect only happens when using *SCREEN 1,1*. This is useful, because the colors are hard to control, and could pose a problem when not wanted (i.e., drawing electrical schematics). If you end up with unwanted colors in your Hi-Res programs, then use *SCREEN1,0* (more information can be found in Chapter 4, *Going Ahead With Extended Color BASIC*).

The first program is titled *8-COLOR*. It shows eight lines on the screen, each one a different color. You may have to adjust some of the controls on your TV to tell the difference, but the difference is there. I won't give names for the different colors, because they are slightly different on every CRT, and they can be altered by the color and tint controls.

Line 50 draws a vertical line in an even-numbered column (100). Line 60 draws a vertical line in an odd-numbered column (121). The others use the moire effect to create other colors by blending lines that are next to each other. As far as I know, horizontal lines are the same color at all times.

The second program shows a true moire pattern. Before computers, moire patterns were two designs on transparent materials that created optical illusions when placed one atop another. Often, one was a set of concentric circles, and the other consisted of many lines going from the middle to a little more than the outermost circle. (See Figures 1 and 2.) This program draws 85 lines in *PMODE 4*, from (129,0) to the bottom of the screen in steps of three, from (0,191) to (255,191). Without the space, it would look like a triangle at the bottom of the screen, and with a larger space the optical illusion wouldn't exist.

When the patterns of Figure 1 and Figure 2 are placed on top of each other the moire pattern is formed.

The third program is one that caused much frustration when we wrote it, and sometimes an FC Error will still result, but very rarely. It randomly picks two points, and draws a line between the two. Then it moves both points and draws a new line. When the point hits the end of the screen, it will bounce off at a 45-degree angle. This can result in some very interesting turns. I have added many options, which have made the program much more enjoyable, but can easily be removed to save time and memory by deleting Lines 240 to 320, and 350 to the end.

The CLEAR key will clear the graphics screen, because after awhile the screen will look jumbled. Hitting 'E' will End the program, and the computer will tell you the coordinates used. 'S' will let you change the Step increment (or gap between lines). This is set by the computer at four when you start. 'W' will run a new random line using your specified Step increment, while 'R' will Run a new random line using the computer's set gap of four. 'P' wil Pause the program (like SHIFT and '@'), but you must hit 'P' again to continue. 'O' will run the same pattern Over again and 'C' will Change screens (from *SCREEN 1, 1* to 1, 0 or 1, 0 to 1, 1), to show you how to prevent the various colors. 'H' will display a Help menu in case you are like me and will forget which letters to use.

The fourth program is a demonstration of my favorite pattern from the bouncing lines program. To fully appreciate it, you must wait until it starts overlapping itself, then pause it and play with the color controls. It contains no options, because there is no need for them.

I would like to know if someone can figure out how to make the third program 100 percent errorless.

Listing 1:

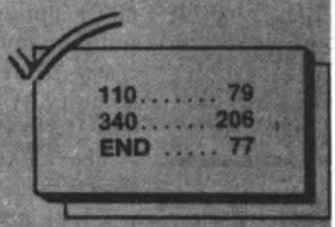
```
10 REM 8 COLORS IN PMODE 4
20 PMODE4,1
30 PCLS
40 SCREEN1,1
50 LINE(100,0)-(100,191),PSET
60 LINE(121,0)-(121,191),PSET
70 LINE(130,0)-(130,191),PSET:LI
NE(131,0)-(131,191),PSET
80 LINE(151,0)-(151,191),PSET:LI
NE(152,0)-(152,191),PSET
90 LINE(169,0)-(169,191),PSET:LI
NE(170,0)-(170,191),PSET:LINE(17
1,0)-(171,191),PSET
100 LINE(186,0)-(186,191),PSET:L
INE(187,0)-(187,191),PSET:LINE(1
```

```
88,0)-(188,191),PSET
110 LINE(201,0)-(201,191),PSET:L
INE(202,0)-(202,191),PSET:LINE(2
03,0)-(203,191),PSET:LINE(204,0)
-(204,191),PSET
120 LINE(212,0)-(212,191),PSET:L
INE(213,0)-(213,191),PSET:LINE(2
14,0)-(214,191),PSET:LINE(215,0)
-(215,191),PSET
130 GOTO 130
```

Listing 2:

```
10 PMODE 4,1
20 PCLS
```

```
30 SCREEN1,1
100 FOR T=0 TO 255 STEP3:LINE(12
9,0)-(T,191),PSET:NEXT
200 GOTO 200
```



Listing 3:

```
10 REM BOUNCING LINE
20 Q=4
30 SCR=1
40 PMODE4,1
50 PCLS
60 SCREEN1,1
70 V1=Q:V2=Q:V3=Q:V4=Q
80 REM STRING ART DESIGN
90 X=(RND(INT(255/Q)))*Q:Y=(RND(
INT(191/Q)))*Q
100 A=(RND(INT(255/Q)))*Q:B=(RND
(INT(191/Q)))*Q
110 SA=A:SB=B:SX=X:SY=Y
120 A=A+V1
130 B=B+V2
140 X=X+V3
150 Y=Y+V4
160 IF A=(INT(255/Q)-1)*Q THEN V
1=V1*-1
170 IF B=(INT(191/Q)-1)*Q THEN V
2=V2*-1
180 IF A=<0 THEN V1=V1*-1
190 IF B=<0 THEN V2=V2*-1
200 IF X=(INT(255/Q)-1)*Q THEN V
3=V3*-1
210 IF Y=(INT(191/Q)-1)*Q THEN V
4=V4*-1
220 IF X=<0 THEN V3=V3*-1
230 IF Y=<0 THEN V4=V4*-1
240 A$=INKEY$:IF A$="E" THEN 350
250 IF A$="S" GOTO 600
260 IF A$="R" THEN RUN
270 IF A$=CHR$(12) THEN PCLS
280 IF A$="P" GOSUB 380
290 IF A$="O" THEN 390
300 IF A$="H" GOSUB 430
310 IF A$="C" GOSUB 580
320 IF A$="W" THEN 30
330 LINE(X,Y)-(A,B),PSET
340 GOTO 120
350 CLS:PRINT:PRINT"THE VARIABLE
SX,Y,A,B WERE:":PRINTSX,SY,SA,SB
360 PRINT:PRINT
370 END
380 X$=INKEY$:IF X$<>"P" THEN 38
0 ELSERETURN
390 X=SX:Y=SY:A=SA:B=SB
400 V1=Q:V2=Q:V3=Q:V4=Q
410 PCLS
420 GOTO 120
430 SCREEN0,0
440 CLS
450 PRINT"           help"
460 PRINT" 'E' = eND"
470 PRINT" 'S' = sTEP CHANGE"
480 PRINT" 'R' = rUN"
490 PRINT" 'P' = pAUSE"
500 PRINT" 'CLEAR' = clear
510 PRINT" 'O' = RUN oVER"
520 PRINT" 'C' = cHANGE SCREEN"
530 PRINT" 'W' = RUN wITH NEW IN
CREMENT"
540 PRINT" 'H' = hELP"
550 INPUT" PRESS 'ENTER'";ZZ
560 SCREEN1,1
570 RETURN
580 IF SCR=1 THEN SCR=0 ELSE SC=
1
590 SCREEN 1,SC:RETURN
600 SCREEN 0,0
610 CLS
620 INPUT" WHAT IS THE STEP INCR
EMENT YOU   WANT";Q
630 SCREEN1,1
640 GOTO 30
```

Listing 4:

```
10 PMODE4,1
20 PCLS
30 SCREEN1,1
40 V1=4:V2=4:V3=4:V4=4
50 REM STRING ART DESIGN
60 X=132:Y=148
70 A=56:B=112
80 LINE(X,Y)-(A,B),PSET
90 A=A+V1
100 B=B+V2
110 X=X+V3
120 Y=Y+V4
130 IF A=252 THEN V1=V1*-1
140 IF B=188 THEN V2=V2*-1
150 IF A=0 THEN V1=V1*-1
160 IF B=0 THEN V2=V2*-1
170 REM ALL DONE FOR A AND B
180 IF X=252 THEN V3=V3*-1
190 IF Y=188 THEN V4=V4*-1
200 IF X=0 THEN V3=V3*-1
210 IF Y=0 THEN V4=V4*-1
220 GOTO 80
```

Welcome To The World Of Telecommunications

R. Wayne Day

The world of communications for a Color Computer enthusiast is almost unlimited. CoCo owners are hooking into mainframe computers, radios, satellites, bulletin boards . . . you name it, and a CoCo owner has probably tried it.

Beginning this month, THE RAINBOW and I will try to ease your journey through the world of telecommunications through this column, as well as give you an idea of some of the other communications applications that you can try with your CoCo.

I'd like your help in this effort, though. If you have some particular problem that's been plaguing you, or you have an interesting application for using the CoCo to communicate with someone or something else, be sure to let us know about it.

What Are We Talking About?

A simple description of telecommunications (simple if you talk computerese) is "remote operation of a host system through a modem by means of a terminal emulator."

Simple, right?

Let's take "terminal emulator" and see what we can decipher in that, first.

In the world of computers, quite a few of those used by businesses and universities are not the personal computers that you and I are familiar with. Unlike a CoCo, they normally do not have a keyboard attached to the computer itself, and certainly don't use a television for output.

So, how do the "big guys" talk to their computers?

They attach terminals, input/output devices, to them.

Terminals come in all shapes and sizes. Most all of them include a keyboard for the operator to input data; some of them have video screens for the output, while others have printers attached to them.

How does that fit in with the CoCo?

What we're doing with the CoCo is running a program that makes our computer *act like a terminal* . . . converting anything we type on the keyboard into ASCII (American Standards Code for Information Interchange) standard characters that can be understood by another system.

On the flip side, we translate the ASCII characters the other computer is sending us, into data that the CoCo can display.

The "remote" portion of that sentence means that we're somewhere else than the system we're "talking" to (also known as the "host").

And, "modem" is an abbreviation for MOdulator/DEModulator, the hardware device that goes in between our computer and the telephone lines. It modulates, or converts, the electrical pulses the terminal program generates into audio tones that can be sent over the phone lines.

In return, it demodulates the audio tones from the host system, into electrical pulses that our computer can use.

Who Are These Hosts I Can Talk To?

The electronic "hosts" come in all shapes and sizes.

One of the more popular aspects of modem-ing is to hook into one of the thousands of Bulletin Board Systems that have sprung up around the world in the last few years.

These BBSs, which we'll go into more detail about later, may be of general interest, or dedicated to a particular subject. There are quite a few BBSs dedicated to the Color Computer, for example, while others may be of particular interest to owners of other computer systems, while still others may be dedicated to Adventure games, amateur radio, geneology, or maybe one of the many "boards" that are in-tune with a more adult crowd.

There are also quite a few "VIDEOTEXT" services available throughout the United States and Canada today that are available to CoCo owners through their modems.

One such service, called "STARTEXT" is a joint project of the *Fort Worth Star-Telegram* newspaper and Radio Shack, and is available to computer users in the Fort Worth-Dallas area.

Subscribers dial up a local telephone number and are answered by the STARTEXT host computer, which dishes out the news items, classifieds or other information that the computer user has asked for, all at $7.95 a month for unlimited access to the system.

Other popular electronic services are the nationwide consumer-oriented information services, such as CompuServe and The Source.

On these services, you'll find a multiplicity of things to do — shopping at home, on-line travel planning, electronic magazines, and a variety of services that are geared to a specific family of computer users.

One such service is The Color Computer Special Interest Group (CCSIG) on CompuServe, of which I am the systems operator (SYSOP).

The CCSIG is devoted to topics of specific interest to Color Computer users and can be compared to one of the local bulletin boards that are dedicated to the CoCo, but only up to a point.

For example, a local BBS normally can handle only one person using it at a time, while on the CCSIG, since CompuServe is a multi-user facility, there's really no limit to the folks that can be "on" the SIG at one time.

One of our more popular features, as a matter of fact, is based on the multi-user concept; CONFERENCE, where anyone can come on and "talk" to other users on the SIG. We have had as many as 30 users all "talking" to each other during a conference featuring Ken Kaplan, the president of Microware, who was talking about the OS-9 implementation on the CoCo.

Another popular feature of the CCSIG, as well as on local BBSs, is the database section of the service. In these databases, you can usually find programs which are submitted by other users, available for "downloading" into your computer, so you can load it and run it later.

Both The Source and CompuServe

can be accessed through commercial communications networks called TYMENET and TELENET. Additionally, CompuServe also maintains its own communications network, so that access to either network is usually just a *local* phone call away.

On-line systems are not limited to hobby use, either.

Color Computer users with terminal programs and a modem can also send "electronic mail" through the Post Office's ECOM and MCI Mail. The advantage in using one of these services, over the electronic mail offered by CompuServe or The Source, is that the recipient does not have to own a computer in order to retrieve the mail. Instead, the correspondence can be sent through the normal carrier-delivered mail.

Typesetting firms are now establishing electronic links so that computer users can come on-line with their systems, and send a text file to be typeset by the firm, saving the time needed to send the copy, be it advertising or a newsletter, to the typesetter, as well as reducing the manual labor needed to get the copy into typeset form.

I'm Convinced! What Do I Need To Get Started?

To telecommunicate over the telephone, the minimum configuration you'll need is a computer, a modem, and terminal software.

All of the Color Computers have an RS-232 port on the back of the computer. The modem will attach to the RS-232 port, either with a cable supplied by the modem manufacturer, by Radio Shack, or one that you can make on your own.

Modems come in two basic styles: direct connection to the telephone lines, or acoustically coupling the telephone with the modem.

The acoustically coupled modem is usually less expensive, and does not require any electrical connection to the telephone system, if that's a concern, or you plan to use the modem on a party line or a pay phone (that's a no-no for direct-connect modems, according to federal regulations).

The handset fits into two rubber cups on the modem, and a little speaker and a microphone inside those cups take the tones from the phone, and work them into the innards of the modem.

A direct-connect modem, on the other hand, usually has a modular-type plug so that you can just plug the phone line right into the modem. Being directly connected to the phone line, there's less chance of incorrect data being passed because of background noise in the room.

Direct-connect modems can be "plain Jane" like the Radio Shack Modem 1, or can have some extra bells and whistles like automatically dialing the phone for you, or automatically answering the phone, which is needed if you ever want to run a BBS of your own.

Do you need a fancy modem? No, not really. What you're paying extra for is convenience and extra capabilities. You might want to get started with a less expensive modem, then upgrade later, as your needs and desires increase.

In a future column, we'll take a more in-depth look at modems, and discuss some of the various configurations they come in.

The Terminal Program

The terminal program is the real key to successful telecommunicating.

Terminal programs come in two basic versions — smart and dumb.

The "dumb" terminals allow you to type information on your keyboard and have it sent out the RS-232 port. They also receive info on the RS-232 line, and put it up on the screen. Usually, that's about all they do.

Assume, though, for a minute, that you are going to call up a local BBS and want to check your electronic mail, and get a couple of new programs that have been put on the board by other users.

If you had a terminal program that would allow you to load a message that you had composed on a word-processor before you logged onto the BBS, that would save you a lot of time over hand-entering the message to someone else. If the BBS was a long-distance phone call, that time would equal money in your pocket.

Then, too, if your terminal program had the capability to "capture" the data sent from the BBS, you wouldn't have to write down your mail, letter by letter, in order to have some way of getting it on paper. You could just print a copy of the mail out on your printer.

Now, on that downloaded program you wanted to save, you could just print it out and re-enter it if it were in BASIC, but what happens if it's in machine language? If your terminal had some way of sending and receiving data, and checking to make sure that data was correct, and it could save that data to disk or tape for you, your problem would be solved.

All of those capabilities exist, in some form or another, on programs we refer to as "smart terminals," and that will be our major topic of discussion next month in THE RAINBOW's annual Data Communications issue.

TURN OF THE SCREW

The Modem To Printer Connection

Tony DiStefano

Of all my projects, the short and fast ones seem to be the most popular. The ones that seem to better the computer and help the user on his quest for good computing are the ones that people call me to thank me for. I also get ideas from these people. For instance, the "Dual Cassette" project was an idea I got from a reader. When I presented this, I had forgotten his name, and wanted him to call me. Well, he did; his name is Lennie James. Thank you, Lennie, for the idea. The basis of this month's article actually came from several people. It is based on the RS-232 port of the Color Computer. The original question was this: Is there a way to connect a printer and a modem together so that everything that comes from the modem can also go to the printer at the same time? The answer is "yes." There are many ways of doing this. Some are very easy and fast, others require a bit more work and money. I'll tell you the theory on how to do it and let you decide on what method to use.

What is RS-232 anyway? The full name for this is EIA RS-232C. EIA stands for Electronic Industries Association. The EIA RS-232C standard defines the interfacing between data terminal equipment and data communications equipment employing serial binary data interchange. Electrical signal and mechanical aspects of the interface are well specified. The complete RS-232C interface consists of 25 data lines. This would seem to be enough signals for a complex parallel communication line, but many of the 25 lines are very specialized and a few are undefined. Most computer terminals only require from three to five of these lines to be operational. Table 1 briefly describes all 25 of the defined lines.

Table 1

PIN	DESCRIPTION
1	Protective Ground
2	Transmitted Data
3	Received Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Received Line Signal Detector
9	Unassigned
10	Unassigned
11	Unassigned
12	Sec. Rec'd Line Sig. Detector
13	Sec. Clear to Send
14	Sec. Transmitted Data
15	Transmission Signal Element Timing
16	Sec. Received Data
17	Receiver Signal Element Timing
18	Unassigned
19	Sec. Request to Send
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicator
23	Data Signal Rate Selector
24	Transmit Signal Element Timing
25	Unassigned

Table 2

PIN	DESCRIPTION
1	CD — Status Input Line
2	RS232IN — Serial Data Input
3	GROUND — Zero Voltage Reference
4	RS232OUT — Serial Data Out

The Color Computer uses only four of these lines. They are the four most used in small computers. Table 2 shows the pin and description for the Color Computer version of the RS-232. Pin 1 on the computer is equal to pin 5 or pin 8 on the EIA RS-232C; pin 2 on the computer is equal to pin 3; pin 3 on the computer is equal to pin 7; and pin 4 on the computer is equal to pin 2.

So much for the theory, now for the good part. The secret to this is to connect the Transmit (Serial Output) of the modem to the Receive (Serial Input) of the printer. Now there are many ways to do this. It all depends on what kind of equipment you have. If you are one who just unplugs your printer cable to plug in your modem, you will have the most to do. If you have one of the several switchers available for your modem and printer, all you need is a switch and a piece of wire.

Step 1

Follow these instructions if you have a switcher. If you don't have an SPST switch, RS #275-624 is good and small. First you have to take the switcher apart. You will need the right screwdriver. After the switcher is apart, locate the connector that the modem connects to. Solder one end of a piece of wire to pin 2 of that connector. Solder the other end of this wire to one end of a SPST switch. Solder one end of another piece of wire to the other end of the switch. Now locate the connector that the printer connects to. Solder the last end of wire to pin 4 of that connector. Mount the new switch somewhere in the switcher. Close up the switcher. I'll show you how to use it later.

Step 2

Follow these instructions if you do not have a switcher. Undo the modem connector that plugs into the computer. Solder a wire to pin 3 in the connector. Using a piece of tape, label this wire "G" for ground. Solder another wire to pin 2 of the connector. Reassemble the connector. Undo the printer connector that plugs into the computer. Solder a wire to pin 3 in the connector. Label this wire "G" for ground. Solder another wire to pin 4 of the connector. Reassemble the connector. Solder the two wires labeled G together. Solder the other two wires to each side of an SPST switch. Mount the switch any way you want.

In order that the printer prints all that comes in on the modem, the printer parameters must be set correctly. Most modem communications use 300 Baud. That means your printer must be set to 300 Baud. Other parameters, like seven or eight bits, even, odd or no parity, must also be set right. That will depend on what parameters the host computer is using. The fact is that all these parameters must be looked into before the printer will function right. Another thing I should mention is that the printer may or may not print what you type. That depends if you are working in full or half duplex mode. If you are in half

duplex, you will not see on paper what you type; with full duplex you will see it. At certain times you may not want to see what you type in, so just change to half duplex if the host computer will allow you.

The next thing you must do is set up the wiring correctly. If you are using Step 1, then you must set the switch you installed to the "on" position and the switcher to the modem side. When you want to use the printer alone, make sure that the switch is in the "off" position and the switcher is set to the printer side. If you followed Step 2, then plug in the modem connector and turn the switch on. When you want to use the printer, turn the switch off and plug the printer connector on.

During normal printing, there is handshaking going on between the printer and the computer. That is, before the printer sends out a character to the printer, the computer checks if the printer is busy. If it is, the computer will wait until the printer is ready. In modem communication, there is no such handshaking. That means if the printer is busy and the modem transmits a character, the printer will miss that character and not print it. This is especially true when the printer is doing a carriage return or line feed. If your printer has an input buffer and can print faster than about 30 characters per second (300 Baud) or 120 characters per second (1200 Baud) you will not miss any characters. Another way to avoid missing characters is if the host computer can be programmed to wait after every carriage return; the printer would have time to catch up.

If you have problems with one of my projects or you want to discuss one of your own projects, I have reserved Monday nights for this. I'll be happy to talk with you if you call me then. The number to call is (514) 473-4910. But limit the calls to Monday nights, any other time is forbidden fruit.

Well, that is it for this time, good modem printing.

EDUCATION NOTES

16K ECB

the RAINBOW

If At First You Don't Succeed . . . Read The Directions!

Steve Blyn

Are you the type who gets a new software utility, loads it in, and then promptly declares that it doesn't work? Or are you the type who reads the directions first? Too many of us are the first type. We have been wrongly trained in this 'Age of Marvels' to expect things to work automatically. Much to our dismay, sometimes they don't work that easily.

Children often follow in the footsteps of their parents. My children too often expect software packages to operate automatically. They reflect the shortcomings of the adults around them. Learning to read and follow directions are skills that we need to stress in the elementary grades and on through the higher grades.

A careless approach to new material is not restricted by any means to software. The same person who overlooks software directions will also overlook directions in a variety of other areas. New household items and appliances are frequent examples. Did you ever try to assemble a backyard swing set without reading the directions?

A child who does not read or listen to directions carefully in school can often receive a grade lower than his true abilities. Sometimes directions indicate certain ways of entering answers so they are not overlooked by the scorer. A careful reading of the directions would indicate this. Other times, children who don't read directions may respond incorrectly, fully believing that they are correct. An example of this would be giving synonyms when antonyms were asked for, or some other unintentional mistake.

Children are especially prone to overlooking directions on many of the standardized tests given to them. These tests can greatly affect decisions on the class placement of the child. Realizing the problem, many times teachers are instructed to read the directions aloud to the children while they read them to themselves before the test begins.

One exercise I have always found successful in classes is utilized in this month's article. A sheet of paper is distributed to each child. There are either nine numbers or letters printed on the sheet in Tic-Tac-Toe fashion. The point is to follow the leader's directions involving these numbers or letters. The directions may either be printed on the back of the sheet of paper, or read to the children by the leader.

The leader may either be the teacher, or another student. Of course, parents could do this at home to reinforce the skill. Also, the exercise is worthwhile and at the same time entertaining for a

parent and an impatient child waiting in a doctor's office, or an airport, as a non-computer activity. This exercise works equally well for oral or written directions.

1. Read this entire paper first
2. Draw a box around C
3. Draw a box around G
4. Draw a line from C to G
5. Draw a triangle around W
6. Draw a figure 8 around B and S
7. Draw a circle around M
8. Draw a #4 between the letters K and A

The first time that you try this exercise on a youngster, it is a good idea to add the additional written instruction,

9. Do not write anything on this paper.

You will be surprised how many youngsters will not get to the last instruction until they have done all of the instructions. It is a learned habit for many of us to ignore instruction one and plunge headlong into the task.

This 'trick' will, of course, only work the first time on any individual, but it should serve to drive home the point.

Our program draws the nine letters and the geometric shapes needed to complete the directions. Line 40 tells the computer to *GOSUB 300*. Lines 300 to 380 contain the drawings for the nine letters needed for the exercise. Line 390 draws a number 4 which we also need. Line 400 *RETURN* back to Line 50.

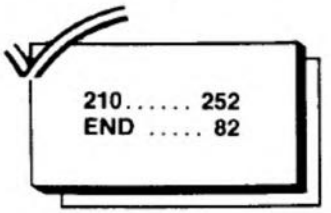
Lines 80 through 160 draw the nine letters on the screen in Tic-Tac-Toe fashion. Lines 170 through 180 wait for you to press the ENTER key. *(CHR$(13)* represents the ENTER key). When it is pressed, the program proceeds. Lines 190 through 250 draw the completed diagram with the instructions carried out. Pressing ENTER again will repeat the procedure.

This program will enable the child or class to view the results of the instruction set on the computer screen. They can easily compare it to their own paper to check for accuracy. The program could, perhaps, be used as a pretest and retest for before and after some of your teaching on the topic of improving the reading of directions.

It is an easy job to create other similar worksheets for the children to practice on. The directions are given by you on a separate sheet of paper or are read aloud by you. The only program changes would be on Lines 190 through 250.

It is both fun and good practice in learning about CoCo graphics to figure out the drawing of the various graphics around the letters. Perhaps some of your children could assist in creating some of these additional graphics. Either their ideas or their actual programming assistance could be encouraged.

The need to follow directions is just as important for teachers as for their students. We should all be careful to search the directions of our software purchases. Often there is an important first or last instruction that may tell us to make a backup copy to protect the original master disk. Another overlooked instruction may be to enter a password before beginning the program. We really can't guess what all of the essential instructions are until we carefully read the directions ourselves.



The listing:

```
10 REM"FOLLOWING DIRECTIONS"
20 REM"STEVE BLYN,COMPUTER ISLAN
D,NY,1984
30 PCLS:SCREEN1,0:PMODE3,1:CIRCL
E(50,50),10
40 CLS:GOSUB300
50 PCLS:PMODE3,1:SCREEN1,1
60 COLOR6:LINE(15,20)-(240,170),
PSET,B
70 PAINT(1,1),7,6
80 DRAW"S12;C8;BM50,50"+K$
90 DRAW"BM110,50"+G$
100 DRAW"BM170,50"+M$
110 DRAW"BM50,100"+A$
120 DRAW"BM110,100"+B$
130 DRAW"BM170,100"+S$
140 DRAW"BM50,150"+C$
150 DRAW"BM110,150"+Y$
160 DRAW"BM170,150"+W$
170 EN$=INKEY$
180 IF EN$=CHR$(13) THEN 190 ELS
E 170
190 COLOR6:LINE(40,130)-(70,155)
,PSET,B:REM"A BOX AROUND C
200 LINE(100,60)-(130,30),PSET,B
:REM"A BOX AROUND G
210 LINE(100,60)-(70,130),PSET:R
EM"A LINE FROM C TO G"
220 LINE(175,120)-(150,160),PSET
:LINE-(200,160),PSET:LINE-(175,1
20),PSET:REM"A TRIANGLE AROUND W
"
230 CIRCLE(125,90),30,6,.5:CIRCL
E(180,90),30,6,.5:REM"A FIGURE 8
 AROUND B AND S"
240 CIRCLE(178,45),20:REM"A CIRC
LE AROUND M
250 DRAW"S8;BM50,75"+FR$:REM" A
4 BETWEEN K AND A"
260 EN$=INKEY$
270 IF EN$=CHR$(13) THEN RUN ELS
E 260
280 GOTO 280
290 REM"HERE ARE THE LETTERS USE
D IN THE PROGRAM"
300 A$="BU4R3FDHL2GDFR2ENU2FBR2"
310 B$="BU6RD6NLBUFR2EU2HL2GBF3B
R3"
320 C$="BRNR2HU2ER2FBD2GBR3"
330 G$="BRNR2HU2ER2FNUD2NGD2GL2H
BEBR5"
340 K$="RU6NLBD4R2NE2F2BR2"
350 M$="BU4FND3EFND3EFD3BR2"
360 S$="BUFR2EHL2HER2FBD3BR2"
370 W$="BUNU3FENU3FENU3BR2BD"
380 Y$="BUNU3FR2ENU3D2GL2HBUBR6"
390 FR$="BR3U5G3R4BD2BR3"
400 RETURN
```

Trap That Error

By H. Allen Curtis

When the Color Computer is compared with the TRS-80 Model III by those who have used both, usually the Color Computer is favored. However, it is generally agreed that the Color Computer's keyboard and character set (no true lowercase) are inferior to the Model III's. Use of kits can eliminate the keyboard inferiority. Use of software can give the Color Computer a high resolution character set with a true lowercase. Sometimes the Color Computer is criticized because of its lack of BASIC commands for error trapping. The purpose of this article is to overcome this criticism.

The vehicle for overcoming the criticism is a program that adds to the Color Computer's repertoire of BASIC command analogs of the Model III's *ON ERROR GOTO, ERROR, ERL, ERR/2+1* and *RESUME* commands. Before the program called *Trapperr* is presented, a brief description of the new commands will be given. Then, after the presentation of *Trapperr*, there will follow a set of examples illustrating in detail the workings and applications of the new commands.

The *ETRAP* command is analogous to the Model III's *ON ERROR GOTO*. The format of *ETRAP* is simply as follows:

ETRAP line number

With *ETRAP* you provide a means of entry to an error-trapping routine which will permit your program to analyze an error when it occurs, take suitable action, and continue the program without any break in execution. The error-trapping routine starts at the line specified in the *ETRAP* command.

The next command, *ER#*, allows you to simulate a specified error. Its format is *ER# code* where code is any integer from one through 27 for Extended Color BASIC or from one through 38 for Disk BASIC. Associated with each code is a distinct type of error. Table 1 shows the codes and their associated errors. Execution of the *ER#* command will cause the computer to react as though the associated error had occurred. This command is a handy tool in the composition and testing of error-trapping routines. The *ER#* command is not generally used in your program once it has been completely developed.

The *ERL* command of the Color Computer is a function that returns the line number in which an error has occurred. *ERL* is primarily used in an error-trapping made accessible by an *ETRAP* command. If an error occurs in the direct mode, *ERL* returns -1.

The *ERR* command is the Color Computer analog of the *ERR/2+1* command of the Model III. *ERR* is a function that returns the code number (see Table 1) associated with the error that occurred. *ERR*, like *ERL*, is primarily used in an error-trapping routine accessed by an *ETRAP* command.

The final error handling command to be added to BASIC's command repertoire is *RESUME. RESUME* is used to terminate an error-trapping routine by specifying where normal execution is to resume. *RESUME* followed by a line number causes the computer to resume execution at the specified line. *RESUME* without a following line number causes the computer to return to the statement in which the error occurred. *RESUME NEXT* causes the computer to go to the statement directly after that in which the error occurred. If a programming bug causes the error-trapping routine to be reached without the occurrence of an error, you will receive an ID Error message. The ID Error message in such a case is intended to inform you that a *RESUME* has been executed without the occurrence of any of the errors in Table 1.

Trapperr, the program that adds *ETRAP, ER#, ERL, ERR* and *RESUME* to the BASIC command repertoire of your computer is shown in Listing 1. *Trapperr* employs a combination of two languages — BASIC and machine language. Each value in the *DATA* statements of Lines 90 through 500 is a byte of the machine language portion of *Trapperr*. This machine language routine consists of two distinct sections. The first section, found in Lines 90 through 190, initializes the mechanism that adds the error-trapping commands to the computer's command repertoire. The second section, found in the rest of the *DATA* statements, acts as an extension of the BASIC ROM and executes each error-trapping command when issued.

The purpose of the BASIC portion of *Trapperr* is to generate the machine language routine, reserve and protect high RAM, and transfer the routine to the reserved memory area. Lines 10 and 20 take care of the reservation and protection of the high 336 bytes of RAM. Lines 30 through 60 generate and transfer the machine language routine to the protected RAM. Line 70 provides a check on the accuracy of your typing the *DATA* statement values. Line 80 causes the execution of the initialization section of the machine language routine. This section is no longer needed after its completion and return to BASIC. Therefore, its 90 bytes are released from protection. The addition of the five error-trapping commands comes at a memory cost of only 246 bytes. There is an additional cost for those *without* Disk BASIC; you may no longer employ the first three *USR* functions *USR0, USR1* and *USR2*.

After you have correctly keyed in *Trapperr*, run it; it takes about five seconds. Then save *Trapperr* on tape or disk. Erase *Trapperr* by typing and entering *NEW*. Now, you are ready to become more familiar with the error-trapping commands via illustrative examples.

Example 1: Type and enter the following:

```
10  ETRAP1000
900 STOP
1000 PRINT"TEST"
```

Then type *RUN* and press ENTER. The result is the message, BREAK IN 900. In Line 10 the entry Line 1000 is established for the error-trapping routine. Because no error occurred in Lines 10 and 900, the error-trapping routine was never reached.

Try typing and entering *RUN2000*. Ordinarily, you would receive a UL Error message, but because of the previous execution of the *ETRAP* command, the occurrence of the UL Error caused entry to the routine at Line 1000 and the subsequent printing of the word TEST on the screen.

Again, type and enter *RUN2000*. This time the message UL Error is indeed displayed. This illustrates that once an error has occurred, another *ETRAP* command must be issued in order to trap the next error.

Example 2: Add to the Example 1 program the following:

```
20   PRINT"THIS IS A ":ER#16:STOP
```

Then change Line 1000 to:

```
1000 PRINT"TEST":RESUME
```

Displayed on the screen as a result of running this program is:

```
THIS IS A TEST
?ST ERROR IN 20
```

This display shows that after the completion of the *PRINT* statement of Line 20, the *ER#16* command simulated an ST Error that caused an interruption of the execution of Line 20 and an entry to the routine at 1000. At Line 1000 following the printing of TEST on the screen, the *RESUME* command brought about a return to the *ER#16* command in Line 20. Because there was no second execution of an *ETRAP, ER#16* this time produced the message, ?ST ERROR IN 20. Change Line 1000 to *1000 PRINT "TEST":RESUMENEXT* and then run the program again. Now, you should get printed on the screen the following:

```
THIS IS A TEST
BREAK IN 20
```

Because of the *RESUMENEXT* in Line 1000, the return was made to the *STOP* statement just after and *next* to the *ER#16* command. Thus, *ER#16* was only executed once in the changed version of the example program. The *STOP* in Line 20 accounted for the "BREAK IN 20" message.

Example 3: In the program of Example 2 change Line 1000 to:

```
1000 PRINT"TEST":PRINTERL:PRINTERR:
RESUMENEXT
```

Running this program yields the display:

```
THIS IS A TEST
20
16
BREAK IN 20
```

The execution of *PRINT ERL* and *PRINT ERR* in Line 1000 caused the printing of 20 and 16, respectively, before the execution of the *STOP* command in Line 20. The printed number 20 is the line in which the error occurred and 16 is the code number of the error.

Now, add the following line: *30 ER#5* and change Line 20 to *20 PRINT"THIS IS A ":ER#16*. As you would now expect, running this program should produce the following display:

```
THIS IS A TEST
20
16
?FC ERROR IN 30
```

Change Line 1000 to:

```
1000 ETRAP1000:PRINT"TEST":PRINTERL:
PRINTERR:RESUMENEXT
```

Running this program yields the following:

```
THIS IS A TEST
20
16
TEST
30
5
BREAK IN 900
```

The insertion of *ETRAP1000* in Line 1000 permitted the trapping of the error produced by the *ER#5* command in Line 30. The printing of 30 and 5 announced that the error occurred in Line 30 and was of the type having a code number 5.

Delete Line 900. Now the program has a bug: the error-trapping routine can be reached without the occurrence of an error. Run the program to get the following:

```
THIS IS A TEST
20
16
TEST
30
5
TEST
30
5
TEST
1000
12
```

The first half of the display is not surprising. The second printing of *TEST*, 30 and 5 is explained as follows: After the *ER#5* error at Line 30, the *RESUMENEXT* caused a return to the statement, immediately after *ER#5*, which happens to be the *ETRAP 1000* statement of Line 1000. The *ERL* and *ERR* values did not have a chance to change; hence, 30 and 5 were again printed after *TEST*. Because the error-trapping routine was reached illegally, an ID Error resulted. The occurrence of the ID error caused a legal entry to the error-trapping routine and thus the printing of 1000 and 12 after *TEST*. The error was at Line 1000 and was an error with code number 12. The final *RESUMENEXT* set up a return to the statement following the final statement of Line 1000. Since no line follows 1000, the program ended.

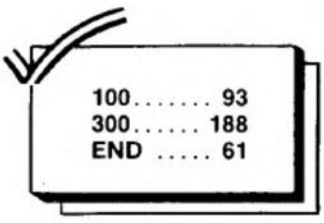
Example 4: Using *NEW*, erase the program of Example 3. Then key in the program of Listing 2. After you respond to the program's request to enter a message, the program forms a string consisting of a repetition of your message. The string formation continues until the program consumes all the string space allotted. Then the string is printed on the screen. This example illustrates the *ERL* function in an *IF . . . THEN* statement; it also demonstrates the use of a *RESUME* followed by a line number.

Table 1

Code	Error	Description
1	NF	NEXT without FOR
2	SN	Syntax error
3	RG	RETURN without GOSUB
4	OD	Out of DATA
5	FC	Illegal function call
6	OV	Overflow
7	OM	Out of memory
8	UL	Undefined line
9	BS	Bad subscript
10	DD	Redimensioned array
11	/0	Division by zero
12	ID	Illegal direct
13	TM	Type mismatch
14	OS	Out of string space
15	LS	String too long
16	ST	String formula too complex
17	CN	Can't continue
18	FD	Bad file data
19	AO	File already open
20	DN	Device number error
21	IO	Input-Output error
22	FM	Bad file mode
23	NO	File not open
24	IE	Input past End of file
25	DS	Direct statement
26	UF	Undefined function
27	NE	Can't find file
28	BR	Bad Record Number
29	DF	Disk full
30	OB	Out of Buffer space
31	WP	Write Protected
32	FN	Bad filename
33	FS	Bad file structure
34	AE	File already exists
35	FO	Field Overflow
36	SE	Set to non-field string
37	VF	Verification
38	ER	Past End of Record

Example 5: Erase the program of Listing 2 and type the program of Listing 3. The *ER#s* in Lines 20 through 50 are included for illustrative and testing purposes only and would be deleted in an actual working program. Have you ever wished that the Color Computer provided more than abbreviated error messages? This example indicates how you can go about replacing the abbreviated error messages with descriptive ones.

Before the conclusion of this article, there is an important comment which should be made. *Trapperr* must be loaded and run before you load and run any program containing error-trapping commands. Furthermore, it is a good policy to load and run *Trapperr* before keying in any program using error-trapping commands. *Trapperr* must be present to permit the error-trapping commands to be tokenized. If you should start composing a program containing error-trapping commands without *Trapperr* residing in the computer, all is not lost. Merely save the program in ASCII format. When you wish to use that program, load and run *Trapperr* first; then the error-trapping commands will be automatically tokenized upon the loading of your program.



Listing 1:

```
10 CLS:X=256*PEEK(116)-81
20 CLEAR100,X
30 X=256*PEEK(116)-80
40 FORZ=X TOX+335
50 READA$:A=VAL("&H"+A$):W=W+A
60 POKEZ,A:NEXT
70 IFW<>36869THENPRINT"DATA ERRO
R":STOP
80 EXEC X:CLEAR200,X+89
90 DATA CE,1,34,F6,1,92,C1,C2,26
100 DATA 17,30,8C,6C,86,E4,A7,81
110 DATA 86,E1,A7,8,86,52,A7,D
120 DATA 86,50,30,F,A7,4,33,4A
130 DATA 30,8C,2B,C6,B,BD,A5,9A
140 DATA E7,44,DC,74,33,56,A7,C1
150 DATA A7,C1,A7,41,A7,43,30,8C
160 DATA 34,C6,5,A7,81,5A,26,FB
170 DATA CE,1,8E,A7,41,86,7E,A7
180 DATA C4,86,7A,A7,42,39,3,3F
190 DATA 10,3F,28,2,3F,A,3F,37,0
200 DATA 45,52,CC,45,52,D2,45,52
210 DATA A3,45,54,52,41,D0,52,45
220 DATA 53,55,4D,C5,3F,45,3F,4D
230 DATA 3F,53,3F,5D,3F,A7,81,D1
240 DATA 25,3,7E,B2,77,30,8C,F0
250 DATA 80,CE,7E,AD,D4,C1,46,23
260 DATA 2,20,EF,C0,44,30,8C,DC
270 DATA 7E,B2,CE,30,8C,5B,EC,84
280 DATA 7E,B4,F4,30,8C,50,7E,B7
290 DATA 52,9D,A5,BD,AF,67,5A,58
300 DATA 7E,AC,46,DC,A6,34,6,BD
310 DATA AE,A4,33,8C,3E,AF,C4,35
320 DATA 6,DD,A6,9D,9F,27,4,81
330 DATA 3A,26,F8,C6,1,E7,59,39
340 DATA 34,12,30,8C,1F,A6,84,26
350 DATA 2,35,92,6F,80,A7,80,57
360 DATA 5C,E7,80,DC,A6,ED,81,DC
370 DATA 68,ED,81,EC,81,DD,A6,9E
380 DATA 21,32,1D,39,0,0,0,0,0,0
390 DATA 0,0,0,30,8C,F5,A6,84,27
400 DATA 46,6F,81,9D,A5,27,1A,81
410 DATA 3A,27,16,81,8B,27,3,7E
```

```
420 DATA AE,A6,8D,38,A6,C0,27,4
430 DATA 81,3A,26,F8,33,5F,DF,A6
440 DATA 39,8D,29,9E,19,30,2,EC
450 DATA 84,10,93,68,27,4,AE,1E
460 DATA 20,F3,30,2,9F,2B,11,93
470 DATA 2B,27,8,A6,C2,81,3A,26
480 DATA F5,20,DB,33,1B,20,D7,C6
490 DATA 16,7E,AC,46,EE,81,EC,84
500 DATA DD,68,39
```

Listing 2:

```
5 CLEAR500
10 ETRAP1000
20 CLS:PRINT@259,"ENTER A MESSAG
E"
30 LINEINPUTA$
40 B$=" "+A$
50 IFI<101THENA$=A$+B$:I=I+1:GOT
050
60 PRINT@288,A$
900 STOP
1000 IFERL=50THENRESUME60ELSERES
UME
```

Listing 3:

```
10 ETRAP2000
20 ER#2
30 ER#3
40 ER#1
50 ER#10
900 STOP
1000 PRINT"NEXT without FOR":RET
URN
1010 PRINT"SYNTAX error":RETURN
1020 PRINT"RETURN without GOSUB"
:RETURN
2000 ON ERR GOSUB1000,1010,1020
2010 IFERR>3THENRESUMEELSEETRAP2
000:RESUMENEXT
```

UTILITY

4K

EZ List

Michael Davidson

EZ List, a machine language utility program, was written to eliminate the need to use the SHIFT @ keys to control program listing. This program will allow you to use the up and down arrow keys to list one line at a time.

Type in the BASIC listing and save a copy to tape before running it. It will be erased after it has put the machine code into memory.

First the BASIC program will find the top of memory (Line 140). It will then clear enough space for the machine code (Line 150) and find the new top of memory on Line 160.

Lines 170 to 200 read the *DATA* statements and place them in the protected memory. Line 210 starts the machine code. Line 220 finds the start of BASIC to be used by Line 230. Line 230 places two zeros at the start of BASIC to effectively erase the BASIC program. Line 240 places zeros in a memory location that is used by the interpreter to remember what line it is working on.

The start section changes two memory locations to enable the computer to jump to the *EZ List* code when a key is pressed.

The second section, *PRINT*, is the first place the computer jumps to when a key is pressed. The print routine checks to see if the up arrow key has been pressed. If it has, its value will be changed from 94 to one. This will prevent the computer from printing the up arrow.

The third section, *INPUT*, is the main part of the program. It checks to see if either of the arrow keys have been pressed. If so, this routine finds the next higher and lower line numbers and their addresses. It then calls the *LIST* subroutine.

The *LIST* routine calls the ROM routines that decode the program line and print it on the screen.

One final note: If you are going to be loading several programs with *EZ List* running, use the *LIST* command as soon as the new program is loaded from tape or disk. The computer remembers where the last line was, even when a new program is loaded. Depending on what the line addresses are, *EZ List* may find the next line. If it doesn't, the computer will hang up. If this should happen, don't worry, just press the Reset switch and type *LIST* and press ENTER. This will restore control without destroying your program.

```
110 ..... 57
260 ..... 98
END ..... 180
```

The listing:

```
10 'EZLIST
20 'BY MICHAEL DAVIDSON
30 'FARGO ND
40 '11/83
50 'THIS BASIC PROGRAM IS USED
   TO LOAD THE MACHINE CODE
   INTO UPPER MEMORY
60 'AFTER THE MACHINE CODE IS IN
   PLACE THE BASIC PROGRAM WILL
   BE ERASED
70 'USE THE UP AND DOWN ARROW
   KEYS TO LIST ONE LINE AT A
   TIME
80 'THE LIST COMMAND WILL STILL
   WORK AS IT DID BEFORE.BUT
```

```
      NOW THE SHIFT @ COMBINATION
      WILL NOT BE NEEDED
  90  'WHEN LOADING PROGRAMS FROM
      TAPE OR DISK USE THE LIST
      COMMAND BEFORE USING THE UP
      AND DOWN ARROW KEYS
 100  'THIS WILL PREVENT THE
      PROGRAM FROM LOCKING UP
 110  'THIS HAPPENS WHEN THE LAST
      LINE LISTED IS HIGHER THAN
      THE HIGHEST LINE OF THE NEW
      PROGRAM
 120  'IF YOU FORGET AND IT LOCKS
      UP, JUST HIT THE RESET AND
      TYPE LIST AND <ENTER>
 130  CLS
 140  TM=PEEK(39)*256+PEEK(40)
 150  CLEAR 200,TM-240
 160  TM=PEEK(39)*256+PEEK(40)
 170  FOR X=TM TO TM+223
 180  READ A
 190  POKE X,A
 200  NEXT X
 210  EXEC TM
 220  P=PEEK(25)*256+PEEK(26)
 230  POKE P,0:POKE P+1,0
 240  POKE &H2B,0:POKE &H2C,0
 250  DATA 49,141,0,220,190,1,107,
175,164,48,141,0,36,191,1,107,49
,141,0
 260  DATA 214,190,1,104,175,164,4
8,141,0,4,191,1,104,57,129,94,38
,8,111
 270  DATA 130,90,134,1,50,98,57,1
10,157,0,185,52,119,51,141,0,169
,129
 280  DATA 1,39,20,129,10,38,10,14
1,26,236,70,221,43,174,72,141,12
2,53,119
 290  DATA 110,157,0,145,141,10,23
6,66,221,43,174,68,141,106,32,23
8,158
 300  DATA 25,236,132,38,4,50,98,3
2,228,220,43,39,60,16,131,255,25
5,39,54
 310  DATA 237,66,175,68,16,174,13
2,16,174,164,38,5,237,70,175,72,
57,16
 320  DATA 163,2,39,14,52,6,236,2,
237,66,53,6,175,68,174,132,32,23
7,16
 330  DATA 174,132,16,174,164,39,2
,174,132,236,2,237,70,175,72,32,
218,236
 340  DATA 2,237,70,175,72,237,66,
175,68,16,174,132,16,174,164,39,
200,175
 350  DATA 68,236,2,237,66,174,132
,38,238,32,188,52,16,189,189,204
,189
 360  DATA 185,172,53,16,189,183,1
94,206,2,221,166,192,39,5,189,18
5,177,32,247,189,185,92,57
```

Expanding

LITTLE E's Powers

H. Allen Curtis

Little E was first presented in the April '84 RAINBOW. It gave much-needed editing powers to the MC-10 and the CoCo with Color BASIC, and also gave the Coco with Extended Color BASIC cursor controlled editing facilities.

Little E will presently be endowed with vastly greater powers:

1) It will be used for the direct entry of new lines of BASIC, as well as for editing.
2) After the completion of typing a new line or editing an old one, the next line will be displayed for you to type or edit.
3) The cursor will be allowed upward movement. When the down arrow reaches its downward limit, it will move to a position three spaces from the top leftmost portion of the screen.
4) Whether typing or editing a line on the MC-10, you will be able to type commands with a single keystroke when the CONTROL key is depressed.
5) *Little E* will be the foundation of a rudimentary word processor which can even be used with a 4K CoCo or MC-10.

Upgraded versions of *Little E* for the CoCo and the MC-10 are presented in Listings 1 and 2, respectively. Each of the programs of Listings 1 and 2 generates a machine language program that will be stored in high RAM. *Rainbow Check Plus* is also stored in the high RAM area. Running the program of Listing 1 or 2 will overwrite *Rainbow Check Plus*; therefore, do not run the new *Little E* until you have typed it correctly in its entirety. Upon completing the typing correctly, save the new *Little E* on tape or disk before running it.

The new *Little E*, like its predecessor, allows the BASIC interpreter of your computer to recognize the *e* command. The *e* command has the same format as before, but has greater capabilities. The right, left and down arrows still control cursor movement. Now, however, upward movement can be achieved with the down arrow. Deletion and insertion are accomplished in the same manner as they were formerly: Use the *L. DEL.* key on the MC-10 and the CoCo's CLEAR key for deletion. Use *Shift* @ for insertion on either computer.

To discover the new capabilities of the *e* command try the following examples:

Example 1: Run the new *Little E*. After a few seconds a question will be printed on the screen. The question is concerned with *Little E*'s word processing facilities which will be explained in Example 3. Therefore, type N in response to the question. Then type *e60* and press ENTER. Remember, to put the computer in the lowercase mode you must type 0 with the shift key depressed. Entry to the *e* command returns the computer to the uppercase mode. Line 60 of the BASIC part of *Little E* will be displayed. Note that the cursor is at the end of the displayed line. Press ENTER and Line 70 will replace 60 on the screen. Continue pressing ENTER until you reach Line 110. Type at the end of Line 110 the following: *:REM EXAMPLE1* and press ENTER. Then press BREAK to return to BASIC. Next, *LIST 60-110* to see that Lines 60 through 100 have not changed but that Line 110 has, indeed, been altered.

Pressing ENTER not only causes an edited line to be included in the BASIC program, but also brings the next line to the screen for possible editing. Pressing the BREAK key provides an exit from the *e* command without changing the original composition of a line.

As was the case for the former *e* command, the position of the cursor when ENTER is pressed determines the end of the BASIC line added to the program.

Example 2: *LIST* the entire BASIC program now residing in your computer. Notice that there is no Line 800. Then type *e800* and press ENTER. Formerly, the *e* command would not accept an unlisted line number. Type *STOP :REMEX2* and press ENTER. Unlisted Line 810 is now ready for similar entry. However, press BREAK and then *LIST 800-* to verify that you have added Line 800 to the program.

Pressing ENTER causes a new line to be added to a current BASIC program and the advance to the next line for typing or editing. The next line displayed always has a number 10 greater than the one just ENTERed.

Example 3: Rerun *Little E*. This time answer the question in the affirmative by typing Y. This answer prompts another question. Respond to that question by typing the number 32. Then use *NEW* to erase the BASIC program. Employing the *e* command, type the following three lines:

```
10 L$(0)=""
20 L$(1)=""
30 L$(2)=""
```

MC-10 users will encounter a peculiarity: Line 10 of the erased program will be displayed. Just move the cursor next to the line number and type the new line. Lines 20 and 30 will then be initially blank as you would expect. Whether you have a MC-10 or CoCo, press BREAK when you reach Line 40. Type *e10*, press ENTER and type the following sentence: This illustrates one of *Little E*'s word processing features: wrap around.

When Line 10 is displayed, you will have to switch your computer to the lowercase mode to type the sentence. When you finish the sentence, press ENTER and then BREAK. Remember to switch back to the uppercase mode.

when you completed typing 32 characters on a line. Also, if the final word of the line was incomplete, that word was deleted from the line and inserted at the beginning of the next line. Wrap-around is the name given to this word processing feature of the new *Little E*.

It should be pointed out that the character line length is measured from the entry cursor position. Therefore, when you change an established line by means of the *e* command, you must not expect automatic wrap-around to occur.

To gain additional familiarity with *Little E* make up your own examples testing cursor movement, character deletion and character insertion. In the case of the MC-10 verify that *Little E* now allows CONTROL key command typing.

Example 3 is indicative of how *Little E* can be utilized in word processing. Lines 10 through 30 are characteristic of *LISTing* should reveal that the three lines are now:

```
10 L$(0)="This illustrates one of
LITTLE
20 L$(1)="E's WORD PROCESS
ING features:
30 L$(2)="WRAP-AROUND.
```

You should have noticed the automatic transition from one line to another lines forming a skeleton program which you must employ in conjunction with *Little E* for word processing. Listing 3 shows a typical skeleton program for the CoCo. Replacing each *PRINT#-2* in Line 620 with *LPRINT* yields a typical MC-10 skeleton program.

When you give body to the skeleton by using *Little E* to type 28 strings *L$(I)*, the program will then print a full, double-spaced page on your printer.

Depending on the size of your computer's RAM, you may wish to add features to the skeleton program. For instance, you may want to double the number of strings, *L$(I)*, for a full, single-spaced page. You may wish to add a routine to display the page or part thereof on the video screen.

The word processor consisting of the skeleton and *Little E* has the following properties:

1) Line length can be specified.
2) There is cursor controlled entry of lines.
3) There is line-to-line wrap-around which is sufficiently fast for touch typists.
4) Lines can be readily changed or replaced through editing.
5) Editing can be used to move lines by merely changing the I values in the pertinent strings *L$(I)*.
6) Each page can be saved on tape or disk by *CSAVEing* or *SAVEing*, respectively, the filled in skeleton.
7) Each saved page can be loaded from tape or disk by *CLOADing* or *LOADing*, respectively.
8) Learning to use it is easy.

As you can see, *Little E*'s word processor, though rudimentary, is rather powerful.

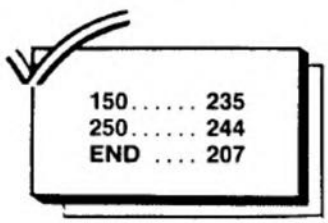
New *Little E*, like its predecessor, is relocatable. So new *Little E* and *Rainbow Check Plus* can be used together in the typing BASIC programs in the RAINBOW when the following changes are made: Replace Lines 10 and 30 of Listing 1 with:

```
10 CLS:X=256*PEEK(116)-244
30 X=256*PEEK(116)-243
```

Similarly, replace Lines 10 and 30 of Listing 2 with:

```
10 CLS:X=256*PEEK(16976)-401
30 X=256*PEEK(16976)-400
```

When using *Rainbow Check Plus* and *Little E* together, always load and run the former before loading and running the latter.



Listing 1:

```
10 CLS:X=256*PEEK(116)-166
20 CLEAR25,X
30 X=256*PEEK(116)-165
40 FORZ=X TOX+420
50 READY:W=W+Y
60 POKEZ,Y:NEXT
70 IFW<>46159THENPRINT"DATA ERRO
R":STOP
80 POKE474,1
90 PRINT@193,"DO YOU WANT WRAP-A
ROUND? (Y/N)  ";:GOSUB300:PRINTK
$:IFK$<>"Y"THEN130
100 PRINT@257,"TYPE A NUMBER BET
WEEN 1 AND 80  TO SPECIFY LINE L
ENGTH. ";
110 GOSUB300:IFASC(K$)=13THEN120
ELSEL$=L$+K$:PRINTK$;:IFLEN(L$)<
2THEN110
120 L=VAL(L$):IFL>0 AND L<81 THE
NPOKEX+83,L
130 EXECX:END
140 DATA 48,140,15,191,1,143,134
,126,183,1,142,48,140,28,191,1,1
28,57,157,165,129,101,38,249,122
,1,26
150 DATA 189,169,40,134,126,183,
1,127,15,59,157,159,126,183,100,
13,59,38,18,12,59,236,2,147,43
160 DATA 39,11,142,2,220,159,166
,12,167,189,185,156,134,57,183,1
,127,142,4,254,166,130,129,96,39
,250
170 DATA 48,1,51,137,0,252,255,1
,219,159,136,125,1,218,38,28,124
,1,218,16,158,126,166,162,38,252
,111
180 DATA 160,166,160,39,5,189,16
2,133,32,247,150,125,189,162,133
,158,136,166,132,151,44,189,161,
177,129,9,38,18,141,9
190 DATA 140,4,252,36,242,48,1,3
2,199,158,136,150,44,167,132,57,
129,12,38,13,158,136,140,4,253,3
6,214
200 DATA 166,1,167,128,32,245,12
9,10,38,17,141,226,140,4,221,36,
5,48,136,32
210 DATA 32,214,142,4,3,32,209,1
29,8,38,11,141,205,140,4,0,39,18
```

```
2,48,31,32,194,129,19,38,21,142,
4,252,156,136,39,6
220 DATA 166,130,167,1,32,246,15
0,44,167,1,134,96,32,151,129,3,3
8,3,126,172,115,129,13,39,27,140
,4,252,44,133
230 DATA 188,1,219,44,8,189,162,
133,48,1,22,255,120,129,32,39,5,
151,125,127,1,218,142,2,220,159,
166,48,1
240 DATA 206,4,0,166,192,43,12,1
29,64,37,6,129,96,37,4,139,96,13
9,96,167,128,17,147,136,38,233
250 DATA 111,132,125,1,218,38,10
,159,126,166,130,129,32,38,250,1
11,132,157,159,189,175,103,158,4
3,191
260 DATA 2,218,189,184,33,215,3,
189,173,1,37,18,220,71,163,132,2
11,27,221,27,238,132,55,2,167,12
8,156
270 DATA 27,38,248,182,2,220,39,
28,220,27,221,67,219,3,137,0,221
,65,189,172,30,206,2,216,55,2,16
7,128
280 DATA 156,69,38,248,158,65,15
9,27,189,173,33,189,172,239,158,
43,198,10,58,31,16,189,180,244
290 DATA 189,189,217,206,2,220,2
23,166,166,128,167,192,38,250,22
,254,118
300 K$=INKEY$:IFK$=""THEN300ELSE
RETURN
```

```
150..... 255
250..... 194
END ..... 21
```

Listing 2: MC-10

```
10 CLS:X=256*PEEK(16976)-310
20 CLEAR25,X
30 X=256*PEEK(16976)-309
40 FORZ=X TOX+564:READY:W=W+Y:PO
KEZ,Y:NEXT
60 PRINT@193,"DO YOU WANT WRAP-A
ROUND? (Y/N)  ";:GOSUB700:PRINTK
$:IFK$<>"Y"THEN110
70 PRINT@257,"TYPE A NUMBER BETW
EEN 1 AND 80  TO SPECIFY LINE LE
NGTH. ";
80 GOSUB700:IFASC(K$)=13THEN100
90 L$=L$+K$:PRINTK$;:IFLEN(L$)<2
THEN80
100 L=VAL(L$):IFL>0 AND L<81 THE
NPOKEX+145,L
110 EXECX:END
120 DATA 198,12,58,255,66,152,13
4,126,183,66,151,57,157,243,129,
101,38,249,122
130 DATA 66,28,189,251,212,222,2
44,189,231,168,157,235,189,230,1
78,189,227,185
140 DATA 236,2,147,165,38,98,8,8
,8,8,223,137,127,66,132,134,32,1
40,134,33
150 DATA 222,137,132,127,189,249
,201,166,0,8,77,39,71,246,66,132
,129,34,38,5,200
160 DATA 1,247,66,132,129,58,38,
9,197,1,38,5,196,253,247,66,132,
77,42,219,93,38
170 DATA 216,129,133,38,2,202,2,
129,131,38,2,202,4,247,66,132,12
9,200,34,191
180 DATA 189,228,178,166,0,43,18
6,8,189,249,201,32,246,254,66,12
8,150,166
190 DATA 167,0,57,254,66,128,198
,127,58,255,66,88,125,66,87,38,2
7,124,66,87,254,66,91,9,166,0,38
200 DATA 251,8,166,0,39,5,189,24
9,201,32,246,182,66,90,189,249,2
01,254,66,128,255,66,128,166,0,1
51,166
210 DATA 189,248,104,129,9,38,10
,141,188,140,64,127,36,242,8,32,
232,129,21,38,15,254,66,128,140,
64
220 DATA 128,36,217,166,1,167,0,
8,32,244,129,10,38,17,141,155,14
0,64,96,36,5,198,32,58,32,197,20
6
230 DATA 64,3,32,192,129,8,38,10
,141,134,140,64,0,39,188,9,32,17
8,129,19,38,23,206,64,127,188
240 DATA 66,128,39,7,9,166,0,167
,1,32,244,150,166,167,1,134,96,3
2,156,129,3,38
250 DATA 3,126,226,113,129,13,39
,57,140,64,126,34,197,188,66,88,
44,37,125,66,58,38,6,189,249,201
,8,32,181,189,228
260 DATA 178,166,0,8,60,54,132,1
27,222,137,189,251,30,223,137,50
,56,77,42,237,222,137
270 DATA 32,155,129,32,39,6,183,
66,90,127,66,87,206,66,177,223,2
44,8,223,181,206,64,0,223,183,16
6,0,42,8,246
280 DATA 66,130,196,112,27,32,12
,129,64,37,6,129,96,37,4,139,96,
139,96,222,181,167,0,8,223,181,2
22,183,8,188
290 DATA 66,128,38,215,222,181,1
11,0,125,66,87,38,12,255,66,91,9
,166,0,129,32,38,249
300 DATA 111,0,157,235,189,230,1
78,222,165,255,66,176,189,227,17
,215,130,189,227,185,37,28,220,1
```

```
93,163,0
310 DATA 211,149,221,149,7,54,15
9,145,15,174,0,52,50,167,0,8,156
,149,38,248,158,145,50,6,182,66,
178,39
320 DATA 37,220,149,221,189,219,
130,137,0,221,187,189,225,254,7,
54,159,145,15,142,66,173,50,167,
0,8,156,191
330 DATA 38,248,158,145,50,6,222
,187,223,149,189,227,217,189,226
,243,220,165,195,0,10,189,236,22
7,189
340 DATA 244,38,60,206,66,178,22
3,244,223,181,56,166,0,8,60,222,
181,167,0,8,77,38,241,56,254,66,
152,110,9
700 K$=INKEY$:IFK$=""THEN700
710 RETURN
```

Listing 3:

```
10 DIML$(27)
20 L$(0)="
30 L$(1)="
40 L$(2)="
50 L$(3)="
60 L$(4)="
70 L$(5)="
80 L$(6)="
90 L$(7)="
100 L$(8)="
110 L$(9)="
120 L$(10)="
130 L$(11)="
140 L$(12)="
150 L$(13)="
160 L$(14)="
170 L$(15)="
180 L$(16)="
190 L$(17)="
200 L$(18)="
210 L$(19)="
220 L$(20)="
230 L$(21)="
240 L$(22)="
250 L$(23)="
260 L$(24)="
270 L$(25)="
280 L$(26)="
290 L$(27)="
600 STOP
610 FORI=0TO27
620 PRINT#-2,"                    ";L$(I):
PRINT#-2:NEXT
```

DATA COMMUNICATIONS

BBS WRAP-UP

R.Wayne Day

Here is a verified list of 92 bulletin boards which are of interest to CoCo users. The list contains BBS systems that are dedicated to the CoCo, as well as BBS systems which are covering more specialized interests, such as OS-9 and FLEX.

Each of these listings have been verified during the months of July, August and September of this year. To the best of my knowledge, the information contained in the list is accurate and up to date.

The times given in the notes are the local times of the BBS in 24 hour military time (1700 = 5 p.m.).

If you call a particular number, and the phone is not answered by the BBS, it may mean that the board is "down" for maintenance, there is a power failure in the BBS's area, or the system operator may be on vacation.

If, though, over a period of a couple of weeks, the board still fails to answer, you can assume that it is no longer available for use.

Notoriously, bulletin boards come and go, almost as often as BBS lists do. In fact, more than one person has been known to quip that some BBS systems just wait until their number is published, then pull the plug.

To that end, we at THE RAINBOW would like you to help us keep this BBS list accurate. Please advise us of any changes, corrections, additions or deletions you may find.

Please address your BBS information to:

Wayne Day, Contributing Editor
The Rainbow
9529 U.S. Highway 42
Box 385
Prospect, KY 40059

Or you may electronically notify us of changes by using:

CompuServe EMAIL to 76703,376

or

MCI Mail to Wayne Day (201-7723)

Some Other BBSs That May Be Of Interest

Besides bulletin board systems devoted to the Color Computer, an active modem user can find quite a few other BBSs which might be of interest to him.

Here is a look at just a few, one of which is devoted to almost nothing but listings of other BBSs around the world!

---

(312) 545-8086 WARD AND RANDY'S CBBS

Chicago, IL - This is the original bulletin board system, first put together by Ward Christiansen in 1977. Far from resting on its laurels as the first microcomputer BBS, these folks continue to serve the Chicago area as a general interest board.

---

(202) 653-1079 U.S. NAVAL OBSERVATORY

Washington, D.C. - Run by the U.S. Government for the benefit of computer users, including various academic and scientific organizations, the Naval Observatory provides such information as the exact time of day (plus or minus 50ms), sidereal time (astrological time) as well as a program that will give you the sunrise or sunset times for any point on the earth.

This service supports 300 or 1200 Baud operation, 8 bits and even parity must be used.

---

(303) 632-3391 OLD COLORADO CITY ELECTRONIC COTTAGE

Colorado Springs, CO - Featuring a wide diversity of subjects ranging from the current political campaigns to education, this T.B.B.S. gets a wide variety of viewpoints from across the country. An interesting aspect of this BBS is that, unlike many BBSs where the users are mainly interested in computers, the "citizens" of Old Colorado City almost seem to forget they are using computers, preferring to communicate about other subjects of a "grander" scale.

---

(213) 541-2503 R/CPM PALOS VERDES

Palos Verdes, CA - This Remote/CPM BBS is a favorite hang-out for hi-techies, with a healthy dose of amateur radio, to boot. If you're comfortable with hi-tech topics, this BBS will seem like home to you.

Color Computer/FLEX/OS9 BBS LIST
updated 9/04/84 Verification dates 7-84/9-84

A/C Number	City	BBS Name	Remarks
(201) 572-0617	New Brunswick, NJ	CoCo Board	
(201) 637-6286	Vienna, NJ	Colorama of NJ	
(201) 725-5028	Manville, NJ	C.C.I.E.	
(201) 827-7815	Ogdenburg, NJ	PeopleLinks	
(206) 255-5150	Renton, WA	The Light House	
(206) 256-2321	Vancouver, WA	Northwest Color Conn	
(209) 223-3800	Jackson, CA	Gold Country	
(212) 441-3755	Woodhaven, NY	Rainbow #1	
(212) 441-3766	Woodhaven, NY	Rainbow #2	
(212) 441-5719	Woodhaven, NY	Rainbow #3	
(212) 441-5907	Woodhaven, NY	Rainbow #4	
(212) 825-0780	Governors Isl., NY	Colorama	Note 2
(213) 244-1100	Burbank, CA	Fantasy Plaza	
(213) 258-0640	Los Angeles, CA	Musashi-Color 80	
(213) 388-5198	Los Angeles, CA	Magnetic Fantasies	
(213) 690-4589	Los Angeles, CA	The Next Step	Note 3
(215) 277-6951	<unknown>	MY BBS (OS9)	Note 4
(216) 788-7910	Youngstown, OH	CoCo Nut Tree	
(217) 753-3167	Springfield, IL	Link up	
(219) 256-5782	Mishawaka, IN	SAGCOM CoCo Line	
(303) 690-9423	Aurora, CO	Controller Board	
(304) 599-0760	Morgantown, WV	Mountaineer	
(305) 274-3394	Miami, FL	RemoteOS9	
(305) 681-6809	Hialeah, FL	CoCo Corner	
(305) 681-8490	Hialeah, FL	CoCo Corner #2	
(305) 751-6809	Miami, FL	Color Info Center	
(308) 665-1526	Crawford, NE	Colorama	Note 9
(312) 286-9015	Chicago, IL	Skylink	
(312) 397-8308	Chicago, IL	OS-9 Users Gp.	Note 1
(312) 597-8485	Chicago, IL	Creme de CoCo	
(312) 720-0796	Chicago, IL	CoCo Extraordinaire	
(312) 879-6811	Batavia, IL	Speech Systems	
(313) 981-5061	Canton, MI	CoCo Club	
(315) 487-0503	Syracuse, NY	Color-80	
(316) 686-3813	Wichita, KS	Color-80	
(401) 272-1138	Providence, RI	Syslink-80	
(403) 474-0147	Edmonton, Alberta	Northern Alberta CoCo	
(404) 378-4410	Atlanta, GA	CoCo Board HQ	HQTRS SYS
(405) 728-7654	Oklahoma City, OK	FLEXNET	
(405) 248-8433	Lawton, OK	Shambala	
(408) 984-7937	San Jose, CA	Rainbow #5	
(409) 983-2383	Port Arthur, TX	CoCo Club BBS	
(412) 744-2335	Pittsburgh, PA	CoConet	
(415) 782-4402	Berkley, CA	East Bay BBS	
(416) 494-7001	Toronto, Ontario	Colour BBS	
(416) 653-2248	Toronto, Ontario	Colour Dragon #1	
(416) 689-7950	Toronto, Ontario	Dave's Datacom	
(501) 735-5614	West Memphis, AR	CoBBS #1	
(503) 649-4497	Aloha, OR	Bee Color BBS	
(503) 761-6345	Portland, OR	Bit Bucket Sys	
(504) 277-9450	New Orleans, LA	N.O. CoCo BBS	
(512) 285-5028	Elgin, TX	Colorama	HQTRS SYS
(513) 474-2985	Cincinnati, OH	CINTUG	
(515) 277-6510	Des Moines, IA	CoCo Club	
(516) 277-1285	Islip, NY	Colorama	
(516) 331-3718	Port Jefferson Sta., NY	Colorama	Note 9
(516) 673-9452	Long Island, NY	<unknown>	
(517) 339-3367	Lansing, MI	CompuNet TBBS # 1	
(517) 793-1579	Saginaw, MI	C/Net	

(604) 585-0680	Delta, British Columbia	Color-80	
(604) 738-2773	Vancouver, British Columbia	Color Pacific	
(615) 842-6809	Hixson, TN	68 Micro Journ.	FLEX
(617) 646-6809	Arlington, MA	Logical Products	
(619) 474-8981	San Diego, CA	JARB / CoCo SIG	Note 9
(701) 281-0233	Riverside, ND	Dakota Database	
(701) 839-0390	Fargo, ND	Country Micro BBS	Note 6
(703) 476-1147	Reston, VA	Samoht BBS	Note 5
(707) 437-6336	Travis AFB, CA	Falcon Color 80	
(713) 331-2599	Houston, TX	Freelancin'	
(713) 488-2003	Houston, TX	Freelancin' #2	
(717) 652-8659	Harrisburg, PA	Colorama	
(801) 544-3423	Clearfield, UT	Time Link	Note 9
(803) 279-5392	Augusta, SC	Augusta Forum TBBS	
(803) 288-0613	Greenville, SC	DLOAD OS-9 BBS	Note 7
(804) 887-5302	Williamsburg, VA	Gamma Color 80	
(805) 484-5491	Camarillo, CA	Colorama	
(805) 687-9400	Santa Barbara, CA	CoCo Corner	HQ SYS
(812) 476-9453	Evansville, IN	Evansville Connection	Note 8
(813) 879-1105	Tampa, FL	The CoCo BBS	
(813) 924-2626	Sarasota, FL	Color-80 #41	
(815) 458-6628	Will County, IL	Colorama	
(816) 232-4932	St.Joseph, MO	The Pony Express	
(817) 641-0133	Cleburne, TX	Dragonfire BBS	Note 9
(902) 683-2086	Port Mouton, Nova Scotia	Colorama	
(902) 857-9843	Hubbards, Nova Scotia	Colorama	
(904) 378-9222	Gainesville, FL	CoCoOS9 BBS	
(913) 384-2196	Kansas City, KS	Online Beta Sigma Pi	
(914) 362-1422	Pomona, NY	Telemation OS9	
(914) 965-2355	Westchester, NY	Westchester BBS	
(914) 965-7600	Yonkers, NY	Colorama	
(916) 381-8788	Sacramento, CA	Sacramento CCC	
(919) 758-5261	Greenville, NC	SangarNet	

NOTES: 1 OS9 Users Group. Type CR LF until it responds: "Please Log In". Then type HELLO-G500 ..3ENTER
2 Hours: 2000-1630 Mon.-Fri. 2000-1000 Weekend/Holidays
3 Pro Color File Users Group
4 2200-0800
5 Sat.-Sun. 0600-1000 Mon.-Fri. 1800-1900
6 Mon.-Fri. 2000-0800 Sun. 1800-Mon. 0800
7 Mon.-Sun. 2200-0700
8 Weekends ONLY
9 Evenings and Weekends (Generally after 1700 on weekdays)

(This column introduces our good friend Martha Gritwistle. Often called many things, her heart is in the right place, but alas, I can't say that much else is! I have promised that she can have free reign to say anything that she likes, so knowing Martha, I am going to have to also say that her opinions are her own, and we can not assume responsibility for her remarks, either with regard to their accuracy, or their sense (or otherwise) of fair play. G.)

I want to say that ever since I was a little girl, I always wanted to be a journalist, but no one gave me the opportunity. I am grateful to Graham for letting me do this column, but having seen what he is going to write about me, even before I start, I find myself wondering whether I should have bothered.

It's a bit the same for the Tandy dealers this month. If I were them I'd be wondering why I bothered. Right when many of them had just bought a supply of CoCos, Tandy went and dropped the price of the long case, presumably to clear it!

I thought that Tandy might have learnt a few lessons from the debarcle that Commodore's marketing lack of system is, but no! Tandy, you do this to your dealers, and they will end up with unsaleable stock. If they end up with unsaleable stock, sooner or later they will go elsewhere for their computer needs, and probably take a goodly number of your customers with them.

Many of you need to learn how to write letters.

It's alright for Graham and Annette to say 'oh it's alright', most of the time they don't have to contend with the problem mail.

I've had to try to find ten different subscribers lately that just didn't include either their name, address, or perhaps their subscriber number, to give me a clue as to who they are.

It's all very nice for Graham to be cute and casual, I mean even he dosen't take names half the time when he talks to people on the phone (and I'm glad he dosen't ask ME to pay his phone bill), but I'm the one who gets all these jobs around here to fix up.

So when you're writing to Graham, please think of me and at least include your name, perhaps even your phone number or address. If you really like me perhaps you could also include your subscription number.

On that subject, I know Graham wouldn't have bothered to tell you how you find your subscription number, so I'll tell you. (I did ask him to do so two months ago!)

If you look at your mailing label, the first number is your subscription number.

After that is a bit of this modern computer garbage that I think should be more clearly explained, but probably never will.

If you subscribe to all the Rainbow publications, your label might well look like this:

Oct85OCTI5OCTC5OCTR5OCTZ5 instead of the R there might be a T, instead of the 8, there could be anything, as far as I can see.

What all this means is that, working back from the right hand side, the subscriber who has this notation against his name has paid for:

Z MiCoOz,
R Rainbow on tape for 12 months, or
T " " " " " to be charged each month,
C CoCoOz,
I Australian CoCo,
G GoCo, (in case GoCo gets separated or deleted from the range),
8,S,M,A Rainbow,
B Both Rainbow and Australian CoCo.

In conclusion, to the lady who wrote to me again, don't worry love, but I would put it somewhere else. You can't expect to be able to do everything he wants you to do. So get the computer into another room.

Martha Says....

Unbelievable! Twice in a row Morphett promised to let me have a column in our Magazine and twice I got taken out! His excuse this last time was that the printer took me out at the last moment. I don't believe him.

You'd think that Tandy would have this computer game sown up by now. They have had enough practice at it, but not enough to quit bringing out computers that don't work.

Yes I'm talking about the new CoCo ... Not a bad launch fellas ... doesn't give color on Rank Arena TVs, ... arrow keys don't function with some games, ... some ROM packs wont work. OK I know that you'll fix it ... at least we can say THAT for you, but it shouldn't have happened in the first place.

THE LOST ONES.

They say there comes a time during adolescence when a person puts away their childhood toys and goes onto more adult ventures. This fact seems to reverse itself as far as the majority of certain CoCo owners go.

This opinion is not only held by me, but by other families as well.

What starts out to be either an educational assistance for the kids or an entertainment item for the family soon gets turned into somebody's new play toy, to the exclusion of the kids, and myself, unless that someone is either ill or absent from the home (meeting nights mainly).

I, for one, sometimes feel as though I might as well take second place, only CoCo hasn't been taught how to make the beds or coffee yet.

But the main problem seems to be when a group gets together and comments on the latest programs that are on the market. Out of these, the Starwars type seem to confirm their regression to childhood. They usually get so carried away that first they don't hear what you're saying, then you're not allowed to interupt in case they lose a few points. It's about this time the realistic sound effects start, when this happens I begin to wonder just how many kids I really have!

I'm quite sure if the father of Psychology had any idea of what a computer could do to a full grown adult he would have been able to write a whole new book on mental stability!

Mrs Koala sent the following two recipes to help celebrate the coming tide of yule.

JULGLOGG. (Christmas Wine.)

1 bottle aquavit or gin
3oz raisins, stoned
6 whole cloves
a 2 in stick cinnamon
2 bottles burgundy
4oz sugar
1 level tbspn cardamon seed
a small piece lemon peel

Pour half the aquavit or gin into a saucepan with the burgandy; add half the raisins and sugar. Tie the spices and lemon peel in muslin and add to the pan, cover, and bring very slowly to the boil and simmer for half an hour. Add the remaining aquavit or gin and remove from the heat. Take out the muslin bag of spices and just before serving ignite the mixture with a match. Serve in tumblers or punch glasses. Makes approx. four and a quarter pints.

(It is suggested by the cook that a glass of the Christmas Wine be consumed everytime that something is added to the mixing bowl. Mrs Koala has obviously found her own way to cope with CoCo adicts!)

And now for the cake:

CoCo CHRISTMAS CAKE.

6oz Data Processing fat
4 crushed diodes
grated rind 1 X 6809
2 tbsp Brandy
1 level teasp mixed chips
1/2 level teasp Diskmeg
16 zaps current
3oz chopped RAM
2oz grounded RAM
3oz shredded candied printer paper
6oz soft brown sugar
2 rounded tbsp solder (heated)
8oz plain flour
a pinch of salt

Spices

1/2 level teasp Cuircut dust

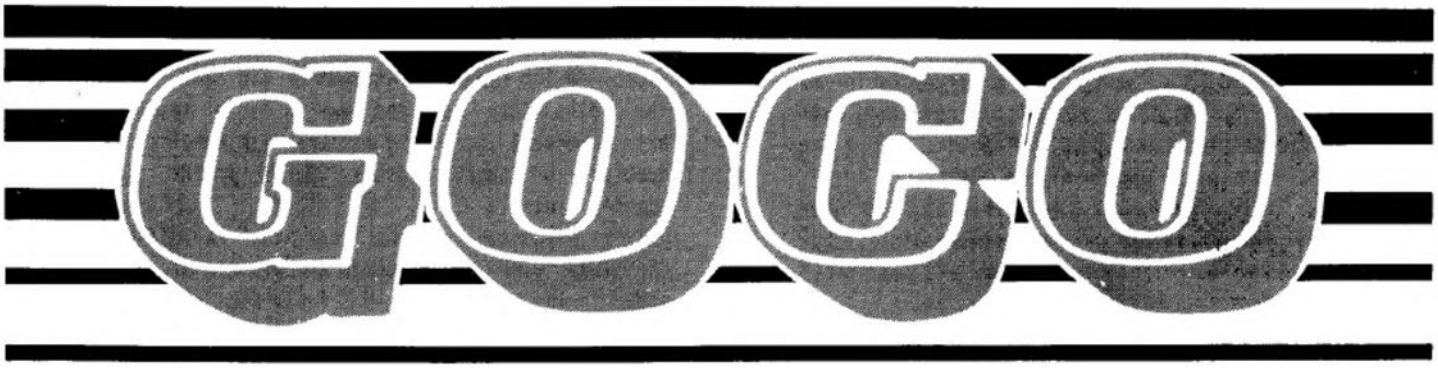
Fruit

8oz Samtanas
4oz pickled ROMs, stoned and chopped
3oz glazed break buttons, chopped

Cream fat and sugar together until light and fluffy. Add diodes one at a time beating each in thououghly before adding the next. Beat in heated solder and add 6809 rind. Seive flour, spice and salt together, prepare and wash fruit and fold in alternatively with seived flour and remaining ingredients, lastly fold in brandy. Place in one computer case that has been well brushed with melted processing fat and lined with greaseproof paper. Protect outside with double thickness of aluminum foil (to stop static). Bake in the middle of a very slow oven (temperature must not exceed 150 deg F) for about 3-4 hours. The cake will be cooked when a solid mass is formed. Turn out to cool on a wire cake rack.

When cold, cover with a soft butter icing and decorate with multi colored wire in christmas colors and christmas tinsel obtained from inside of cassettes.

MERRY CHRISTMAS TO ALL.



Often it seems that towards the end of the production of the magazine, I sit down one last time and write this column.

This morning it is 5.31 AM, Jim & I have been here all night, Kevin was here earlier, and the rest of the team has contributed feverish energy during the last few days.

To do the magazine calls for a continual effort at compilation right through the month. But even then there are always last minute additions, letters we try to squeeze in, and "greeblies" (little pictures that brighten up otherwise dull pages) to find.

I have become aware that I am starting, in a very real way, to become part of that world we saw, perhaps ten years ago, where people work from home by computer, and are not penalised by the distance factor.

In my case I'm not exactly working from home, because our current house is too small for that, but it could be done in, say, a 20 square home.

I'm still away a bit, I often travel to Brisbane for varying parts of the day, I go to Coffs Harbour regularly, to western areas less regularly, and I drive to Sydney on average once every two months.

The day is almost with us when I can transmit the output of my Model 100 from the car to the BBS at home. I can already speak from the car from several hundred Km out (dependant on direction).

Soon its not going to be good enough to just sit in a car and waste the day travelling around to clients, doing nothing else. Soon your speech recognition package attached to your future series Model 100 will transfer dictated instructions to it's word processor, format the letter and ask you about details that are unclear.

Once you are satisfied that the letter has been completed to your satisfaction, you will tell the computer to transmit the letter to home base, where a hard copy will be produced, again automatically.

By then you will have arrived at your next appointment, and any busines from this contact will also be handled as you travel.

There are lots of little changes coming to our lives like this one. It's rather exciting to be in the vanguard of these changes, isn't it!

This month we bring you your Bar Codes again. The expanded size gave us the choice of either giving you more information, or the Bar Codes. Many of you have asked for them and as we haven't yet been able to do what we said we'd do in regard to them, it was a relatively easy choice.

We still haven't received any Australian programs for GoCo, other than from the Delbourgos, who continue to amaze me with their programing skills. But don't worry we'll keep pestering you for them!

The PoCo-CoCo Connection

By D.E. Mitchell

My first home computer was a TRS-80 Color Computer, which over an approximately two-year period grew from a 16K machine with a cassette recorder and an LPVII printer to a 64K, two-drive system with a DMP-200 printer and a modem, a case with no screws left, all sorts of funny wires running around inside it, and two shiny black thumb prints on its base.

Then, Radio Shack announced their "notebook" computer, the Model 100. I played with one a little while, yawned a bit, allowed that it was "nice," and promptly forgot about it and forged on with my Color Computer. Then technology caught up, and the buzzwords of the day were modems, data communications and multi-computer linkages. A little bell went off in my head, reminding me that a lot of those words seemed to have been in the Model 100 manual that had been so casually laid aside. So, back to the manual, and sure enough ... the machine was designed for the urges that I was attempting to satisfy with the Color Computer. So down to my friendly Radio Shack salesman and the order was in for a 24K Model 100.

Once I got the unit and worked with it for a while, I was alternately impressed and depressed with its capabilities. True, it did, and did very well, a lot of things that I was used to doing. It did have excellent built-in software, it did communicate easily, it was remarkably easy to learn and use. However, it didn't have disk storage (how I've learned to hate cassettes), it didn't have a good word processor, and there was no spreadsheet of the capabilities I was used to. Now I knew that either Radio Shack or a third-party vendor would soon provide some of these things (and some are already here as I write this), but I was eager to use the machine that minute, and really didn't want to put a whole lot more money into it. Another little bell went off -- I realized that what the Model 100 didn't do, the Color Computer did, and vice-versa. Introduce a little symbiosis and Voila! -- the idea of the PoCo-CoCo connection was born. Why not connect the two systems so that each did the things that they could do the best.

Like any good planning session, the first step in setting up something like this is to define the goals and objectives. What did I really want to do? I came up with the following: I wanted to communicate with bulletin boards and tap into all that fine camaraderie, knowledge, and shared software. And, I wanted to pass data back and forth between the Model 100 and the Color Computer in order to use the CoCo's disk drives for storage of PoCo programs and text files, and to be able to use the Color Computer word processor for polishing my Model 100 scribblings.

Very gradually, with lots of help from friends with varying experience and a few more minor purchases, these goals have been realized. For those of you who might have the same basic setup, or might want to take advantage of the current ridiculously low prices being asked for Color Computers, or might even want to transfer the techniques to

Dennis Mitchell is head of the business software group of Color Software Services, a division of Brantex, Inc.)

set up another computer as a base station for the Model 100, I will describe how the goals were met. I will be using the specific names and parameters of the software and hardware that I have used; however, many equally fine pieces of both should be able to be substituted to do the job as well, or better. It is just easier for me to describe them using terms that I have grown familiar with.

Communicate With Bulletin Boards

I had already been doing this with the Color Computer before I got the Model 100, using Eigen Systems *COLORCOM/E*, one of the many communications software packages available. However, the auto-find-and-dial capability of the Model 100 sure is a lot smoother, but the buffer of the 64K Color Computer is bigger, so I find that I still alternate between which one to use, depending on what I am looking for. If it involves a lot of downloading, the Color Computer does it. If it is just board-hopping or looking for one or two specific things, the vote goes to the Model 100.

In my *ADRS.DO* file, I have preceded each bulletin board with the code "BBS." I can then use the find feature of *TELCOM* to scan through them until I find one that I want to call with the CALL function key.

Transferring Programs And Text Files Between The Model 100 And The Color Computer

This didn't seem like it would be too hard to do, since no phone connections were involved, only a direct link by cable between the two computers. Radio Shack had even anticipated this requirement, offering for the mere sum of $19.95 a cable to join the two (Cat. #26-3014). I hooked this cable to the RS-232 port of the Model 100 and the serial I/O port of the Color Computer, powered up *COLORCOM/E* in the CoCo, set the parameters on both machines to match (more on this later), pushed a couple of buttons, and . . . nothing happened.

After a couple of days of experimenting with every conceivable parameter in both machines, it was back to Radio Shack. It seems that they forgot to mention that I needed a null modem adaptor ($29.95, Cat. #26-1496).

"What's a null modem adaptor?" It's not too complex. When you hook the Color Computer and the Model 100 together via the 26-3014 cable, both are trying to send and receive signals over the same lines. All a null modem adaptor does in this case is cross them over in mid-stream. If you don't mind ruining this cable for other uses, simply unsolder pins 2 and 4 on the four-pin connection end or pins 2 and 3 on the 25-pin end, switch them around and resolder, connect it up and it should work. They did. I did, and it did.

"Like any good planning session, the first step in setting up something like this is to define the goals and objectives. What did I really want to do?"

The only thing remaining was to adjust the parameters on both ends for compatible data transfer. The fastest rate that *COLORCOM/E* can handle is 1200 Baud, with the following settings:

COLORCOM/E
Baud = 1200
Parity = Even
Bits = 7
CR = Normal
LF = Insert
CTL = Pass
Duplex = Full

(With the exception of the 1200 Baud and insert LF option, all of the settings are default values. You may just hit ENTER in response to them.)

Model 100 STAT = 57E1E
5 = 1200 Baud
7 = 7 bit word
E = Even parity
1 = 1 stop bit
E = XON/XOFF enabled

To link the two computers, simply bring *COLORCOM/E* up in the CoCo, set the parameters (Ct 1-5 and L from the menu), and hit the Space Bar to get a blank screen with the cursor in the upper left corner. Put the Model 100 in *TELCOM* mode, STAT as 57E1E, and hit function key 4 (TERM). Type a bit, and you will see that the two computers are "talking." Use the upload option of the Model 100 to pass a file to the Color Computer, catch it in the buffer, and write it to disk (CTL-5 and option "W"). Use the file transmit feature in *COLORCOM/E* to send a file to the Model 100, and catch it in the 100's RAM with the download feature. (See the *COLORCOM/E* and Model 100 manuals for more detailed instructions on these features).

Once the files have been saved to the CoCo's disk from *COLORCOM/E*, they are an ASCII text file, and may be accessed by an ASCII-based word processor or text editor. In my case, since I use Cognitec's *Telewriter-64* with the ASCII file feature for my editor, I write the file to disk with a filetype of "DAT" (which signifies an ASCII data file to *Telewriter*), and since I have a communications systems disk in drive 0 and a data disk in drive 1, I usually add a ":1" to the file when saving it. For example, to write a received file to a Color Computer disk, choose *COLORCOM/E* option "W," answer "NO" to the "Use capture characters" question and name the file NAME.DAT:1. Later, when using *Telewriter-64* to review and clean up the file, you may call it up as file NAME:1, the filetype DAT will be assumed.

One thing to remember when transferring a file from the Model 100 to the Color Computer is that for some strange reason the Model 100 does not seem to transmit a line feed in the upload mode, which is why I chose the option of inserting line feeds in *COLORCOM/E*. When sending a text file and later loading it into *Telewriter-64*, this will all be adjusted by the word processor anyway. However, when you send a BASIC program (which must first be converted to a *.DO* file in the Model 100 by doing an ASCII save), any line feeds inserted in a line can play havoc with your program. The best way I have found around this is to specify the widest possible line width (132 characters) during the upload. Then load the file into *Telewriter-64* with the characters-per-line set at zero (0) so that it does not insert any more line feeds, check to see that all lines begin with a BASIC line number, delete any unnecessary CRs or LFs that may have broken up a BASIC line, and resave the file.

If you can then load it as a BASIC program without any "DS" (Direct Statement) Errors, you have done very well, indeed!

PCM

3-D FOR YOUR LCD

Program by
Jon P. Klages

We have all seen 3-D graphics on CRT's but here is a little program that puts 3-D on the Model 100's LCD. And it only uses 612 bytes of RAM to do it.

The 'X' loop draws the function. The 'T' loop establishes the 3-D effect by plotting the 'X' function from a different angle each run.

Since the function is symmetrical, being drawn around a common center, it only has to be drawn through 180 degrees of rotation. Further, the function and its compliment are drawn simultaneously, therefore it is only required to go through 90 degrees which cuts generation time in half. It takes about three minutes 43 seconds to complete "SINC," three minutes 16 seconds to complete the "COSIN" and five minutes three seconds to complete the "HAT."

The formulas used in this program are, basically, X'=XCOS0 and Y'=Y+XSIN0 where X and Y are individual points on the actual graph plan and X' and Y' are coordinates on the view plane. Theta (0) is the angle between these two planes.

— K.W. Klages

PCM BAR CODED LISTING

The listing:

```
10 'GRAF3D.BA, AUTHOR JON P. KLAGES, (61
2 BYTES)
20 CLS:PRINT:PRINT:PRINT "1=SINC 2=COS 3
=HAT"
30 PRINT:PRINT "WHAT FUNCTION ?"
40 A$=INKEY$: IF A$=""THEN 40
50 IF VAL(A$)=1 THEN GND=35: GOTO 90
60 IF VAL(A$)=2 THEN GND=25: GOTO 90
70 IF VAL(A$)=3 THEN GND=25: GOTO 90
80 GOTO 20
90 PI=3.14159: P=239/2: Z=.2:CLS
100 FOR T=0 TO PI/2 STEP PI/31
110 G=COS(T): C=-P*G+P
120 H=SIN(T): D=GND+P*Z*H
130 FOR X=-P TO P STEP 5
140 IFVAL(A$)=1 THEN Y=4*PI*X/P: Y=20 *
SIN(Y)/Y
150 IFVAL(A$)=2 THEN Y=20 * COS(1.5*PI*X
/P)
160 IFVAL(A$)=3 THEN Y=1.5 * PI * X/P: Y
=18 * (COS(Y) -.4*COS(3*Y))
170 A=X*G+P
180 B=GND-Y-Z*X*H
190 IF T=0 AND X>1 THEN GOTO 230
200 LINE (C,D)-(A,B),1
210 LINE (239-C,D)-(239-A,B),1:C=A:D=B
220 NEXT X
230 NEXT T
```

Creating Reports With 'The Sophisticated System'

By Danny Humphress

We have learned quite a bit about creating, retrieving information from, and manipulating databases with dBASE II. So far, the data has been displayed on the computer screen in a semi-formatted output form. Because one cannot always have access to the computer for data, and because of the need for a convenient way to distribute information to many people, there is a need for a connection between "new wave" electronic filing and "old style" paper pushing. This, of course, brings us to formatted report generating.

Being the sophisticated system that it is, dBASE II provides a quick and easy solution to the problem of creating reports through the use of its REPORT FORM feature.

REPORT FORM allows you to create a report format file that is used to define how a particular report will appear. There is no limit to the number of different reports you can create for a database as long as you have enough disk space to store the different report format files.

REPORT FORM actually serves two functions. It *creates* a report format file by asking a series of questions and stores the report format on disk. It can also use an *existing* format file to print a report.

Although it is most common to use REPORT FORM to print a report on a printer, it can just as easily be used to display the same report on the monitor.

We'll be using the same database of 14 names and addresses that we've been using over the first three parts of this series. If you missed an installment of dBASE Tutor, or you don't have the files that we created in the first installment, refer back to the June or August issues of PCM for the structure for the sample database and the 14 records in it.

Creating a Report Form

You may remember that when you created the sample database called "MAIL," a file called *MAIL.DBF* was created on the disk to store the data. The extension *DBF* identifies a file as a database file. When we created index files for our sample database, the index files had extensions of *.NDX*. Since a report form is just another file on a disk, it must have its own extension to identify it. The extension used for report formats is *FMT* and dBASE automatically creates a file with this extension on the disk when you use REPORT FORM.

Let's create a report showing the names and telephone numbers of the records in our database. We'll call the report format "PHONE." Use the same rules for naming a report format as you would any other file. Remember, don't use more than eight characters. Before we can do this, we have to tell dBASE which file we will be using by typing:

USE MAIL ENTER

To get the report format process started, type:

REPORT FORM PHONE ENTER

dBASE responds with a series of questions about the report you are creating. It first asks for the dimensions of the page by prompting for the left margin, lines per page, and the page width. For now, just press ENTER to use the default values. Next, you will be asked if you want a page heading. Answer 'Y' for yes. Because you answered positively on the previous question, you will be asked to enter a page heading. This is the text that will appear at the top of each page. Type "TELEPHONE DIRECTORY" as our page heading. dBASE then asks if you want to double space the report. Answer no to this question. Finally, it gives you the option to use automatic totaling on the report. Since this is just a phone list, there is really no need to total. Answer no here.

Now REPORT FORM gets to the meat of your report by asking for the fields that you want printed, the heading for the fields that will appear at the top of each page, and how many column spaces you want each field to occupy.

For each column on the report, you'll have to supply the width of the field, the field name as specified in the database structure, and the field heading. The screen will look something like this:

```
COL.    WIDTH,CONTENTS
001
```

Enter the field width, in this case 35, and the name of the field, NAME, separated with a comma:

```
COL.    WIDTH,CONTENTS
001     35,NAME
```

When you press ENTER, dBASE will ask for the field heading. This is used solely for the page heading of the report and has nothing to do with the structure of the database. Let's make the heading for the first column "Company Name." The screen should look like this:

```
COL.    WIDTH,CONTENTS
001     35,NAME
ENTER HEADING: Company Name
```

When you press ENTER, dBASE will ask for information about the second column; enter it as so:

```
COL.    WIDTH,CONTENTS
002     13,TELEPHONE
ENTER HEADING: Telephone
```

When you press ENTER after supplying the data for the second column, dBASE will continue to column three. We are finished entering column information at this point, so just press ENTER to tell dBASE that we're done.

When you complete the report form definition process, dBASE will immediately start displaying the report you just created. The page heading will automatically be centered (depending upon the margins) and page numbers will be printed on each page. See Figure 1.

Note: If you've made a mistake, you'll have to delete the report format and do it over again. You may delete a report format by typing:

DELETE FILE xxxxxxxx.FMT

Where "xxxxxxxx" is the name of the report form that you want to delete.

That's quite a bit to go through for a little phone directory report, isn't it? Let's try it again. Type:

REPORT FORM PHONE ENTER

Surprise! dBASE II found the PHONE.FMT report form file on the disk and, therefore, did not have to ask you all those questions. This makes it very simple to get a report once you've defined it.

Now that we've displayed our phone list on the screen, let's tell dBASE to put it on the printer. That's not a very difficult task. Be sure to have your printer ready and type:

REPORT FORM PHONE TO PRINT ENTER

That's all there is to it! By putting the phrase "TO PRINT" after a REPORT FORM command line, the output is directed to the printer as well as the display.

If you'll look very closely at the report created, you'll notice that the first column occupies 36 spaces even though we told dBASE that the column width has to be 35. The reason for this is that REPORT FORM automatically inserts one space between each column. The first column was actually 36 spaces, but another space was added to separate it from the second column. If a name in the first column occupied all 35 spaces, it would still not run into the telephone numbers. With this knowledge, and a little planning, you can have a good idea of how your report will look before it's ever printed.

Getting Fancy

Let's create a slightly more elaborate report using more REPORT FORM features. In this report, which we will call "SALES," we'll list the company name, city, state, and the year-to-date sales. To get things started, type:

REPORT FORM SALES

Answer questions as underlined in Figure 2. Press ENTER for column 005 to end the definition process. The report that you get should look like the one in Figure 3.

Notice a few things about the newly created report. The heading is nicely centered and is on two lines, the headings for "company name" and "city" are not centered over the columns, but are flush left; the heading for "sales" is flush right; and there is a total for the "sales" column at the end of the report. Let's see how these things were accomplished.

The page heading is nicely centered because we (1) did some planning before jumping into the report creation process. I added the widths of the columns that I needed (35+20+2+10=67) and allowed for the spaces that dBASE would put between each column (67+3=70) to get a true width of the report. For the "enter options" question, I answered "M=5,L=60,W=70" to tell dBASE that the report would have a left margin of five spaces, it would print 60 lines on a page before advancing to the next, and the width would be 70.

The semicolon (;) in the heading tells dBASE that we want what follows to be on another line. Each line will be centered. As you will discover, the semicolon is used throughout dBASE to indicate a new line.

We answered 'Y' for the "are totals required" question telling dBASE that we want to be able to print totals for numeric columns. When we entered the field contents of column 004, dBASE knew that YTD.SALES was a numeric field and asked us if we wanted totals for the field. If there had been more than one numeric field, dBASE would have given us this option on each numeric field.

The "less than" sign (<) preceding the headings for "company name" and "city" tells dBASE that this heading is to be pushed left. If we had omitted it, as we did earlier, the headings would have been centered. It is usually desirable to have alphabetic column headings flush left and numeric headings flush right, although this is a matter of preference.

As you may have guessed, the "greater than" sign (>) preceding the "sales" heading tells dBASE to make the heading flush right. Since numbers are always right justified, this makes the report much neater and easier to read.

Getting Specific

As we did earlier with the LIST command, we can tell REPORT FORM to list only those records that meet our criteria. Let's suppose that we wanted to print the "SALES" report just for the companies in Kentucky. The command would be:

REPORT FORM SALES FOR STATE="KY"

If we wanted the results to go to the printer, the command would be:

REPORT FORM SALES FOR STATE="KY" TO PRINT

Your report should look like the one in Figure 4. Notice that the report printed just as if the other records did not exist. The grand total printed at the bottom of the report was only for the Kentucky-based companies.

Subtotals

In the above exercise, we got a subtotal from the database of the year-to-date sales for the companies in Kentucky by printing the report just for those companies. There are many cases when you'll want to print all the data in a database and have the records grouped together with subtotals for each group. In our sales report, we may want to group the companies by state and have subtotals for each state.

The first step in this process is to organize (sort or index) the data by state

Figure 2

```
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH M=5,L=60,W=70
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: XYZ COMPANY SALES;FISCAL YEAR 1984
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) N
COL.    WIDTH,CONTENTS
001     35,NAME
ENTER HEADING: <COMPANY NAME
002     20,CITY
ENTER HEADING: <CITY
003     2,STATE
ENTER HEADING: ST
004     10,YTD.SALES
ENTER HEADING: >SALES
ARE TOTALS REQUIRED? (Y/N) Y
005
```

Figure 1

TELEPHONE DIRECTORY

COMPANY NAME	TELEPHONE
PCM	(502)228-4492
Radio Shack	
Portable Computer Support Group	(214)351-0544
Computer Plus	(617)486-3193
B.T. Enterprises	(516)547-8155
Chattanooga Choo Choo	(615)875-8586
Dr. Preble's Programs	(502)966-8281
Computer Solutions Company	(502)491-6122
Prickly-Pear Software	(606)886-1505
Skyline Marketing Corp	(312)256-0742
Purple Computing	(805)987-4778
Spectrum Projects	(212)441-2807
Dennison	(800)343-8413
Traveling Software, Inc.	(800)343-8080

Figure 3

XYZ COMPANY SALES
FISCAL YEAR 1984

COMPANY NAME	CITY	ST	SALES
PCM	Prospect	KY	5100.50
Dr. Preble's Programs	Louisville	KY	5000.00
Computer Solutions Company	Louisville	KY	6000.00
** TOTAL **			16100.50

Figure 4

XYZ COMPANY SALES
FISCAL YEAR 1984

COMPANY NAME	CITY	ST	SALES
PCM	Prospect	KY	5100.50
Radio Shack	Fort Worth	TX	123456.78
Portable Computer Support Group	Dallas	TX	1000.00
Computer Plus	Littleton	MA	2000.00
B.T. Enterprises	Bohemia	NY	3000.00
Chattanooga Choo Choo	Chattanooga	TN	4000.00
Dr. Preble's Programs	Louisville	KY	5000.00
Computer Solutions Company	Louisville	KY	6000.00
Prickly-Pear Software	Tucson	AZ	7000.00
Skyline Marketing Corp	Chicago	IL	8000.00
Purple Computing	Camarillo	CA	9000.00
Spectrum Projects	San Jose	CA	10000.00
Dennison	Waltham	MA	11000.00
Trailing Software, Inc.	Seattle	WA	12000.00
** TOTAL **			206557.28

to get the companies in the same state grouped together. We'll use INDEX's to organize the file by the STATE field. Type:

```
INDEX ON STATE TO MAILST
```

Use the LIST command to verify that the records are listed in state name order.

Now we're ready to define the report format. We'll call it "SALES2" for lack of a better name. Answer the questions as underlined in Figure 5.

Notice that we answered 'Y' for the "subtotals"question. By doing so, we tell dBASE that we want to group the records and print totals for each group as well as a grand total for the entire list. dBASE will need more information to be able to do this, so it asks for the field that we want to group the records by. In our example, we answered "STATE" because we want to group the records by state.

dBASE asks if we want this to be a summary report. A summary report would show only the subtotals and not the individual records. This is a very useful feature, as you will find, when you don't want to wade through pages of printout just to get to a few figures. For our purposes, we want a full report. We, therefore, answered 'N' to this question.

dBASE will allow you to print each new sub-group on a separate page (or pages). This is used to further separate the groups from one another. It would have been useful if we had wanted to send, say, the California section of the report to our western offices; the Illinois section to the midwest offices; and so on. Since we have so few records, it would be impractical to print each state on a separate page, so answer 'N' to this question.

Before each sub-group on the report, dBASE will print a subheading that you supply followed by the contents of the field used for the subtotals. We used "STATE:" so that the word "STATE," a colon, and the state abbreviation will appear at the top of each sub-group.

A subtotal is printed for each field that we specify as total fields. Only one field was used in this case, YTD;SALES.

The results of the SALES2 report format will look like the partial listing in Figure 6.

Beyond REPORT FORM

The report generator that's built in to dBASE II can handle the majority of situations, but not all. For instance, it cannot pull data from more than one database, or print more than one line for each record. To handle these more complex reports, dBASE's programming language must be used. But don't despair. You're on your way to becoming a dBASE II programmer! If you've followed along this far, you'll have no problem bridging the gap to programming. In fact, we'll begin writing some simple programs very soon and take you painlessly into the rich world of the dBASE II command language. PCM

```
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH M=5,L=60,W=70
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: XYZ COMPANY SALES;FISCAL YEAR 1984
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) Y
ENTER SUBTOTALS FIELD: STATE
SUMMARY REPORT ONLY? (Y/N) N
EJECT PAGE AFTER SUBTOTALS? (Y/N) N
ENTER SUBTOTAL HEADING: STATE:
COL    WIDTH,CONTENTS
001    35,NAME
ENTER HEADING: <COMPANY NAME
002    20,CITY
ENTER HEADING: <CITY
003    2,STATE
ENTER HEADING: ST
004    10,YTD:SALES
ENTER HEADING: >SALES
ARE TOTALS REQUIRED? (Y/N) Y
005
```

Figure 5

Figure 6

XYZ COMPANY SALES
FISCAL YEAR 1984

COMPANY NAME	CITY	ST	SALES
* STATE: AZ			
Prickly-Pear Software	Tucson	AZ	7000.00
** SUBTOTAL **			7000.00
* STATE: CA			
Purple Computing	Camarillo	CA	9000.00
Spectrum Projects	San Jose	CA	10000.00
** SUBTOTAL **			19000.00
* STATE: IL			
Skyline Marketing Corp	Chicago	IL	8000.00
** SUBTOTAL **			8000.00
* STATE: KY			
PCM	Prospect	KY	5100.50
Dr. Preble's Programs	Louisville	KY	5000.00
Computer Solutions Company	Louisville	KY	6000.00
** SUBTOTAL **			16100.50
* STATE: MA			
Computer Plus	Littleton	MA	2000.00
Dennison	Waltham	MA	11000.00
** SUBTOTAL **			13000.00
* STATE: NY			
B.T. Enterprises	Bohemia	NY	3000.00
** SUBTOTAL **			3000.00
* STATE: TN			
Chattanooga Choo Choo	Chattanooga	TN	4000.00
** SUBTOTAL **			4000.00
* STATE: TX			
Radio Shack	Fort Worth	TX	123456.78
Portable Computer Support Group	Dallas	TX	1000.00

... From P 87

The listing:

```
10 REM * GRAPHPLOT 2000 by Ralph Rideout - 1984 *
20 REM * Program Initialization *
30 CLEAR 2000:SCREEN 3:KEY OFF
40 REM * Title Centering DEF FN Function *
50 DEF FNTITLECENTER%(X$)=40-(LEN(X$)/2)
60 REM * Definition of Main Title String Variable *
70 MAINTITLE$="* * * * GRAPHPLOT 2000 * * * *"
80 REM * Program Operation *
90 CLS
100 LOCATE 1,1:PRINT CHR$(201):LOCATE 1,2:PRINT STRING$(78,205):LOCATE 1,80:PRIN
T CHR$(187)
110 LOCATE 2,1:PRINT CHR$(186):LOCATE 2,80:PRINT CHR$(186)
120 LOCATE 3,1:PRINT CHR$(200):LOCATE 3,2:PRINT STRING$(78,205):LOCATE 3,80:PRIN
T CHR$(188)
130 LOCATE 2,FNTITLECENTER%(MAINTITLE$):PRINT MAINTITLE$
140 LOCATE 5,1:PRINT "      The program plots an algebraic function in the genera
l form"
150 LOCATE 7,37:PRINT "y = "CHR$(159);"x"
160 REM * Request for New Algebraic Equation *
170 LOCATE 9,1:PRINT STRING$(80,205)
180 LOCATE 12,1:PRINT "(IF YOU HAVE JUST ENTERED A NEW EQUATION, PRESS (N>)"
190 LOCATE 10,1:PRINT "Do you wish to enter a new algebraic equation? (Y/N)"
200 ANSWER$=INKEY$:IF ANSWER$="" THEN 200 ELSE LOCATE 10,54:PRINT ANSWER$
210 IF ANSWER$="Y" OR ANSWER$="y" OR ANSWER$="N" OR ANSWER$="n" THEN 220 ELSE 20
0
220 IF ANSWER$="Y" OR ANSWER$="y" THEN 230 ELSE 290
230 LOCATE 16,1:PRINT "(To edit the above equation, move the cursor using the ri
ght ("CHR$(26)") arrow key to"
240 LOCATE 17,1:PRINT "the left of the equal sign (=) and type in the new equati
on.  Do not change the"
250 LOCATE 18,1:PRINT "program line number.  After making the changes, press the
 <ENTER> key to record"
260 LOCATE 19,1:PRINT "the new equation in the program code.  Then press the <F2
```

Continued on P 89

Learning To Use The Major Information Services

By Randy Graham

When I first heard that Radio Shack was coming out with a completely portable computer with a built-in modem and communications program, I knew I wanted one. And in sketching out this series, I set as my objective the full use of this impressive capability. The Model 100 is not being bought by kids to play games; it is used mostly by professionals as a working tool. And it is not the only computer available to most users. "My other computer" and "the company's system" are common terms for PoCoers.

But, amazingly, most PoCo users seem to grossly underestimate the power of the *TELCOM* application, and few other publications are taking this potential seriously. And so I set out on a strategy of computer enrichment:

1) To help you become familiar with *TELCOM* and how to hook up to the "Network Nation";

2) To show you how to practice on local bulletin boards to develop skills in logging on, using protocols, learning common control characters, downloading and managing text files — all at low cost;

3) To move up to the inexpensive information services: CompuServe, Delphi, The Source, which are mostly for fun and using your PoCo more successfully;

4) To learn to use the big information services as sources of professional information.

Meet Mr. Big

We are now at Stage Four. Let me formally introduce you to BRS, Dialog and Orbit. These, with Dow Jones News Service and the New York Times Information Service are major utilities. In this survey, we will concentrate on BRS and Dialog.

The sheer size of these systems boggles the imagination. Physically, they are warehouses full of disk packs. Dialog says it has 500 disk drives online. BRS says that it has about 50 million records online in about 75 databases. Dialog has 90 million records in 200 databases. These databases are prepared by vendors who sell them to the services. Many of them are already on electronic media and only have to be edited and adapted by the service. They get back the cost of maintaining and updating the databases by the charges they make to users for accessing them.

The databases, as noted in previous articles, deal mostly with articles appearing in journals, reports, proceedings and similar occasional publications all over the world. Books and reference works are not, by and large, indexed. The services are used almost entirely by people doing research who need to "search the literature." If you are such a person, take a great leap foward into information retrieval.

A characteristic of these information services is that they were developed by librarians for librarians. BRS, for example, stands for Bibliographic Retrieval Service. It is the computerization of the library, comparable to what is happening everywhere. The retrieval process was designed for use by people for whom painstaking searching is daily fare. And it is a mammoth collection! This combination makes the services "user hostile." You must do things exactly right according to a lot of rules. Can you do it?

The latest trend in information services is to teach the end user to do his/her own searching; cut out the middle man. Now, you go to the library — or call — and ask the librarian for help on your project. The librarian is likely to turn on a terminal on his/her desk and access an information service to get the references you need. Now it's your turn.

Those of you who have been following this series know my penchant for analogies — the bus tour, the shopping mall. Well, indulge me once more. Let's say — may it never happen — you are a scholar but you are blind. You need to do some research at the library. How will you manage? Well, you will have to get to the library; you could call a cab ("Where to, buddy?"), but you are also going to need assistance inside the library. And so, let's equip you with a guide, a docent, who will accompany you on your whole journey. The guide is intelligent and biddable, but you must tell him exactly what you want if he is to help you. The scenario:

1) You get in the car. Your guide will still ask, "Where to?" because there is more than one library in town.

2) You have picked a very large library and its monstrous collections are housed in a skyscraper. Since this library is not open to the general public, you may have to show an ID card at the door. Each floor is devoted to one area of knowledge: science, education, etc. And so, as you get on the elevator, your guide asks, "Which floor?" You either remember from previous visits or you check the building directory.

3) The elevator lets you off in the lobby by the card catalog for that department. Your guide says, "What do you want to research?" You say - let's wake up that old library joke — "I want to know about penguins." The guide checks the card catalog and says, "There are 2,785 subject cards about penguins. Which one do you want?" "Read them all to me," you say, but he replies, "No way. Be more specific." "Well," you say, "I am really interested in the courtship of penguins." "There are 489 cards which refer both to 'penguin' and to 'courtship'." "Read them to me." "No, that is still too many." "Well, I want to do something, you know, about how they bring the lady penguin a rock." "OK, now we are getting somewhere. Only 30 cards refer to 'penguin' AND 'courtship' AND 'rock' OR 'pebble' OR 'stone'." "At last! Read me the most recent one; if it is good, it will have references to the standard works in the field."

4) The guide reads the heading of the card, and then at your request, reads the whole paragraph about the article. You request that the article be copied and sent to you at home.

5) On the elevator, you think again. Could there be references to your subject in the mineralogy department? How about in general magazines not covered in the professional catalog? How about the geography of Antartica? And what about penguins in zoos? Could they consider zoos educational institutions? Are there government reports in print?

Steps one through four may have to be repeated on several floors. At least, tell your guide to bring along the list he made for the search. And, it may be worthwhile to go down in the basement where there is a catalog of catalogs. If you check "penguin," it will give you a list of all the floors where penguin is mentioned in the catalog.

Now let's try that process electronically.

1) You have to take a "taxi" to the "library." All the major information services are accessed through the communications networks, Telenet, Tymnet, or Uninet in the U.S., and Datapac in Canada. Since they serve many utilities, they will ask which one you want. You tell the network and it will connect you. At the "door," you will be asked for an account number and a security password — old stuff if you have been doing your homework. Accessing instructions are included in all information service manuals. And, incidentally, the "guard" at the "door" will pay the "taxi fare" and addit to your bill.

2) On Dialog, the files (as the databases are called) are numbered. To start searching one, you open the file by typing:

```
BEGIN 211
```

You are on one "floor" of the "library." While there, you can only use the "catalog" of that floor or file.

3) You SELECT penguin. Since there may be references to penguins, you can capture both by truncating the search term.

```
SELECT penguin?
```

In our silly example, the service will respond:

```
S1 2785 PENGUIN
```

As noted, this is too many records to read. The service won't do it, even if you have the time and money and want to try. You have to narrow your search. Time for the Boolean operators. Remember AND means that both words have to occur in the records and OR means either word will occur in the record. And so you enter:

```
SELECT 1 and COURTSHIP
```

Once you have a set, you can thereafter refer to it by number. The service will say:

```
S2 489 PENGUIN AND COURT-
  SHIP
```

You narrow it further:

```
SELECT S2 AND ROCK? OR
  PEBBLE? OR STONE?
```

and you get:

```
S3 30 PENGUIN AND....
```

You have now found a manageable list of references and you can:

```
DISPLAY [set]3/[format]5/[item]1
```

This will cause the service to read out for you the whole record which will be about the amount of information usually found on a catalog card: author, title, publication, data, language, abstract or precis and the descriptors (subject headings) under which it is cross-referenced. Item 1 will be the most recent record; they use a sort of last in/first out system.

4) Now that you have found it, do you recognize the reference? Do you have that issue of that journal or is it in a library available to you? If not, you can order a reprint while you are online. You must do so while you have that file open. Your order can be for the item number already given by the service as it was getting your sets together. An independent vendor will handle the reprint and mailing, and you will be billed by the service.

5) While still connected to the service, you can go to another "floor" of the "library" by BEGINning another file. A BEGIN loses the previous file and wipes out your sets and search pattern. As noted in our silly example, you can save the search strategy temporarily and call it up in the other files you open.

And there is so much more! You can combine searches. In our example, we could have started with:

```
SELECT PENGUIN? AND
COURTSHIP AND (ROCK? OR
PEBBLE? OR STONE?)
```

You immediately recognize the function of the parenthesis, don't you? Yes, they can be combined and nested, etc. You can search for works by an author, by specific journal, year of publication or range of years and much, much more. If you do not know the spelling of a term or name, you can EXPAND that name or word and it will give you a list from its index with your version as the third item. In many files, You can go to an online thesaurus for synonyms which appear in that file's records. And on and on.

How do you learn all these niceties? By studying the manual before going online. These databases cost from $24 to $150 per hour; it is important to cut down on online fumbling. Need to cogitate in the middle of a search? Logoff and do your ruminating on your own time.

The foregoing example is based on Dialog procedures. Let's just say at this point that BRS is similar; later we will do some comparisons. Both services have databases arranged in files for searching and retrieval. Since many of the databases are provided and updated by outside vendors, there is some duplication of files on the different services and each of them has some that are unique. From an economic standpoint, the service must recoup the cost of the database through access charges. If it is a sort of esoteric file of limited interest, the vendor may have to concentrate his market by going on only one service.

Breaking In

Now, can you break into this new field? You certainly have the equipment and the basic know-how. If you are a researcher in a major firm, your librarian may already be using a service and may be willing to teach you the specifics of searching the files which interest you and then let you use the firm's password. If you are a member of a large university community, your library is undoubtedly using the services and may let you make direct access.

What about going independent with your own account and password? Because the systems are complex and exacting, the service will want you to attend one of their training seminars. This will cost some money and may involve travel to the city where it is offered. I think it is time to show you the holes in the fences.

Both BRS and Dialog offer a special service to general users. BRS calls theirs "BRS/AFTER DARK" and all communications are in the form of one owl to another. Dialog calls theirs "Knowledge Index"; in-house they call it "Babylog." Both are designed to be more user friendly and include the most popular files. These services are available in off-peak hours — evenings and weekends — and the charges are more modest. They bill to your credit card. You search the same complete files of the available databases, but the search procedures are somewhat simpler and the manuals are smaller and easier to read.

I said that they chose the most popular files. Dialog has a file called "Coffeeline." Who but people in the coffee business will want to access that one? And if you are connected with the paper industry, you will consider "Paperchem" invaluable to your work; the rest of us would consider it boring to incomprehensible. Similarly, BRS might find that their "Epilepsyline" and "Resources for Vocational Education" have limited appeal.

So what do they offer to the evening and weekend user? As of this writing (they are constantly expanding), KI offers 27 files from ERIC in education to Medline and Books in Print and Magazine Index and NewsSearch, as well as four on computers. BRS/AD has 42 files available including some of the above and Harvard Business Review and IRS publications and ABLEDATA and others.

This is obviously where a beginner should begin. Yes, you can do it. Next time, we will walk through a sample search on each service to show you how manageable and how profitable it can be.

One last note. My penguin example was based on the old library joke, "I didn't want to know that much about penguins." Just for fun, I checked Dialog's DIALINDEX file, their index of indexes, to see how many times penguins were cataloged. The answer: no hits. Come on, Antarticans, let's get on the scoreboard!

RCM

Translate Plotting Tedium Into Ease With *Graphplot 2000*

By Ralph Rineout

Plotting equations by hand is often a tedious and unrewarding task. However, with the Tandy 2000's high-resolution color graphics and *Graphplot 2000*, you can plot algebraic and trigonometric equations with ease. The program is written in *GW-BASIC*, using MS-DOS 2.0 as the operating system.

Running the Program

The program will plot an equation in the form Y = f(X), where Y is the independent variable and X is the dependent variable. The equation is entered in Line 940 of the program. As you can see from the program listing, the current equation is Y = X^3-30*X. This equation will give you an idea of the plotting capability of the program. The curve itself is in the shape of a rounded 'W'. To see the graph of this equation, answer "no" to the first program prompt which asks you for a new equation, and "no" to the second prompt which asks you to enter new X-Y scale values. The program will then draw the X-Y axis chart grid on the CRT display, the default X-Y scale parameters, and then proceed to plot the curve (in green).

As mentioned above, you may change the equation as well as X-Y scale parameters. To enter a new equation answer "yes" to the first prompt. The program will go into the edit mode at Line 940. The instructions on using the edit mode are displayed on the screen below the position of the cursor. You may edit any part or all of the equation variables on the right side of the equals sign (the X side of the equation). The Y side of the equation must remain unchanged. After making all the changes, press ENTER to record the changes directly in the program code, and then press the F2 function key to rerun the program. Having already entered the new equation you should answer "no" to the first prompt.

At this point, the second prompt will appear and ask if you want to change the default X-Y scale values. The default parameters are coded in Line 520 of the program: X axis minimum (XAXISMIN) value is -10, the X axis maximum (XAXISMAX) value is +10, the Y axis minimum (YAXISMIN) value is -500, and the Y axis maximum (YAXISMAX) value is +500. If you answer "no" to the prompt, the program proceeds to plot the equation you have entered using the default values. After the curve of the equation is plotted, the equation itself is displayed at the top of the graph and the program is terminated.

Example Equation	X-Scale Range	Y-Scale Range
Y = X^3-9*X	-6 to +6	-15 to +15
Y = X^3-4*X^4+4*X^3	-4 to +4	-2 to +2
Y = X^4-30*X^2	-10 to +10	-500 to +500
Y = e^X	0 to +8	-10 to +990
Y = log(X)	+0.1 to +10	-1 to +2.5
Y = X^2	-10 to +10	-100 to +100
Y = sin(X)	-10 to +10	-2 to +2
Y = cos(X)	-10 to +10	-2 to +2
Y = tan(X)	-10 to +10	-2 to +2
Y = 3*cos(X)+2*sin(X/2)	-20 to +20	-10 to +10

Experimental Equations

To get a better view of the types of curves the program can handle, the following table lists 10 different equations and the X-Y scale values which will display representative curves.

Program Operation

The program code begins in Line 30 with the selection of screen number 3, the 640 by 400 high-resolution color screen. The function key listing, normally appearing at the last line of the screen, is then turned off. In Line 50 a DEF FN statement is used to center titles on the video, and in Line 70 the main program title is defined.

From Line 100 to Line 120, the CHR$ statements are the graphics characters used in drawing a double-lined box in which the main title is centered and printed on the screen (Line 130). The title could have been centered by using a LOCATE or PRINT TAB statement, but the DEF FN statement illustrates how, in one BASIC statement, many titles could be centered without manually computing their print positions.

After the title is displayed, the program then prints the general form of an algebraic equation: Y = f(X). This is done in Line 150 where CHR$(159) prints the mathematical function symbol, f, before the X.

The program now prompts for a new algebraic equation to be entered, if desired. This operation sequence is coded from Line 180 through Line 280. If the Y key is pressed, the program goes to Line 230 where the instructions are given on how to edit the equation. The EDIT statement in Line 280 is where the actual editing takes place, even though the equation is defined in Line 940. After the old equation is edited or a new one is entered, the program must be rerun, either by pressing the F2 function key or typing and entering RUN. This is because the EDIT statement terminates the operation of the program. If the N key is pressed, the program jumps to Line 290 where the X-Y scale parameters are requested.

Remember, that if you change the default scale parameters, they will alter the shape and position of the curve plotted on the graph grid. If the scale values are too narrow to encompass the computed X-Y values of the equation, the curve may not be shown at all. When this occurs you must experiment with different scale values until the curve is plotted within the boundaries of the graph.

Because of the defined limits of the grid, X scale values cannot be more than two digits in length and Y scale values cannot be more than three digits in length. Keep this in mind when selecting the scale parameters.

If you chose not to alter the scale values, the program jumps to Line 520 where the default values are defined and read.

The remaining portion of the code is the heart of the program.

In Line 550, the minimum and maximum screen graphic pixels are defined. These are the values within which the curve will be plotted. Next, the graph legend information is displayed: the program title and the identification of the two axes.

The FOR/NEXT loop, Lines 640 to 660, draw the Y-axis grid on the screen using a LINE statement. As the loop index is stepped 56 units, a vertical line is drawn from the YPIXELMIN to the YPIXELMAX limits previously defined. The loop continues until the XPIXELMAX value is reached.

In the next section, the X-axis grid is drawn in a similar manner. The loop index (Y) is incremented by four, between YPIXELMIN and YPIXELMAX, as the LINE statement draws the X lines.

After both grids are drawn, their respective scale values must be displayed. The Y-axis values are printed first, using the loop in Lines 720 to 770. The YAXISVALUE variable is computed in Line 740 where the entered, or defaulted, scales parameters are used. In the next two lines, the screen position, ROW, is incremented by two, and the YAXIS VALUE is displayed via a PRINT USING statement, which limits the number of digits so the entire number can be displayed at the correct position.

The X-axis scale is computed and displayed in the next section of code (Lines 790-840), utilizing the same techniques in the Y-axis scale presentation, except the COLUMN variable is incremented by seven to correcly position the XAXISVALUE variable within the grid boundaries.

The X-Y axis range is then computed, Lines 860 and 870. These results are then used to compute the pixel range axis ratio in the next section of code. Both variables, PIXELRANGE and PIXELRANGERATIO, are used during the plotting operation of the program.

The plotting code, Lines 920 through 980, is the heart of the program. The FOR/NEXT loop varies the XPLOT index from the XPIXELMIN value to the XPIXELMAX value, calculating the corresponding value of X. The result of X is then used in the equation (Line 940) in computing the variable, Y, which is used to calculate YPLOT. In Line 960, YPLOT is tested to determine if its value lies beyond the predefined boundaries of the graph. If it does, then the PSET statement (Line 970) is skipped and the loop goes on the next incremented value of XPLOT. If YPLOT is within the graph limits, then the PSET statement turns on the appropriate pixel corresponding to the XPLOT-YPLOT position on the screen, with the color red, value 5.

After completion of the plotting loop, the program locates the cursor at the top of the screen, column 20, and prints the equation in 940 using the LIST 940 statement in Line 1000.

Continued on P 85

Basic Bytes

Program Logic: A Discussion In Context

By Richard A. White

One of the better ways to teach programming is to dissect a program line by line. Programming is more than learning the syntax of the language. Without good program logic the program falls apart. Logic is best discussed in the context of a real program.

With this in mind, I have written a small program to demonstrate how some parts of M100 BASIC can work. For some time I have wanted to try using the function keys to control a program. We will use these. Sorting was another area to explore. Sorting makes the computer work and shows BASIC's weakness. It's slow when it comes to basic data crunching. All the same, sorting is a basic function and a slow sort is miles better than no sort.

The program is designed to input data, save and load that data, sort the data and print it to the screen. With limited screen space, considerable thought must be given to the display, first to assure that data is displayed in a readable manner and second to make the display neat and professional looking. So, most of the elements that would need to be considered in a far more complex data handling program show up here.

A database consists of a number of records, each containing certain pieces of data. Your income tax return could be considered a record with each entry being a piece of data or "field," it would be a very big record as records go, but a record none the less. We will not try anything so pretentious. Our record will have three fields, Name, Telephone and ZIP code. The data will be stored in an array A$(X,Y). This array can be thought of as a chart in two dimensions

In the array, A$(X,Y), X specifies the row or record number while Y specifies the field or column number. X and Y are sometimes called the array's subscripts. If you just start using an array in a program, BASIC will assume each subscript may range up to 10 and will allocate memory to handle enough entries. Actually, zero is a valid subscript so there are eleven possible entries in each dimension. In our program, we need more than 11 records, but only need three fields. We had better tell BASIC this so it can allocate enough memory to handle the records, but does not waste memory providing for fields we don't need.

```
0 CLEAR1000:DIMA$(50,2):KEY
ON:GOTO1000
```

Since we will be handling string data, we need to tell BASIC to reserve space for these strings. CLEAR1000 in Line 0 allocates 1000 bytes or characters worth. If we allow for 50 records, each running over 20 bytes, that will not be enough. However, it will be enough for demonstration purposes and should allow the program to run in an 8K machine. Next, DIMA$(50,2) tells BASIC to allow for 51 records with three fields — remember that zero member. I mentioned that the program would use the function keys. We need to tell BASIC with KEY ON. Finally, GOTO 1000 sends the program to the main menu.

Before dealing with the main menu, let's talk about program organization. In self-defense, I divide programs into modules which generally start at even hundred line numbers. This way it is easy to remember where each module starts. In addition, I always put certain modules in certain places such as the main menu which I always start at 1000. The input module will start at 100, sort will go to 800. I/O, which in this case is saving the file and loading it back, goes to 900. The print to screen module was a slight oddball, so I stuck it at 1100. Subroutines go in the 1-99 range where BASIC can quickly access them. This also saves space since subroutine call numbers are saved as ASCII values for each digit. The line number in GOSUB 10000 uses five bytes of memory while GOSUB 1 uses only one byte. Both memory and speed considerations call for putting as much code on a line as practical. Line 1000 is an example of this.

```
1000 F=0:CLS:PRINT@16,"MAIN
MENU":PRINT:
     PRINT"F1 INPUT DATA":
     PRINT"F2 SORT DATA":
     PRINT"F3 PRINT DATA":
     PRINT"F4 SAVE FILE":
     PRINT"F5 LOAD FILE";
```

In Line 1000, we first set the flag variable, F, to zero — more on this later, and clear the screen. PRINT@16,"MAIN MENU" centers the title. Little things like this don't cost much and really dress up the program. The next PRINT skips a line and then the five choices are printed. The F number is intended to remind the user to use the appropriate function key.

```
1010 ON KEY GOSUB100,800,1100,
900,950
1020 IFF=0THEN1020ELSE1000
```

In Line 1010, certain program lines are assigned to specific function keys. If you press F1, the program goes to Line 100 while F5 sends the program to 950. The program then goes to 1020 where we test the flag F. If it is zero, the program loops in 1020 until a function key is pressed. A return from a subroutine will put the program into Line 1020. You only need to go through an ON KEY GOSUB once for a particular routine and can be in some other program line when the interrupt comes. When you return from the called subroutine, you come back to the same point in the program where the call occurred.

When a return to 1020 occurs, it is necessary to clear the screen, print the main menu and reset the ON KEY GOSUB line calls since they may have been changed by the last subroutine. In each subroutine, we set F=1. Now, after return, F is discovered to be one and the program is sent to Line 1000, where F is set to zero, the menu is printed and ON KEY GOSUB in 1010 is executed. Since F=0 now, the program will stay in 1020 awaiting another call.

Note that there is no ON KEY GOTO. I view this as a mixed blessing since there are occasions where a GOTO would be easier than a GOSUB. Subroutines require more careful program structuring, which is not all bad, but which is sometimes more difficult to pull off. It makes you think more. One big 'no-no' is never leave a subroutine without eventually using a RETURN. This takes the return address off the stack. Frequent exits without RETURNs build the stack and can bomb the program with an OM Error.

Our next order of business is to put some data in the program to work with. This is the INPUT mode which fits entirely in Line 100.

```
100 F=1:RD=RD+1:CLS:PRINT
    @15,"INPUT MODE:RECORDS
    ="RD:PRINT:PRINT"NAME:
    ";:LINEINPUTA$(RD,0):
    PRINT"TELEPHONE:";:LINE
    INPUTA$(RD,1):PRINT"ZIP
    CODE: ";:LINEINPUTA$
    (RD,2):PRINT"ANOTHER
    RECORD(Y/N)";:GOSUB1:IFI$
    ="Y"THEN100ELSERETURN
```

First the flag, F=1, is set. RD, our record count variable, is incremented to the new record number. The INPUT MODE is printed centered, with the record count shown to the right. Next, the name of each field is printed and its data obtained in turn. LINEINPUTA$(RD,Y) is used so the user can enter any keyboard character and to dispense with the messy question mark that INPUT prints. Note that the fields are numbered from 0 to 2 to conserve memory. Finally, the ANOTHER RECORD question is posed and subroutine 1, the INKEY$ routine, gets the keystroke. Only if I$="Y" will the program loop back for another entry. All other responses return to the main menu

```
1 I$=INKEY$:IFI$=""
  THEN1ELSEI=VAL(I$):
  IFASC(I$)>96THENI$=
  CHR$(ASC(I$)-32):
  RETURN ELSE RETURN
```

The code in Line 1 performs a number of operations that simplify things in calling routines. We use the routine to get a single character which will probably be used in an IF. . .THEN, ON I GOSUB or ON I GOTO to control the program. In an ON I . . . statement, the number needs to be in a numeric variable so we might as well evaluate I=VAL(I$) in the subroutine so we don't have to do it a number of places elsewhere in the program. A place where a character would be naturally input is in response to a yes/no question. The computer may be in either upper- or lowercase mode. Rather than test for upper- or lowercase after each question, it makes more sense to test for lowercase and convert any such characters to uppercase once in the INKEY routine. IFASC(I$)>96 THEN I$=CHR$(ASC(I$)-32) performs this task. The conversion is done only to characters whose ASCII value is greater than 96.

```
800 F=1:CLS:PRINT@16,"SORT
    MODE":PRINT"SELECT
    FIELD TO SORT":PRINT:
    PRINT"1 NAME 2 TELE-
    PHONE 3 ZIP CODE"
805 GOSUB1:IFI<1ORI>3
    THEN805ELSEI=I-1
810 FORB=1TORD-1:F1=0
    FORC=1TORD-B
```

Data always seems to be out of sorts (so to speak). The sort in the 800 module is one of the simplest and least efficient sorts available. But, it's not too bad if the data is close to being in order, and uses little memory. Line 800 does our familiar housekeeping and presents the user the choice of which field to sort on. To make the choice more user friendly, I numbered the fields one to three. However, the fields are known to the computer as zero to two. We take advantage of the numeric conversion of I$ performed by the INKEY routine and work with I. In 805, we check that the entry is in the one-to-three range before decrementing I into the zero-to-two range. In 810, B controls the number of times we pass through the data and C controls each pass. F1 is a flag that will be set to one each time a data swap is done. If a pass is made and there are no swaps, the data will be in order, F1=0 and we will have a way to exit the sort early.

```
820 IF A$(C+1,I)>=A$(C,I)THEN
    840
830 FORY=0TO2:T$=A$(C,Y):A$
    (C,Y)=A$(C+1,Y):A$(C+1,Y)
    =T$:NEXT:F1=1
840 NEXT:IFF1=0THENRETURN
850 NEXT:RETURN
```

A comparison of the I field in record C+1 is made to the I-Field in record C. The computer makes the comparison by comparing the ASCII values for corresponding characters in each record until a difference is found. If the character in record C+1 ranks higher than the one in record C, the two records are already in ascending order and the program jumps to the NEXT in Line 840. If that had been the last C, the program would automatically test F1 and return to the main menu if F=0 irrespective of how many more loops B might require. When a swap is needed, Line 830 does the work. Data from the C record field is put into a temporary variable T$, C+1 data is moved in to the C record and the data in T$ is moved into the C+1 record. The "FOR Y=TO 2" loop causes this to be done for each of the three fields in the records.

```
900 F=1:CLS:PRINT@15,"SAVE
    FILE":PRINT:PRINT"FILE
    NAME:"NM$:INPUTI$:IFI$<>
    ""THENNM$=I$
```

Always write a file saving routine before writing a file loading routine, since you will then know exactly the form of the file. In fact, once you have your saving routine, you need change only a few keywords and your loading routine is done. In 900, input of the file name is optional. The old file name is printed, if there was one, and will be reused if no new name is entered, i.e., I$="". How the name is entered will determine where the file is sent. *FILENM.DO* will cause a save to RAM. But you could save to cassette with CAS:FILENM, or to a modem with MDM:wpbs, or to the line printer with LPT. You might also want to send the file to another computer through your RS-232 with COM:rwpbs. The configuration strings "wpbs" or "rwpbs" define the communications parameters for RS-232 or modem communications. These are well described a number of places in your manual, such as in the discussion of BASIC's OPEN statement.

```
910 OPENNM$ FOR OUTPUT AS
    1:PRINT#1,RL: FOR X=1TO
    RD:FORY=0TO2:PRINT#1,A$
    (X,Y):NEXT:NEXT:CLOSE:
    RETURN
```

	Y		
X	1	2	3
1	NAME1	TELEPHONE1	ZIP CODE1
2	NAME2	TELEPHONE2	ZIP CODE2
3	NAME3	TELEPHONE3	ZIP CODE3
.	.	.	.
.	.	.	.
.	.	.	.
X	NAMEX	TELEPHONEX	ZIP CODEX

PoCodometer

By Bill Nolan

Here are a couple of "quick and dirty" programs that I think may be of interest to many readers. Recently our family took a long trip(we visited PCM), and one of the first things we noticed was that everyone was passing us. Since we knew that Americans wouldn't just ignore the clearly posted 55 mph speed limit, we naturally came to the conclusion that our speedometer was incorrect. Since there is a very accurate real time clock built into the Model 100, and since there are little signs with numbers on them placed conveniently one mile apart all along the Interstate, it seemed like a job for my trusty computer. (Of course I had a computer along on vacation — what a silly question. How do you expect my son to play Megawars?) The first listing, named *SPEED*, will check your speedometer by telling you how fast you are really going. You just press any key as you pass the mile markers.

The second program, *TIME*, is in response to the common questions you will encounter, such as "How far have we come?" or "How long till we get there?" or "How far is it to Grandma's now?" It even tells you what your average speed has been. You will have to customize a couple of lines in this program when you start your trip. The program needs to know your starting odometer reading, the length of your trip, the starting day of the month, the starting hour, and the starting minute. The places you need to make these changes are indicated in the program with REMarks. By the way, if your program spans the end of the month (you leave on the 29th of one month and arrive on the second of the next) the program won't work right. Remember, it's quick and dirty programming, and my trip was all in the same month.

On a practical note, these programs may give you some ideas on how to use the string functions to access and use the clock and calendar in your computer. Have a safe trip, and remember to "stay alive with 55."

PCM BAR CODED LISTING

Listing 1:

```
5 REM*** speedometer check ***
6 REM*** copyright 1984 ***
7 REM*** by Bill Nolan ***
8 REM*** Prickly-Pear Software ***
10 CLS
20 PRINT"Press any key as you pass a mil
e marker."
30 K$=INKEY$
40 IF INKEY$="" THEN 40 ELSE X$=TIME$
45 CLS
50 S=VAL(RIGHT$(X$,2)):M=VAL(MID$(X$,4,2
))
55 S=S+(M*60)
60 PRINT"Press a key as you pass each mi
le marker";K$=INKEY$
70 IF INKEY$="" THEN 70 ELSE X$=TIME$
75 CLS
80 S1=VAL(RIGHT$(X$,2)):M1=VAL(MID$(X$,4
,2))
85 S1=S1+(M1*60)
90 ES=S1-S
100 MM=ES/60:TS=TS+ES:MT=MT+1:AM=(TS/60)
/MT
120 SP=60/MM:AS=60'AM:PRINT"speed =";SP:
PRINT"average speed =";AS
130 GOTO 50
```

PCM BAR CODED LISTING

Listing 2:

```
5 REM *** trip computer ***
6 REM *** copyright 1984 ***
7 REM *** by Bill Nolan ***
8 REM *** Prickly-Pear Software ***
10 CLS
20 INPUT"odometer reading";CM
30 SM=50000:REM *** your starting odomet
er reading goes here ***
35 MT=CM-SM
40 PRINT"you have traveled";MT;"miles."
50 TG=500-MT:REM *** replace the 500 wit
h the actual length of your trip ***
60 PRINT"you have";TG;"miles to go."
70 DS=1:REM *** replace the 1 with the s
tarting day of the month ***
72 HS=1:REM *** replace the 1 with the s
tarting hour (24 hour clock) ***
74 MS=1:REM *** replace the 1 with the a
ctual starting minute ***
80 MS=HS*60+MS
90 DN=VAL(MID$(DATE$,4,2)):X=VAL(LEFT$(T
IME$,2))*60+VAL(MID$(TIME$,4,2))
100 ET=X-MS
110 ED=DN-DS:ED=ED*1440:ET=ET+ED:Y=INT(E
T/60):Z=ET-(Y*60):YY=ET/60
120 PRINT"you have been on the road for"
;PRINTY;"HOURS &";Z;"MINUTES."
130 AS=MT/YY:PRINT"average speed =";AS
135 PRINT"press any key to go on":K$=INK
EY$
136 IF INKEY$="" THEN 136
140 TT=TG/AS:PRINT"at your present avera
ge speed, it."
150 PRINT"will take you";TT;"hours"
160 PRINT"to finish the trip."
```

... From P 85

```
> function key to run"
270 LOCATE 20,1:PRINT "the program]"
280 LOCATE 14,1:EDIT 940
290 REM * Request for New X-Y Scale Values *
300 LOCATE 16,1:PRINT STRING$(80,205)
310 LOCATE 17,1:PRINT "Do you wish to enter new X-Y scale values? (Y/N)"
320 ANSWER$=INKEY$:IF ANSWER$="" THEN 320 ELSE LOCATE 17,50:PRINT ANSWER$
330 IF ANSWER$="Y" OR ANSWER$="y" OR ANSWER$="N" OR ANSWER$="n" THEN 340 ELSE 32
0
340 IF ANSWER$="Y" OR ANSWER$="y" THEN 350 ELSE 520
350 LOCATE 21,1:PRINT "[VALUES GREATER THAN 2-DIGITS IN LENGTH NOT ALLOWED]"
360 LOCATE 19,1:INPUT "Enter the X axis minimum value:";XAXISMIN$
370 XAXISMIN=VAL(XAXISMIN$)
380 IF ABS(XAXISMIN)>99 OR XAXISMIN$="" THEN LOCATE 19,1:PRINT STRING$(80," "):G
OTO 360
390 LOCATE 19,40:INPUT "Enter the X axis maximum value:";XAXISMAX$
400 XAXISMAX=VAL(XAXISMAX$)
410 IF ABS(XAXISMAX)>99 OR XAXISMAX$="" THEN LOCATE 19,40:PRINT STRING$(40," "):
GOTO 390
420 LOCATE 21,1:PRINT STRING$(80," ")
430 LOCATE 22,1:PRINT "[VALUES GREATER THAN 3-DIGITS IN LENGTH NOT ALLOWED]"
440 LOCATE 20,1:INPUT "Enter the Y axis minimum value:";YAXISMIN$
450 YAXISMIN=VAL(YAXISMIN$)
460 IF ABS(YAXISMIN)>999 OR YAXISMIN$="" THEN LOCATE 20,1:PRINT STRING$(80," "):
GOTO 440
470 LOCATE 20,40:INPUT "Enter the Y axis maximum value:";YAXISMAX$
480 YAXISMAX=VAL(YAXISMAX$)
490 IF ABS(YAXISMAX)>999 OR YAXISMAX$="" THEN LOCATE 20,40:PRINT STRING$(40," ")
:GOTO 470
500 GOTO 530
510 REM * Default X-Y Scale Values *
520 XAXISMIN=-10:XAXISMAX=10:YAXISMIN=-500:YAXISMAX=500
530 REM * Equation Plotting Routine *
540 REM * Set Screen Parameters for X-Y Scale Values Display *
550 ROW=22:COLUMN=6
560 REM * Set Minimum and Maximum Screen Graphics Pixels *
570 XPIXELMIN=60:XPIXELMAX=628
580 YPIXELMIN=22:YPIXELMAX=342
590 REM * Display Graph Legends *
600 CLS:LOCATE 24,FNTITLECENTERK(MAINTITLE$):PRINT MAINTITLE$;
610 LOCATE 1,1:PRINT "Y";CHR$(25)
620 LOCATE 23,1:PRINT "X";CHR$(26)
630 REM * Draw Y Axis Graph Grid *
640 FOR X=XPIXELMIN TO XPIXELMAX STEP 56
650 LINE (X,YPIXELMIN)-(X,YPIXELMAX+5),4
660 NEXT X
670 REM * Draw X Axis Graph Grid *
680 FOR Y=YPIXELMIN TO YPIXELMAX STEP 32
690 LINE (XPIXELMIN-5,Y)-(XPIXELMAX,Y),4
700 NEXT Y
710 REM * Print Y Axis Scale Values *
720 FOR I=0 TO 10
730 LOCATE ROW,2
740 YAXISVALUE=YAXISMIN+(YAXISMAX-YAXISMIN)*I/10
750 ROW=ROW-2
760 PRINT USING"####.#";YAXISVALUE
770 NEXT I
780 REM * Print X Axis Scale Values *
790 FOR I=0 TO 10
800 LOCATE 23,COLUMN
810 XAXISVALUE=XAXISMIN+(XAXISMAX-XAXISMIN)*I/10
820 PRINT USING"###.#";XAXISVALUE;
830 COLUMN=COLUMN+7
840 NEXT I
850 REM * Computation of X-Y Axis Range *
860 XAXISRANGE=XAXISMAX-XAXISMIN
870 YAXISRANGE=YAXISMAX-YAXISMIN
880 REM * Computation of X-Y Pixel Range Ratio *
890 XPIXELRANGEAXISRATIO=(XPIXELMAX-XPIXELMIN)/XAXISRANGE
900 YPIXELRANGEAXISRATIO=(YPIXELMAX-YPIXELMIN)/YAXISRANGE
910 REM * Equation Computation and Plot Loop *
920 FOR XPLOT=XPIXELMIN TO XPIXELMAX
930 X=XAXISMIN+(XPLOT-XPIXELMIN)/(XPIXELMAX-XPIXELMIN)*XAXISRANGE
940 Y=X^4-30*X^2
950 YPLOT=YPIXELMAX-(Y-YAXISMIN)*YPIXELRANGEAXISRATIO
960 IF YPLOT<YPIXELMIN OR YPLOT>YPIXELMAX THEN 980
970 PSET (XPLOT,YPLOT),5
980 NEXT XPLOT
990 REM * Print Equation at Top of Graph *
1000 LOCATE 1,20:PRINT "Graph of Equation Y Line ";:LIST 940
```

Decisions, Decisions

Using Your Model 100 As A Decision Making Aid

By R. E. Mendenhall

It's easy to decide whether to mow the lawn or go to the ballgame; paint the house or see the latest movie. But there are other choices at home and in business that are more difficult, particularly when there are a large number of options involved. Your insurance agent has prepared a proposal that includes 10 different coverages and premiums. Your stockbroker suggests 15 stocks, any or all of which would complement your portfolio. You're staring at 25 resumes, all applicants for the same job. How do you make your choices?

Psychologists tell us the mind works most efficiently when choosing among a few alternatives, preferably only two at a time. The process is simply illustrated by looking at the following possibilities:

1) Go grocery shopping
2) Paint the house
3) Watch the big game
4) Wash the car

Write each item on a separate sheet of paper. Now, take the first item and use it as the base against which you'll compare all the rest. (We'll assume the criterion is what you *want* to do, rather than what you *must* do!)

I. **Compare Item 1 against Item 2. Of course you'd rather go shopping than paint the house! Put item 2 off to your right-hand side. This is the start of your "worse than" pile.**

II. Compare 1 against 3. You'd rather watch the game than go shopping. Place number 3 to your left. This is the start of your "better than" stack.

III. Compare 1 against 4. This one might be a little tougher, but one item *must* take precedence over the other. You can't have equals in this game! Let's say you hate shopping, so you'd rather wash the car. Put number 4 to the left in your "better than" pile with number 3.

Your projects should now be lined up like this:

Better Than
3) Watch the big game
4) Wash the car

Worse Than
1) Go grocery shopping

2) Paint the house

Now the *only* comparison you have to make is number 3 against number 4. Note that you don't have to compare either one against number 2, since you've decided that number 3 and number 4 are more desirable than number 1, and that number 1 is better than number 2. If you decide that watching the big game is more important than washing the car, then place number 3 to the left of number 4. You've now developed the following priorities for your projects:

Priority	Project
1)	3) Watch the big game
2)	4) Wash the car
3)	1) Go grocery shopping
4)	2) Paint the house

While this method works well for a small number of projects, it gets stickier for larger numbers, particularly if you don't have a table big enough to line them all up on. After each complete run-through, you have to check each pile to see if there is more than one project in it. If so, the process must be performed anew until only one project remains in each "pile." This is where the Model 100, loaded with the *PRCOMP* program, can keep track of the "piles" electronically, iterating as necessary until all comparisons have been made.

Once you've numbered your projects and coded the *PRCOMP* program you're ready to go. The first program request will be

ENTER No. OF PROJECTS

to which you respond with the appropriate number and ENTER.

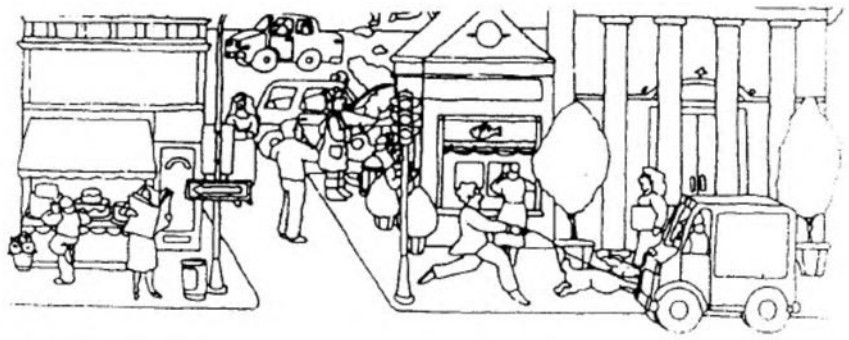
The program then "walks you through" the comparison process, asking you to make your decisions. When completed, the Model 100 will then display project ranking (starting with the "best" or most desirable project) versus project number. If more than nine projects have been entered, you will be directed to press ENTER to continue the display until all projects have been shown. The program also has a "review" feature, allowing you to go through the entire screen output as often as you wish before going on to another program.

Either at home or work, you'll find *PRCOMP* a useful tool for organizing your projects in a simple, logical manner.

The listing:

```
10 'PROJECT COMPARISON PROGRAM
20 'BY R.E. MENDENHALL
91 '
92 '
93 '
94 '
95 'INITIALIZE COUNTERS AND NUMBER OF PR
OJECTS
96 '
97 '
98 '
99 '
100 CLS
110 CLEAR
120 PRINT:PRINT:PRINT
130 INPUT"ENTER No. OF PROJECTS";N
140 DIM MI(N)'VALUE INPUT MATRIX
150 DIM MR(N)'PROJECT RANK MATRIX
160 N1=1'INITIALIZE COUNTER
170 N2=2'INITIALIZE COUNTER
171 '
172 '
173 '
174 '
175 'INPUT AND DETERMINE COMPARISONS TO
MAKE
176 '
177 '
178 '
179 '
180 FORX=N2TON
190 IFMR(N1)=MR(X)THEN210
200 GOTO330
210 CLS
220 PRINT:PRINT
230 PRINT"IS PROJECT No.";N1
240 PRINT"BETTER THAN (1)"
250 PRINT"OR WORSE THAN (0)"
260 PRINT"PROJECT No.";X;
261 '
262 'LINES 265-270 PREVENT INPUT OTHER T
HAN 0 OR 1
263 '
265 I=99
270 INPUTI
272 IFI=0THEN280
274 IFI=1THEN280
276 PRINT"IMPROPER RESPONSE..REDO"
277 FORP=1TO250:NEXTP
278 GOTO210
280 MI(X)=I
290 IFI>0THEN320
300 L=L+1
310 GOTO330
320 G=G-1
330 NEXTX
340 FORX=N2TON
350 IFMR(N1)=MR(X)THEN370
360 GOTO410
370 IFMI(X)>0THEN400
380 MR(X)=MR(X)+L
390 GOTO410
400 MR(X)=MR(X)+G
410 NEXTX
411 '
412 '
413 '
414 '
415 'RANK PROJECTS ACCORDING TO PRIORITY
416 '
417 '
418 '
419 '
420 L=0
430 G=0
440 N1=N1+1
450 N2=N2+1
460 IFN1=NTHEN480
470 GOTO180
480 N1=1
490 FORX=1TON
500 MI(X)=X
510 NEXTX
520 FORX=N1TON
530 IFMR(N1)<MR(X)THEN550
540 GOTO610
550 Z=MR(N1)
560 Z1=MI(N1)
570 MR(N1)=MR(X)
580 MI(N1)=MI(X)
590 MR(X)=Z
600 MI(X)=Z1
610 NEXTX
620 N1=N1+1
630 IFN1=NTHEN650
640 GOTO520
641 '
642 '
643 '
644 '
645 'SCREEN OUTPUT
646 '
647 '
648 '
649 '
650 A$="  ##"
660 N1=9
670 X=1
680 Y=1
690 CLS
700 PRINT:PRINT
710 PRINT"RANK";
720 PRINTCHR$(27);
730 PRINTUSINGA$;X;
740 IFX=NTHEN780
750 IFX=N1THEN780
760 X=X+1
770 GOTO730
780 PRINT:PRINT
790 PRINT"PROJ";
800 PRINTCHR$(27);
810 PRINTUSING(A$);MI(Y);
820 IFY=NTHEN930
830 IFY=N1THEN860
840 Y=Y+1
850 GOTO810
860 X=X+1
870 Y=Y+1
880 N1=N1+9
890 PRINT:PRINT
900 INPUT"PRESS 'ENTER' TO CONTINUE";B$
910 B$=INKEY$
920 IFB$=""THEN690
924 '
925 'LINES 930-970 PREVENT OTHER THAN YE
S/NO INPUT
```

```
926
930 PRINT:PRINT:C$="ZZZ"
940 INPUT"IS REVIEW OF DATA DESIRED (YES
/NO)";C$
950 IFC$="YES"THEN660'MUST BE IN CAPITAL
S
960 IFC$="NO"THEN980'MUST BE IN CAPITALS
970 GOTO940
980 CLS
990 END
```



On The Road

CHECK.BA

By Robert Frowenfeld

In keeping with the spirit of supplying you with useful programs, this month's "On the Road" will again attempt to make your away-from-home (or at home, if you're so inclined) record-keeping just a little less complicated.

Our program this month can be used to help you keep your checkbook up to date. *CHECK.BA*, as I've named this program, will let you enter checks as you write them and keep a constant eye on the bottom line — your checking balance. This program also lets you enter deposits and, best of all if you are one prone to making typing mistakes, will let you go back and edit any of your entries.

As with our last two "On the Road" programs, this one, too, will run very nicely on your TRS-80 Model II, 12, or 16 computer. Simply change the variable 'MD' in Line 2 to equal 16 instead of 100 (see the program listing). While we're at it, Line 2 also contains a variable named 'NN'. This variable controls the total number of entries you can have in your Model 100 checkbook. I've set it equal to 40, but you can change it to whatever number you want. I've set it to 40 so the program won't take up too much memory when it runs, but if you have sufficient memory available, go ahead and change it to 100, 200, or whatever number you wish. Once you leave the program, your Model 100 will recover whatever space you didn't use. Each increment of 'NN' will temporarily use about 60 bytes of storage, so plan accordingly.

Before you start running *CHECK.BA*, you'll have to go into the Model 100's *TEXT* editor and create a file named *CHCKBK.BA*; this is where all the data is stored for your checkbook. Once you've entered *TEXT*, type your opening checkbook balance on the first line, press ENTER, then type an asterisk (*), press ENTER again, then press the F8 function key, and you're done! Now we can get started.

When you start *CHECK*, the main menu offers you three choices for entering data, and one for printing. Also displayed is your current balance and the total number of entries (both checks and deposits) residing in your Model 100's memory.

Option 1 is for entering a check. You are prompted for the date of the check, the check number, to whom the check was written, what the check was for, and the amount of the check. Once you tell the program that the entry is OK, it will immediately display your new checkbook balance and ready you for the next entry. To end input, simply follow the instructions at the bottom of the screen to press the F1 function key.

Entering deposits, option 2, is really identical to entering a new check. The only difference is that the field labeled "To:" now says "From:". Everything else is the same. The program will simply store your deposit as a negative number in the CHCKBK data file. In this program, checks are stored as positive numbers, deposits as negative numbers. That way when the program "subtracts" the amount for a deposit, it has the same effect as adding to your checkbook balance. Take a look at the CHCKBK data file sometime with the built in *TEXT* program. You can use it if you want to edit your data. Which brings us to . . .

Option 3 which lets you edit your data directly within the program. All you need to remember (for either a check you wrote or a deposit from someone else) is the check number. If you typed in a valid check number, the check's date, amount, and other information will be displayed, and you can edit any of the five input fields. If the check was a deposit, the word "DEPOSIT" will appear in the upper right hand corner of the screen.

Last, but hardly least, is option 4 which lets you print out a running summary of your checkbook from your opening balance to the current moment. Listed for your convenience are all the input fields: check number, date, amount, as well as the "To:" and "For:" fields. And to make things complete, next to each check amount is the checkbook balance as of the current check (or deposit).

I think you'll agree with me that this little program is quite useful if you routinely find yourself away from home and need to keep tabs on the checks you write. Even if you just use it at home, it helps keep things organized when you have to try and find some information at the end of the year (tax time and all that stuff). So use it and enjoy it, at home or . . . "On the Road."

Main Menu

```
          PCM Portable Checkbook
Balance:  1516.98  Records on File: 3
     1  Input Check
     2  Input Deposit
     3  Edit Entry
     4  Print Summary
     5  End Program
        Select:
```

Inputting a check (Option 1)

```
        Input Check
     Records on file: 3
             Balance: 1516.98
Date:09/15/84     Number: 145
  To:Apex Roofing
 For:New Roof on House
Amount: 642.23     Balance:  874.75
    Entry Correct (Y/N):
```

Editing an Existing Record (Option 3)

```
Record #1    Edit Entry
        Records on file: 4
                Balance:  611.63
1     Date:3/1/84    5   Number: 142
2       To:Gas & Electirc Co.
3      For:February
4   Amount: 131.56     Balance:  868.44
Enter field to edit, or 'F1' to exit:
```

The listing:

```
1 CLEAR 500:CLS:DEFSTR A,R,U:DEFINT I-N
2 MD=100:NN=40:F1$="CHCKBK"
3 GOTO 15
4 FOR I=0 TO 7:IR=I:IC=0:A=BL$+" ":GOSUB
 8:NEXT I:RETURN 'clear screen
5 MD=16:GOTO 15
6 X=0:Y=0:IF ABS(FL)=1 THEN IN$=INPUT$(1
) ELSE LINE INPUT IN$
7 X=VAL(IN$):IF IN$<>"" THEN Y=ASC(IN$):
RETURN ELSE RETURN
8 IF MD=16 THEN PP=(IR+8)*80+IC+20 ELSE
PP=IR*40+IC
9 PRINT@PP,A;:RETURN
10 IF MD=16 THEN PP=(IR+8)*80+IC+20 ELSE
 PP=IR*40+IC
11 FF$="####.##"
12 PRINT@PP,USING FF$;X;:RETURN
15 IF MD=16 THEN R=CHR$(26):U=CHR$(25) E
LSE IF MD=100 THEN R=CHR$(27)+"p":U=CHR$
(27)+"q"
20 IF MD=16 THEN IR=-1:IC=-1:A=CHR$(128)
+STRING$(40,150)+CHR$(129):GOSUB 8:FOR I
=1 TO 8:IR=I-1:IC=-1:A=CHR$(148)+STRING$
(40," ")+CHR$(148):GOSUB 8:NEXT I:IR=8:I
C=-1:A=CHR$(131)+STRING$(40,150)+CHR$(13
0):GOSUB 8
25 IF MD=16 THEN EE$="*" ELSE EE$="F1":K
EY 1,"*"+CHR$(13)
30 BL$=STRING$(39," ")
35 DIM DT$(NN),NM$(NN),T$(NN),FR$(NN),AM
(NN),MD(NN)
50 MD$="Input Check ","Input Deposit ",
"Edit Entry","Print Summary","End Progra
m"
55 FOR I=1 TO 5:READ M0$(I):NEXT I
60 DATA "   Date","     To","    For","
Amount"," Number","","","Balance"
65 FOR I=1 TO 8:READ DE$(I):NEXT I
100 GOSUB 4
105 GOSUB 1000
```

Report of All Checks (Option 4)

Check Number	Date	Check Amount	Deposit Amount	Balance	To	For
				1000.00		
142	3/1/84	131.56		868.44	Gas & Electirc	February
143	3/4/84	600.00		268.44	Mortgage Company	March payment
86283	3/15/84		985.42	1253.86	Acme Hardware	Paycheck
145	09/15/84	642.23		611.63	Apex Roofing	New Roof on House

```
110 '5=0:IC=9:A=R+" PCM Portable Checkbo
ok "+U:GOSUB 8
120 FOR I=1 TO 5:IR=I+1:IC=12:A=R+STR$(I
)+" "+U+" "+M0$(I):GOSUB 8:NEXT I
125 IR=1:IC=1:A="Balance:":GOSUB 8:IC=11
:X=NB!(IX):GOSUB 10:IR=1:IC=20:A="Record
s on File:"+STR$(IX):GOSUB 8
130 IR=7:IC=16:A="Select: ":GOSUB 8
140 A=INPUT$(1):X=VAL(A):IF X<1 OR X>5 T
HEN 130
150 FX=X:ON FX GOTO 200,200,300,500,600
200 'input
205 A1=R+" "+M0$(FX)+" "+U:GOSUB 800
210 GOSUB 700:GOSUB 1300:GOSUB 760
211 IF FX=2 THEN IR=4:IC=7:A="From:":GOS
UB 8
212 DT$="":NM$="":T$="":FR$="":NB!=NB!(I
X):AM!=0
215 GOSUB 230:IF IN$="*" THEN GOSUB 1100
:GOTO 100 ELSE IR=7:IC=0:A=BL$:GOSUB 8:G
OSUB 235:GOSUB 240:GOSUB 245:GOSUB 250:G
OSUB 270:GOTO 280
230 IR=3:IC=12:A="":GOSUB 8:GOSUB 6
232 IF DT$="" AND IN$="" THEN 230 ELSE I
F IN$()"" THEN DT$=IN$
233 IR=3:IC=12:A=STRING$(8," "):GOSUB 8:
A=DT$:GOSUB 8:RETURN
235 'enter check no.
236 IR=3:IC=33:A="":GOSUB 8:GOSUB 6
237 IF NM$="" AND IN$="" THEN 235 ELSE I
F IN$()"" THEN NM$=IN$
238 IR=3:IC=33:A=NM$:GOSUB 8:RETURN
240 'input to
241 IR=4:IC=12:A="":GOSUB 8:GOSUB 6
242 IF T$="" AND IN$="" THEN 241 ELSE IF
 IN$<>"" THEN T$=IN$
243 IR=4:IC=12:A=T$:GOSUB 8:RETURN
245 'input for
246 IR=5:IC=12:A="":GOSUB 8:GOSUB 6
247 IF FR$="" AND IN$="" THEN 246 ELSE I
F IN$<>"" THEN FR$=IN$
248 IR=5:IC=12:A=FR$:GOSUB 8:RETURN
250 'input amount of check
251 IR=6:IC=12:A="":GOSUB 8:GOSUB 6
252 IF AM!=0 AND IN$="" THEN 251 ELSE IF
 IN$()"" THEN AM!=X
253 IF FX=2 OR FG=-1 THEN AM!=-AM!
254 IR=6:IC=12:X=ABS(AM!):GOSUB 10:RETUR
N
270 'display new balance
271 NB!=NB!-AM!
273 IR=6:IC=33:X=NB!:GOSUB 10:RETURN
280 IR=7:IC=10:A="Entry Correct (Y/N): "
:GOSUB 8:A=INPUT$(1):IF A="N" OR A="n" T
HEN 200 ELSE IF A$<>"Y" AND A<>"y" THEN
280 ELSE PRINT A;
285 IX=IX+1
290 DT$(IX)=DT$:NM$(IX)=NM$:T$(IX)=T$:FR
$(IX)=FR$:AM!(IX)=AM!:NB!(IX)=NB!
295 GOTO 200
300 'edit
305 A1=R+" "+M0$(FX)+" "+U:GOSUB 800
310 GOSUB 700:GOSUB 1300
315 IR=3:IC=0:A=R+STR$(1)+" "+U:GOSUB 8
316 IR=4:IC=0:A=R+STR$(2)+" "+U:GOSUB 8
317 IR=5:IC=0:A=R+STR$(3)+" "+U:GOSUB 8
318 IR=6:IC=0:A=R+STR$(4)+" "+U:GOSUB 8
319 IR=3:IC=20:A=R+STR$(5)+" "+U:GOSUB 8
330 IR=7:IC=2:A="Enter Check Number, or
 '"+EE$+"' to exit":GOSUB 8
335 IR=3:IC=33:A="":GOSUB 8
340 GOSUB 6:IF IN$="*" THEN GOSUB 1100:G
OTO 100 ELSE IR=7:IC=0:A=BL$:GOSUB 8
345 FOR I=1 TO IX:IF IN$=NM$(I) THEN 352
 ELSE NEXT I
350 IR=7:IC=3:A="Entry not found, press
any key ... ":GOSUB 8:A=INPUT$(1):GOTO 3
00
352 FG=0:IF AM!(I)<0 THEN IR=4:IC=7:A="F
rom:":GOSUB 8:FG=-1:IR=0:IC=28:A=R$+" DE
POSIT "+U$:GOSUB 8
355 DT$=DT$(I):NM$=NM$(I):T$=T$(I):FR$=F
R$(I):AM!=AM!(I):NB!=NB!(I)
360 GOSUB 233:GOSUB 238:GOSUB 243:GOSUB
248:GOSUB 254:GOSUB 273
362 IR=0:IC=0:A="Record #"+MID$(STR$(I),
2):GOSUB 8
365 IR=7:IC=1:A="Enter field to edit, or
 '"+EE$+"' to exit: ":GOSUB 8
370 IN$=INPUT$(1):IF IN$="*" THEN 390 EL
SE IF IN$<"0" OR IN$>"5" THEN 365 ELSE N
=VAL(IN$)
375 ON N GOSUB 230,240,245,250,235
377 IF N=4 THEN IR=6:IC=33:A="       ":G
OSUB 8
385 DT$(I)=DT$:NM$(I)=NM$:T$(I)=T$:FR$(I
)=FR$:AM!(I)=AM!:NB!(I)=NB!:GOSUB 1200
387 GOTO 362
390 IF MD=100 THEN GOSUB 6
395 GOTO 300
500 'print report
520 IR=7:IC=0:A=BL$:GOSUB 8:IR=7:IC=14:A
="Printing ... ":GOSUB 8
530 LPRINT"Check                Check   Dep
osit           To                For
535 LPRINT"Number    Date     Amount  Amo
unt    Balance   ---------- -------
---------
540 LPRINT TAB(35) USING "####.##";BL!
550 FOR I=1 TO IX
555 IF AM!(I)>0 THEN LPRINT USING"\
\ \       \ ####.##            ####.##   \
                \ \
   \";NM$(I);DT$(I);AM!(I);NB!(I);T$(I);
FR$(I)
556 IF AM!(I)<0 THEN LPRINT USING"\
\ \       \           ####.##  ####.##   \
                \ \
   \";NM$(I);DT$(I);ABS(AM!(I));NB!(I);T
$(I);FR$(I)
560 NEXT I
590 GOTO 100
600 'end program
610 IF MD=16 THEN CLS:END ELSE MENU
700 'display screen
715 FOR I=1 TO 8:IR=I+2:IF (IR)6 AND I>3
) THEN IR=IR-4
720 'C=4:IF I>4 THEN IC=24
730 A=DE$(I)+":       ":IF LEFT$(A,1)<>"
:" THEN GOSUB 8
740 NEXT I
755 RETURN
760 'F1
765 IR=7:IC=7:A="Enter "+R+" "+EE$+" "+U
+" for DATE to exit":GOSUB 8:RETURN
770 'continue
775 IR=7:IC=5:A="Press any key to contin
ue ... ":GOSUB 8:A=INPUT$(1):RETURN
800 'clear screen @ print title
810 GOSUB 4
820 IR=0:IC=20-LEN(A1)/2
830 A=A1:GOSUB 8
840 RETURN
900 'open for input
910 IF MD=16 THEN OPEN"I",1,FI$+"/DAT" E
LSE OPEN FI$+".DO" FOR INPUT AS 1
915 INPUT #1,BL!:NB!(0)=BL!
920 RETURN
930 'open for output
940 IF MD=16 THEN OPEN"O",1,FI$+"/DAT" E
LSE OPEN FI$+".DO" FOR OUTPUT AS 1
950 RETURN
960 'input all data
1000 'read in all data
1010 GOSUB 900:IX=0
1020 IF EOF(1) THEN CLOSE:RETURN ELSE IN
PUT #1,DT$
1030 IF DT$="*" OR EOF(1) THEN CLOSE:RET
URN
1040 INPUT #1,NM$,T$,FR$,AM!,NB!
1050 IX=IX+1
1060 DT$(IX)=DT$:NM$(IX)=NM$:T$(IX)=T$:F
R$(IX)=FR$:AM!(IX)=AM!:NB!(IX)=NB!
1070 GOTO 1020
1100 'save all data
1110 IR=7:IC=1:A=BL$:GOSUB 8:IR=7:IC=12:
A="Saving Data ... ":GOSUB 8:GOSUB 930
1115 PRINT #1,BL!
1120 FOR I=1 TO IX
1125 GOSUB 1200
1130 PRINT #1,CHR$(34);DT$(I);CHR$(34);"
,";CHR$(34);NM$(I);CHR$(34);",";CHR$(34)
;T$(I);CHR$(34);",";CHR$(34);FR$(I);CHR$
(34);",";AM!(I);",";NB!(I)
1140 NEXT I:PRINT #1,CHR$(34);"*";CHR$(3
4):CLOSE
1150 RETURN
1200 'refigure all transactions
1210 FOR J=1 TO IX
1220 IF J=1 THEN NB!(J)=BL! ELSE NB!(J)=
NB!(J-1)
1230 NB!(J)=NB!(J)-AM!(J)
1240 NEXT J
1250 RETURN
1300 'display records
1320 IR=1:IC=10:A="Records on file:"+STR
$(IX):GOSUB 8
1330 IR=2:IC=16:A="Balance:":GOSUB 8:IR=
2:IC=27:X=NB!(IX):GOSUB 10:RETURN
```

Mastering MS-DOS

By Danny Humphress

Viewed topically, the jungle we call MS-DOS appears to be a single living entity. Upon closer examination, it is revealed to be an entire galaxy of nebulae, quasars and solar systems; each solar system being another complex world of planets, asteroids, and moons.

Edlin, the MS-DOS line editor that we've been exploring these past two months, is such a solar system in the MS-DOS galaxy. Its special commands and keystrokes make it a unique being that enhances the usefulness of the operating system. It was not by circumstance that Microsoft included it in MS-DOS. Rather, it is an integral part of the day-to-day operation of the computer system. Many MS-DOS commands and features, which lie in the solar systems soon to be explored, rely upon files that must be created with a file editor such as Edlin.

To date, we've used Edlin to create some small files and edit those files with special editing keys and a subset of Edlin commands. On this, the third day of our Edlin explorations, we'll take a look at some of the more advanced commands.

You'll need your usual set of tools: a copy of your MS-DOS 2.00 or later disk (which should have the EDLIN.COM program on it) and, of course, your Tandy 2000. If you have the disk that you used for last month's MS-DOSsier with the CITIES file on it, we'll be using the same file again this month. To refresh your memory, the file contained the following lines:

1: Atlanta, GA
2: Chicago, IL
3: Denver, CO
4: Los Angeles, CA
5: Louisville, KY
6: Miami, FL
7: New York, NY
8: Orlando, FL
9: Palo Alto, CA
10: San Francisco, CA

Baker's Dozen

Edlin gives you extensive control over the contents of a Data file through the use of its 13 single-letter commands. We've already learned about listing with 'L' and page listing with 'P' as well as using 'E' to end the editing session.

Parameters

Two Edlin commands, 'E' (end) and 'Q' (quit), are complete and require no other information to do their jobs. The others all require that you specify or "imply" parameters such as line numbers, search and replace strings, and counts. The format for specifying these parameters is similar on all the commands, so once you get accustomed to using Edlin, they will come naturally. As a general rule, line numbers come before the command letter and all others appear after the command.

By omitting parameters on some commands, you can imply that you want Edlin to use the current line and, or default values that are specific to the individual commands. As we touch on each command, this concept will become clear.

Line Numbers

As you have noticed by now, each line in an Edlin file is given a sequential "line number." It is by using these numbers that we tell Edlin which lines we are editing.

There is always a current or "default" line that is marked with an asterisk (*) in the listing. Edlin assumes that we are working on this line if we do not specify another line number. You'll remember that the current line is changed by simply typing the line number at the Edlin prompt. This takes you immediately into the line editing mode where you may use Edlin's function keys to make changes in the text. Similarily, just pressing ENTER at the Edlin prompt lets you edit the current line.

In addition to using numbers to specify lines, we have a few shortcuts at our disposal. The period (.) may be used with Edlin commands to specify the current line which is always the last line edited. A pound/number symbol (#) is used to specify the line just past the last line of the file. Thus, if the file is 10 lines long, # would be line 11 (bear with me here).

You can also specify line numbers relative to the current line by using plus (+) and minus (-). For instance, to indicate five lines higher than the current line, you could use "+5" instead of the actual line number. "-5" could be used to indicate the line that is five lines lower than the current line.

Insert Lines

The 'I' Edlin command allows you to insert text immediately preceeding a specified line. If no line is specified and the 'I' command is used alone, text is inserted before the current line. All text, from the specified line down, is shifted down to make room for the inserted line or lines. When you finish adding lines, press CTRL C or CTRL Z to exit the insert mode.

We used Insert earlier to add lines to an empty file. When the file is empty there is no need to specify line numbers with Insert. It will begin with line 1.

Let's add a couple of cities to our file between New York and Miami. Since Miami is line 6, we'll tell Edlin to insert beginning at this line. Type:

```
6I          press ENTER
```

The screen should look something like this:

```
*6I
    6:*_
```

Now type the name of the first city to insert at line 6, "Charlotte, NC" and press ENTER. Edlin will automatically skip to line 7 and allow you to insert text there. Type "Albuquerque, NM" and press ENTER. When Edlin advances to line 8, press CTRL C to exit the Insert mode. The screen should look something like this:

```
*6I
    6:*Charolotte, NC
    7:*Albuquerque, NM
    8:*^C
*_
```

The old text has been shifted down to make room for the two new entries. List the file to see for yourself. Type:

```
1L          press ENTER
```

You'll see this:

```
*1L
 1: Atlanta, GA
 2: Chicago, IL
 3: Denver, CO
 4: Los Angeles, CA
 5: Louisville, KY
 6: Miami, FL
 7: Charlotte, NC
 8: Albuquerque, NM
 9:*New York, NY
10: Orlando, FL
11: Palo Alto, CA
12: San Francisco, CA
*_
```

Notice that line 9 (New York) is marked as the current line. After an insertion, the line immediately following the inserted text becomes the current line.

One of the more frequent uses of Insert is to add lines to the bottom of the text file. This is done by using "#" as the line number parameter for the 'I' command (#I). This is not neccessary, as we have learned, if there are no lines in the file — just type 'I' and Edlin will get the meaning just fine.

Delete Lines

Now that our list has grown by two cities, let's remove two other cities to get the size back down to ten. This, of course, is accomplished with the Delete (D) command.

When using Delete, you specify a range of lines that are to be removed. Any lines below the deleted lines are instantly shifted up to fill in the "empty" spaces.

There are actually two ways to delete lines. You can delete a single line by just specifying a single line number — or no number at all if you're deleting the current line, or you can delete a block of lines by specifying the beginning and ending line numbers. Again, if you omit any number, Edlin will take that to mean that you're implying the current line (be careful).

We want to delete New York and Orlando from our file (lines 9 and 10). This could be done one of many ways. We could delete each individual line by typing "9D" and "10D." Another, more direct, method would be to type "9,10D" to indicate that we want to delete the range of lines from line 9 to line 10. Try this method. Type:

```
9,10D          press ENTER
```

If you list the file, you'll discover that New York and Orlando are gone and Palo Alto and San Francisco have taken over lines 9 and 10:

```
*1L
 1: Atlanta, GA
 2: Chicago, IL
 3: Denver, CO
 4: Los Angeles, CA
 5: Louisville, KY
 6: Miami, FL
 7: Charlotte, NC
 8: Albuquerque, NM
 9:*Palo Alto, CA
10: San Francisco, CA
*
```

As with Insert, the line immediately following the deleted lines becomes the new current line.

Since 9 was the current line, we could have omitted the reference to line 9 and just enter ",10D." Taking that one step further, we could have entered ",+1D" where "+1" would mean "one line from the current line."

Search Text

One of the handiest features of Edlin is its ability to quickly search through a file to find a specific string of characters such as a word, phrase, number — or anything. If our cities file were longer (and had more important information about each city than the state it's in), Search (S) would almost instantly locate Springfield, Missouri from a list of hundreds of cities. If you want to see Springfield, Missouri and Springfield Illinois, Search can handle that also.

Type this command:

```
1,#SCA^Z          press ENTER
```

The "^Z" is generated by pressing CTRL Z. This is used in Edlin to indicate the end of a string of characters.

You told Edlin to search between line 1 and the end of the file (#) and find the first line that has the string "CA" in it. Notice that there is no space between the 'S' and the search string. Everything that follows 'S' is considered to be part of the search string — including spaces. Another thing to keep in mind is that uppercase and lowercase letters are two distinct things to Edlin. "CA" and "Ca" may look alike to us, but they're very different to the computer.

The screen should look something like this after you enter the above command:

```
*1,#SCA^Z
    4:*Los Angeles, CA
*
```

The search terminates and line 4 becomes the new current line. Type just 'S' and press ENTER. Edlin will search for the next occurrence of "CA" starting with the line after the current line (which is, of course, line 4).

```
9:*Palo Alto, CA
```

Try typing this:

```
1,#?SCA^Z          press ENTER
```

The only difference in this command from the one we just did is the question mark that preceeds the 'S' command. A question mark (?) makes Search stop after each find and wait for you to tell it whether to stop or continue searching. This is used when you expect to find more than one line containing the string but you are looking for one specific line. It is also an easy way to see all the lines containing a search string.

After each find, Edlin will stop and display the message:

```
O.K.?
```

If you press either 'Y' or ENTER, the search stops and the line becomes the current line. If you press any other key, the next matching line is displayed and you are again given the choice to proceed or stop. When the end of the file is reached, you'll get this message:

```
Not found
```

On the above example, press 'N' each time the "O.K.?" question appears. The screen will look like this:

```
*1,#?SCA^Z
    4:*Los Angeles, CA
O.K.?N
    9:*Palo Alto, CA
O.K.?N
   10:*San Francisco, CA
O.K.?N
Not found
*
```

If you omit the first line number in the command, Edlin will assume that you want to begin searching from the line following the current line. If you omit the second number, the search will continue through the last line of the file. If you omit the search string, Edlin remembers the last string specified in a command (if any) and uses it. Thus, when you typed 'S' with no other parameters, Edlin started at the line following the current line and searched through the end of the file for the search string you used earlier.

One final note about Search, the CTRL Z is not always necessary. It is not neccessary to end a string this way if it is the last thing on a command line -- but it is a good habit to get into.

Replace Text

Just as you can search for occurrences of a string in a file, you can also have Edlin automatically find strings and replace them with another string with, you guessed it, Replace.

Replace is used much like Search except that you give it two strings — one for the search and one for the replace.

Let's take the Search example one step further and change the "CA" to "California" on all the text lines. Type the following command:

```
1,#RCA^ZCalifornia^Z
          press ENTER
```

Edlin will zip through the file doing its replacing. Unlike Search, it will not stop after each line, but will continue until it reaches the end of the file. The screen will look like this:

```
*1,#RCA^ZCalifornia^Z
    4: Los Angeles, California
    9: Palo Alto, California
   10: San Francisco, California
*
```

Notice that only the lines containing "CA" were displayed. Had "CA" appeared more than once in a line, Edlin would have repeated scanning the line until all replacements had been made. The line is displayed each time a replacement is done.

You can omit any or both of the line numbers, the same rules apply as with Search. If you omit the second string, the search strings are replaced with... nothing. In other words, they are deleted.

The question mark (?) can be used with Replace to have it show you each replacement and ask for your approval (again with "O.K.?"). If you press ENTER or 'Y' the replacement is made and the next matching string is located. Pressing 'N' rejects the change and leaves the line unchanged. Again, the search will continue.

Are We There Yet?

No, not yet. I wanted to complete our discussion on Edlin with this issue, but there just wasn't room. So, we'll have to wrap it up next month.

But, hey, you've worked so hard today - get outside for a bit and enjoy the refreshing Autumn air. Prepare yourself for the Edlin finale in November.

(CRM)

WHAT'S NOT IN THE MODEL 100 MANUAL

Aileen and John Cornman

We began exploring our Model 100's ROM shortly after it arrived. Although the Disk/Video Interface had not been announced yet, we found several clues that prophesied its coming. Of more practical value, however, were the things we learned about our new computer that were documented only sketchily or not at all in the owner's manual.

The following items are among the more useful and important features we discovered in ROM. We have arranged them in alphabetical order by command name to be consistent with the order followed in the manual.

CLOADM? "filename"

This statement is not mentioned at all in the manual. It verifies a cassette tape recording of a machine language program (.CO file) in the same way that "CLOAD?" does for a BASIC program (.BA file).

EXP(numeric expression)

For those of the Napierian or natural persuasion, it should be pointed out that both the range for "numeric expression" and the result of exceeding it are incorrectly stated on Page 140 of the manual. The lower limit is -147.36544595162. Any number lower than this will result in zero being returned by the function (no error will be caused). The upper limit is 145.06 286085862. Any number higher than this will cause an overflow error

HIMEM

This variable determines how much memory is available to a BASIC program. The usual reason for setting it to a value lower than MAXRAM is to protect a portion of memory which contains a machine language program.

The manual tells how to set HIMEM with the CLEAR command, but neglects to say that the setting remains in effect from program to program unless you change it with another statement such "CLEAR 100,MAXRAM." Statements such as "CLEAR" and "CLEAR 100" will not affect HIMEM. Only if the second parameter is included in the CLEAR command will HIMEM be changed. If you forget to HIMEM back to MAXRAM, you will be shortchanging other BASIC programs that may need the additional memory.

To maximize the amount of memory available to your BASIC programs, include a CLEAR statement in every program to allocate only the amount of string space actually needed, and to set HIMEM to the maximum possible value.

INSTR(start position,search string match string)

The manual describes this statement fairly well, but fails to point out that if "match string" is null (length = 0 INSTR returns the value of "start position."

LINE(x1,y1,z1)-(x2,y2,z2),colo switch,BF

Just as with PSET and PRESE (described below), you may include third argument within the parentheses c1 and c2 may be any number between and 255. Unlike PSET and PRESET however, the third number seems t have no effect on how the line is drawn Although the syntax makes it possibl to specify points in three dimensiona space, we don't expect to see true 3-I PoCo graphics in the immediate future

MAXFILES

MAXFILES controls the amount o RAM that will be taken away from fre memory for use as file buffers. Once the memory is allocated to buffers it is n longer available for any other use. Eac unit of MAXFILES consumes 267 byte of free memory: 265 bytes for the buffe area itself and two bytes for a pointer t its location. Thus, MAXFILES=1 take 267 bytes, MAXFILES=2 takes 534 bytes, etc. On Page 165, the manual ambiguously states that it is the OPEN statement that "allocates a buffer for a file." This is true only in the sense that OPEN "assigns" a specific file to a specific buffer. The buffer(s) are actually "allocated" by the MAXFILES statement in the sense of setting aside memory for them.

By stating that BASIC, by default, sets MAXFILES to 1, the manual can be misleading in two ways, both of which can cause you to waste significant amounts of RAM.

This first misunderstanding could arise because you might reason as follows: "Since '1' is the default for MAXFILES, BASIC must set it to '1' if I do not include a MAXFILES statement in my program." This is a reasonable assumption, but not a correct one. BASIC sets MAXFILES to its default value of '1' only when BASIC is first run after a cold start. After that, MAXFILES retains its value from program to program unless you set it to a new value with an assignment statement. BASIC will *not* automatically reset MAXFILES to '1' every time you select BASIC from the menu or run a new BASIC program. To illustrate how memory can be wasted, assume than you run Program A that contains the statement "MAXFILES = 4." If you then run Program B, which uses no files at all, but contains no MAXFILES statement, there would still be over 1K of RAM allocated to unneeded file buffers that is unavailable to Program B.

The second misunderstanding could arise if you think: "Since '1' is the default value of MAXFILES, '1' must be the lowest value to which MAXFILES can be set." The fact is, however, that you can set MAXFILES to zero if your program will not use any files at all. Thus, MAXFILES = 0 will give your program 267 more bytes to work with than running with the default of MAXFILES = 1. You may set MAXFILES to any number between zero and 15.

As a general rule then, every program should set MAXFILES to the exact maximum number of files that will be simultaneously open at any one time. To save memory, file numbers may be reused as long as one file is closed before the next file is opened with the same file number. MAXFILES also sets a limit on the numbers you can use as file numbers in OPEN statements. If MAXFILES = 2, then '1' and '2' are the only valid file numbers you may use.

MAXRAM

The manual makes it sound like this is a machine-dependent variable in the sense that it will tell you whether you have an 8K, 16K, 24K, or 32K machine. All Model 100s when cold-started, however, have the same value of 62960 for MAXRAM regardless of the amount of RAM installed. The first 8K or RAM occupies the high end of memory space (E000-FFFF), with additional RAM being added in 8K increments at progressively lower addresses until they meet up with ROM at 7FFF. The address of the lowest installed RAM is kept in the two bytes at location 64192 (FAC0). To find out how much RAM is installed at run time, use the formula: RAM = 65536 - (PEEK(64193)*256 + PEEK(64192)).

The only thing that will lower MAXRAM's value from its initial setting is the installation of Disk BASIC or other software that must be in the protected "system RAM area."

MID$(string expression1,position, length) = string expression2

As with other implementations of BASIC, using MID$ on the left side of the equal sign is one way to avoid the delays caused by periodic string space reorganization. By using this form of the statement, "string expression1" is updated in place, which means that no new copy is made of it and no additional string space is used up.

PRESET(x-coordinate,y-coordinate, color)

The manual describes only the first two of the arguments within the parentheses. "Color" is a numeric expression in the range zero through 255. Even values of "color" will cause the pixel to be set, while odd values cause it to be reset (turned off).

PSET(x-coordinate,y-coordinate,color)

"Color" is as described under PRESET, except that the effect is reversed: odd values set the pixel, even values reset it. As you can see, you only need one of these two statements in your program. By using variables for x, y, and color, you can make them do the job of both PSET and PRESET by simply changing the value you assign to the color variable.

REM

If you use the apostrophe as an abbreviation for REM, you are wasting RAM. The "abbreviated" form requires three bytes of memory while the "long" form takes only one. While typing an apostrophe instead of REM saves two keystrokes, each saved keystroke uses a byte. (Fortunately, this is not the case when you abbreviate PRINT by typing a question mark. Both ways use only one byte of RAM because the question mark is changed into the same tokenized keyword as PRINT.)

RND(numeric expression)

The manual incorrectly states that RND will return a new random number as long as "numeric expression" is not zero. This is only true as long as "numeric expression" is positive. If it is negative, RND keeps returning the same number over and over again. If you want a new random number each time, make sure that whatever you put inside the parentheses never evaluates to a negative number.

SCREEN on/off

Before the introduction of the Disk/Video Interface, we wondered why two parameters are needed to set and clear the label line on the LCD. The manual says to write "SCREEN 0,1" to turn the label line on (display it on Line 8), and "SCREEN 0,0" to turn it off (erase it). We suspected that the first number was meant to designate one of two screens and that the second number turns the label line on or off for the designated screen.

We could see in ROM that screen number 0 is the LCD and also the default screen number. Therefore, if you write "SCREEN ,1" or "SCREEN ,0" BASIC will understand you to mean "SCREEN 0,1" and "SCREEN 0,0" respectively and apply the commands to the LCD.

The syntax is also set up to handle statements such as "SCREEN 1" and "SCREEN 0." Of these, only "SCREEN 0" executes in a non-disk system without causing an error. Since both CRT and LCD are defined device codes in ROM, it seemed likely that "SCREEN 1" would be used to switch over to a CRT monitor, while "SCREEN 0" would be used to switch back to the LCD display.

VARPTR(variable name)

The manual does little more than remind you that the value returned by this function is a signed integer. It tells you nothing at all about what you can expect to find at the memory address it returns to you. What you will find there, of course, depends on the kind of variable in question, but the first three bytes in front of the returned address contain the same kind of information, regardless of variable type as shown below:

Address	Contents
VARTPR-3	Variable type (2 = INT, 3 = STR, 4 = SNG, 8 = DBL).
VARPTR-2	Two-byte variable name.

Depending on the variable type, the contents of memory locations at VARPTR and higher addresses are as follows:

For integer variables:

Address	Contents
VARPTR+0	Low order half of the two-byte signed integer.
VARPTR+1	High order half of the two-byte signed integer.

For single precision variables:

Address	Contents
VARPTR+0	Sign and exponent of the number. A decimal 64 means that the decimal point is located to the immediate left of the first digit; 65 locates the decimal point to the left of the second digit and so on. If the number is negative, decimal 128 is added to the exponent.
VARPTR+1	The two most significant digits of the number, as packed BCD digits.
VARPTR+2	The two next most significant BCD digits.
VARPTR+3	The two least significant BCD digits

For double precision variables:

Address	Contents
VARPTR+0	Sign and exponent as for single precision above.
VARPTR+1	The two most significant BCD digits of the number.
VARPTR+2	The two next most significant BCD digits.
VARPTR+3	The two next most significant BCD digits.
VARPTR+4	The two next most significant BCD digits.
VARPTR+5	The two next most significant BCD digits.
VARPTR+6	The two next most significant BCD digits.
VARPTR+7	The two least significant BCD digits.

For string variables:

Address	Contents
VARPTR+0	Length of the string as a one-byte unsigned integer.
VARPTR+1	The low-order half of the address of the string itself. This may be an address within the program or in string space.
VARPTR+2	The high-order half of the address

For array variables of one dimension, e.g. VARPTR(A(0)):

Address	Contents
VARPTR+0	Two bytes, low order half first, containing the length of the rest of the array counting from VARPTR +2 through the end of the array.
VARPTR+2	The number of subscripts needed to reference an element in this array.
VARPTR+3	Two bytes, low order half first, containing 1 plus the DIMension of the last (or only) subscript.
VARPTR+5	The variables themselves (three byte length and address in the case of strings)

For arrays of multiple dimensions, it gets a little more complicated. First of all, what VARPTR returns depends on the number of dimensions. Secondly, there are as many DIM fields (like VARPTR+3 above) as there are subscripts before the actual variables themselves begin. If you need to use VARPTR with multi-dimensioned arrays, use zero for the value of each subscript and determine experimentally what address is being returned in your particular case. You should be able to find a point of reference when you see the variable type and name fields anywhere from 10 or more bytes before the returned address.

VARPTR(#file number)

The manual doesn't mention this use of VARPTR. If you put a point sign and file buffer number in the parentheses, in place of a variable name, VARPTR will return the address of the 265 byte file buffer with that number. The file number must be between one and whatever MAXFILES is set to. The layout of the buffer is as follows:

Address	Contents
VARPTR+0	Flag byte: 00 = not open, 01 = open for input, 02 = open for output or append.
VARPTR+1	(Doesn't appear to be used in a non-disk system.)
VARPTR+2	Two bytes, low order half first, containing the address of the address field in the directory entry for this file.
VARPTR+4	Device: 248 = RAM, 249 = MDM, 250 = LPT, 251 = WAND, 252 = COM, 253 = CAS, 254 = CRT, 255 = LCD. (dec.)
VARPTR+5	(Doesn't appear to be used in a non-disk system.)
VARPTR+6	Pointer to the next byte to be read into or out of the buffer. (0 to 255 dec.)
VARPTR+7	Two bytes, low order half first, containing the length of the file, not counting what is in the buffer that has not yet been stored with the file during output and append operations. On input, it is the number of full or partial "buffer loads" of data that have been read in.
VARPTR+9	256 bytes for the actual buffer area itself, to hold the data that is read from or written to the file.

Undefined Errors Defined

Two of the error codes which the manual calls "Undefined Errors," do, in fact, have specific causes.

Error code 50, "IE," might well stand for Internal Error. It results when unacceptable codes or addresses are found in the system's own control blocks for keeping track of file operations. Probably the only time you will see this error message is if you mistakenly POKE a value in the wrong place.

"FL," error code 57, will occur if you try to create more files than there are slots on the menu and in the file directory. You might think of FL as standing for "Full."

Generating Control Characters From The Keyboard

While all of the first 26 control characters may be entered from the keyboard by using the CTRL key with an alphabetic key, many of them may also be created by pressing other keys which may be more meaningful to the user in particular applications. The following chart lists some of the alternatives:

Decimal	Hex	Standard	Alternative key(s)
1	01	CTRL/A	SHIFT/left-arrow
2	02	CTRL/B	SHIFT/down-arrow
3	03	CTRL/C	BREAK
6	06	CTRL/F	SHIFT/right-arrow
7	07	CTRL/G	(No alternative, but note that you can cause a 'beep' with PRINT CHR$(7).
8	08	CTRL/H	BKSP
9	09	CTRL/I	TAB
11	0B	CTRL/K	(No alternative, but note that you can 'Home' the cursor with PRINT CHR$(11).
12	0C	CTRL/L	(No alternative, but note that you can clear the screen with PRINT CHR$(12).
13	0D	CTRL/M	ENTER
17	11	CTRL/Q	CTRL/left-arrow
18	12	CTRL/R	CTRL/right-arrow
19	13	CTRL/S	PAUSE
20	14	CTRL/T	SHIFT/up-arrow
23	17	CTRL/W	CTRL/up-arrow
26	1A	CTRL/Z	CTRL/down-arrow

Who Are Suzuki, Hayashi And Ricky Anyway?

If you happen to read the names in the file directory just after a cold start, you see the three files named Suzuki, Hayashi and Ricky. Their functions are described below.

Suzuki is the name given to the file in which BASIC keeps a program that you have typed in or read in from cassette, but have not yet saved under its own name. When you select BASIC from the main menu, Suzuki is automatically loaded for you. Once you save a BASIC program under its own name with a SAVE command, Suzuki will be an empty file. You never see Suzuki on the main menu because BASIC has it marked as an "invisible file." You should be aware of Suzuki's existence because it may be costing you memory that you need for another program. To tell if Suzuki is empty or not, enter BASIC from the main menu and press F5. If anything is listed, then Suzuki is not empty. If you do not want to keep the program you see listed as Suzuki, enter the command NEW to make Suzuki an empty file.

Hayashi is the name given to the "paste buffer." Whenever you use the TEXT program to "cut" or "copy" a string of text, that string becomes the contents of the Hayashi file. Like Suzuki, Hayashi is also an invisible file, so you will never see it listed on the menu. Hayashi can also be the cause of wasted memory, because the string will continue to occupy storage until it is replaced by another string. The quickest and easiest way to insure that Hayashi is not wasting any RAM is to enter a CLEAR command. CLEAR will empty Hayashi of whatever string was previously in it.

Ricky is given the honor of occupying the first directory entry which is available for general use. When all other directory slots have been used, Ricky will be the last to go. As an invisible file, Ricky will never appear on the menu.

Memory Map

The layout of the 64K memory space for a 32K machine follows. Where the Hex address is in parentheses, it is the address of the pointer field for that item. The pointer field will contain the actual starting address of the item.

Address	Item
0000-7FFF	ROM.
8000	Always 00.
8001-F5F0	User RAM in this order:
8001	BASIC program files (.BA).
(F99A)	Suzuki file (BASIC program not yet saved by name).
(FBAE)	Text files (.DO).
(F9A5)	Hayashi (paste buffer).
(FBB0)	Machine language program files (.CO).
(FBB2)	Simple BASIC runtime variables.
(FBB4)	BASIC runtime arrays.
(FBB6)	Free memory.
(F678)	BASIC stack (grows downward toward lower address) BASIC string space (between file buffers and stack).
(FB67)	File buffers.
F5F1-FFFF	System RAM, some useful pointers and flags:

Addr.	Name	Explanation
F630	FKFLGS	One-byte flag per FN key: 00 = off, 01 = on.
F63D	LBLFLG	Label line (line 8) flag: FF = labels now showing.
F63E	SCRFLG	Scroll flag: AF = no scroll, 00 = scroll (unlock).
F648	REVFLG	00 = normal, AF = reverse character mode.
F675	PRTFLG	00 = print to LCD, 01 = print to printer.
F678	BEGSTR	Contains address of beginning of string space.
F67A	EXLINO	Current line number, FFFF if immediate mode.
F67C	TOPBAP	Top address of current BASIC program.
F93D	ONTMTM	"ON TIMES" time in six bytes: SMMHH.
FAAF	IPLNAM	Up to nine-byte IPL program name + 0D00;0000=none
FAC0	BEGRAM	Beginning address of all RAM.
FB67	FILBUF	Start address of file buffer area.
FBAE	BEGDOF	Beginning address of .DO files.
FBB0	BEGCOF	Beginning address of .CO files.
FBB2	BEGVAR	Beginning address of BASIC variable pool
FBB4	BEGARR	Beginning address of BASIC arrays.
FBB6	ENDARR	Address of end of arrays and start of free memory.
FBBA	DEFTBL	26 DEF codes: 02=G, 03=S, 04=I, 08=# for letters A-Z.
FF44	SNDFLG	Sound flag: 00 = sound on, AF = sound off.
FFAA	KBQCNT	Count of keys pending in the keyboard queue.

Do Aircraft Interfere With Portables?

By Jim Hawk

Owners of the Model 100 and other portables were occasionally prevented from using them on some commercial airliners for a few months of confusion last year. They've since been given a relatively clean bill of health. The tables have now turned for those who may have suffered some inflight frustration. After earlier suspecting that the slight radio interference generated by portables like the Model 100 might affect critical aircraft radio frequencies, a government industry group is now investigating just the opposite. The head of a special committee of the Radio Technical Commission for Aeronautics in Washington is asking for manufacturers' help in determining whether international flights might pose a data hazard to portable computers. Frank White says in a working paper to committee members, "We have been asked to consider the possibility of HF transmissions interfering with passenger-operated electronic devices, particularly computers." The question came from a ham radio operator named Jack O'Neill, who also happens to be director of aviation electronics for U.S. Air. "HF" stands for High Frequency, or 3 to 18 MHz (as opposed to the much higher VHF or Very High Frequency band of 118-137 MHz normally used for aircraft communications). These HF transmissions occur *only* on international operations since domestic aircraft do not operate on what we used to call "shortwave" radio. That is the same general band where you can hear ham radio operators, Voice of America and the like. For a flight that may be hundreds of miles out over the ocean, the big advantage is that HF can bounce back and forth in the ionosphere, allowing long distance communication. The drawback, at least as far as possibly affecting sensitive electronics, is that you have to use high power to transmit sometimes half a kilowatt. And the wing or tail of an ocean-jumping airplane can become part of the broadcasting antenna in the HF band, meaning signals may be fairly strong inside. Says Chairman White, "I don't know how susceptible computers are, or how much energy we're putting into the cabin, but there's no harm in getting the answers."

A fourth meeting of Special Committee 156 is set in Washington this fall, and the question of aircraft HF transmissions interfering with portable computers is on the agenda. Most of the two-day meeting will be spent going over reams of technical data gathered this past summer dealing with the original charge of the committee: ". . . potential interference to aircraft electronics equipment from devices carried aboard." Douglas Aircraft, Boeing, and the Federal Aviation Administration technical center will all present reports, although an early peek indicates there will be no surprises.

In a memorandum to all members, committee Chairman Frank White states with unusual candor that it may be difficult to get the Federal Communications Commission to agree to lower the limit of permissible radiation from portables without much more convincing evidence, ". . . since increasing the shielding of the portable devices is costly and therefore must be clearly justified."

One solution proposed by White has already been implemented by at least Eastern Airlines: "... recommend that passengers not be permitted to use any portable electronic device during takeoff and landing when belted down, trays are stowed, and seat backs are vertical." This 10- to 15-minute inconvenience seems logical since these are times when navigation and communication are the most critical to the safety of a flight. Though there have been no incidents reported to date, wouldn't it be ironic if the ultimate finding of this committee turned out to be that certain aircraft frequencies might be a bigger hazard to portable computers than the other way around? PCM

New Products

Tandy Model 1200 Announced

Tandy IBM compatible has arrived. Officially named the Tandy TRS-80 Model 1200, this machine is said to function identically to the popular IBM PC XT.

The only Tandy 1200 configuration offered will be a 256K RAM, 360K floppy disk, and 10-meg hard disk system unit and keyboard for $2,995. Monochrome or color displays and adapters will be offered at an additional price. MS-DOS, the disk operating system, is also sold separately. While Tandy includes MS-DOS with its advanced Model 2000, it is keeping true to IBM's form by not including it in the 1200's price.

Expandable Phone Modem With Clock/Calendar

ProModem 1200™ from Prometheus Products is a Hayes compatible Bell 212A, 300 and 1200 Baud phone modem with built-in clock calendar. It offers the option of adding a buffer memory with up to 64K of storage.

Standard features include auto-answer and auto-dial, programmable intelligent dialing, tone and pulse dialing, built-in speaker with volume control, and separate phone and data jacks to permit switching between voice and data. Suggested list price is $495.

The ProModem 1200 can be purchased with the optional buffer installed or it can be added later. The buffer card comes with 2K of battery backed-up CMOS memory to protect time, date, operating parameter, and other data stored in memory from loss during power down. The buffer card lists for $99. Additional memory, in increments of 16K, is available up to a maximum of 64K.

In the auto-answer modem, incoming messages are automatically stored and the time recorded. Operation of Pro-Modem can be unattended, with or without the host computer being operational.

A plug-in 12-character alphanumeric display is available for $99 list price. It shows operating status, diagnostic messages, phone numbers, and time/date information.

For additional information, contact Prometheus Products, Inc., 4-237 Fremont Blvd., Fremont, CA 94538, (415) 490-2370.

Software Utilities

Applied Logic Engineering has introduced a number of new software utilities for the Model 100. Among them:

Barcode print utility — This stand-alone program will take any alphanumeric characters in a .DO file and convert them into 3 of 9-type bar code for printing on the TRS-80 DMP 200 printer. The bar code can then be read using the bar code wand. The program is written for tractor-fed stick-on labels for individual printing. It's in BASIC and can be modified for use on most dot-matrix printers. $12.95.

Clear to send request to send handshaking — An assembly language utility that will allow the programmer to use RTS/CTS hardware handshaking on the serial RS-232 port for communicating with other computers that do not support XON/XOFF software handshaking. Full documentation on use is included. $8.95.

Intel Hex format file loader — This program will allow the user to load an Intel Hex Format object file created on another computer into the proper location in the Model 100. Use your larger micro to create assembly language programs, assemble, and transfer the object file to the Model 100 to execute. $10.95.

Decimal to Hex/Hex to decimal utility — This program converts numbers from decimal notation to hexadecimal notation and vice versa. $7.95.

Contact Applied Logic Engineering, 6320 Boone Avenue North, Suite 204, Brooklyn Park, MN 55428, (612) 535-0094.

PCM BAR CODED LISTINGS

SORT (FROM PAGE 36)

1

2

3

4

5

6

7

8

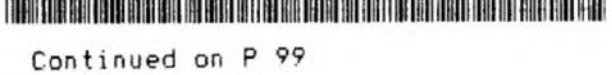
9

10

11

12

13

14


Continued on P 99

GRAF3D (FROM PAGE 30)

1

2

3

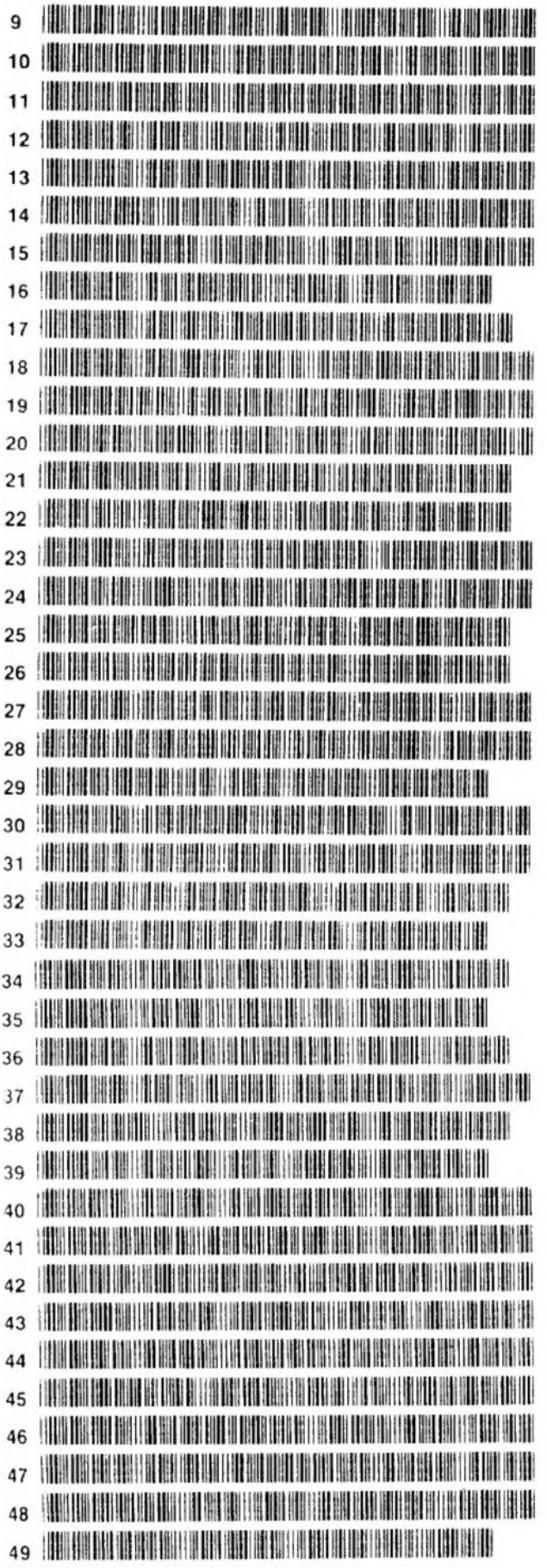
4

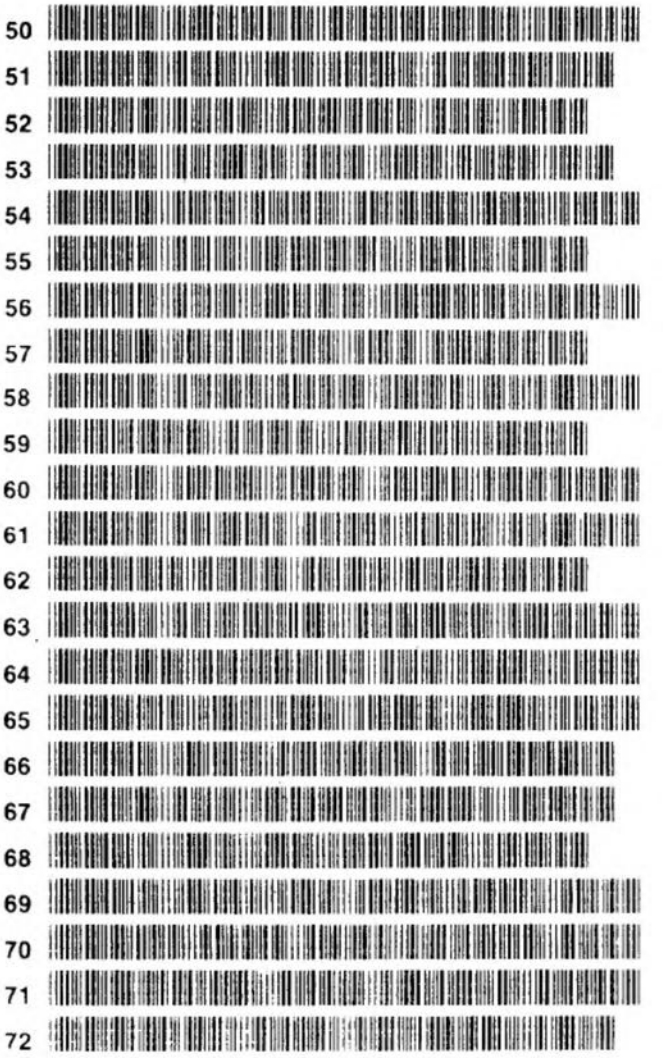
5

6

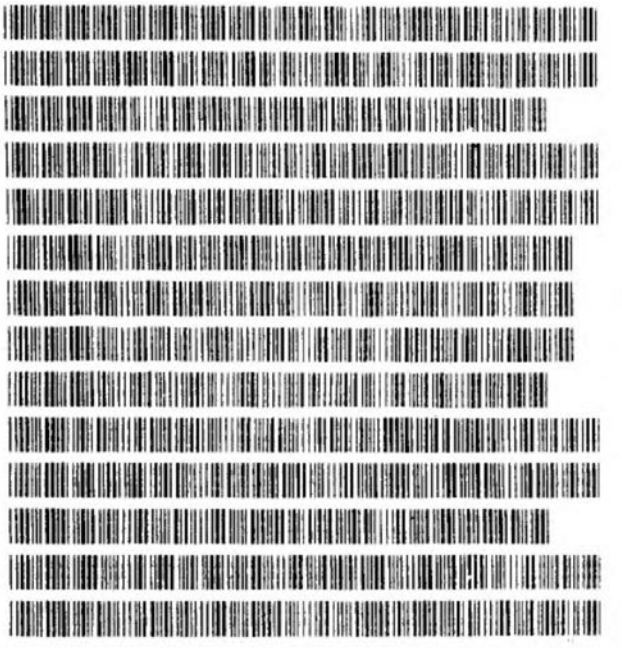
7

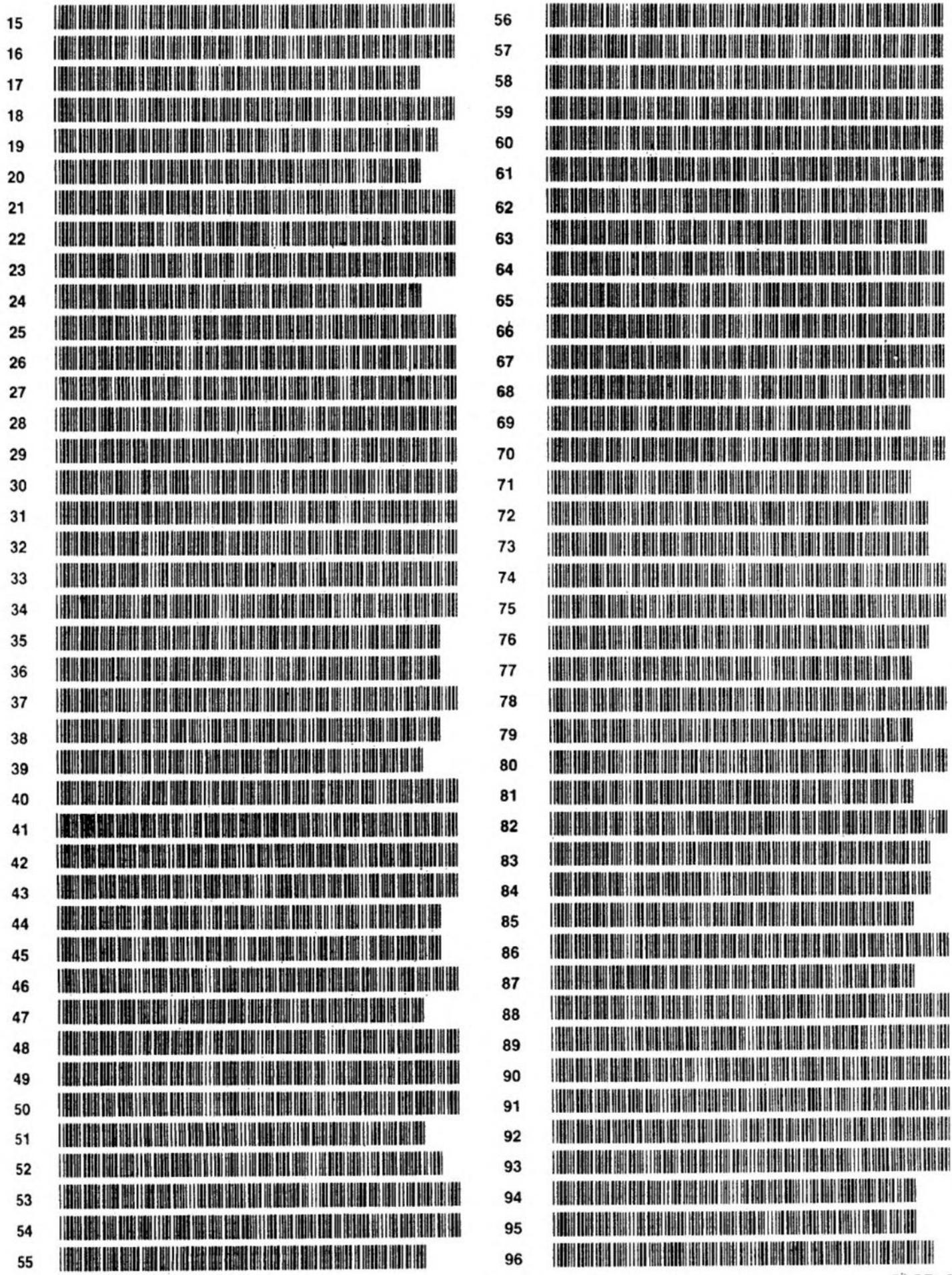
8

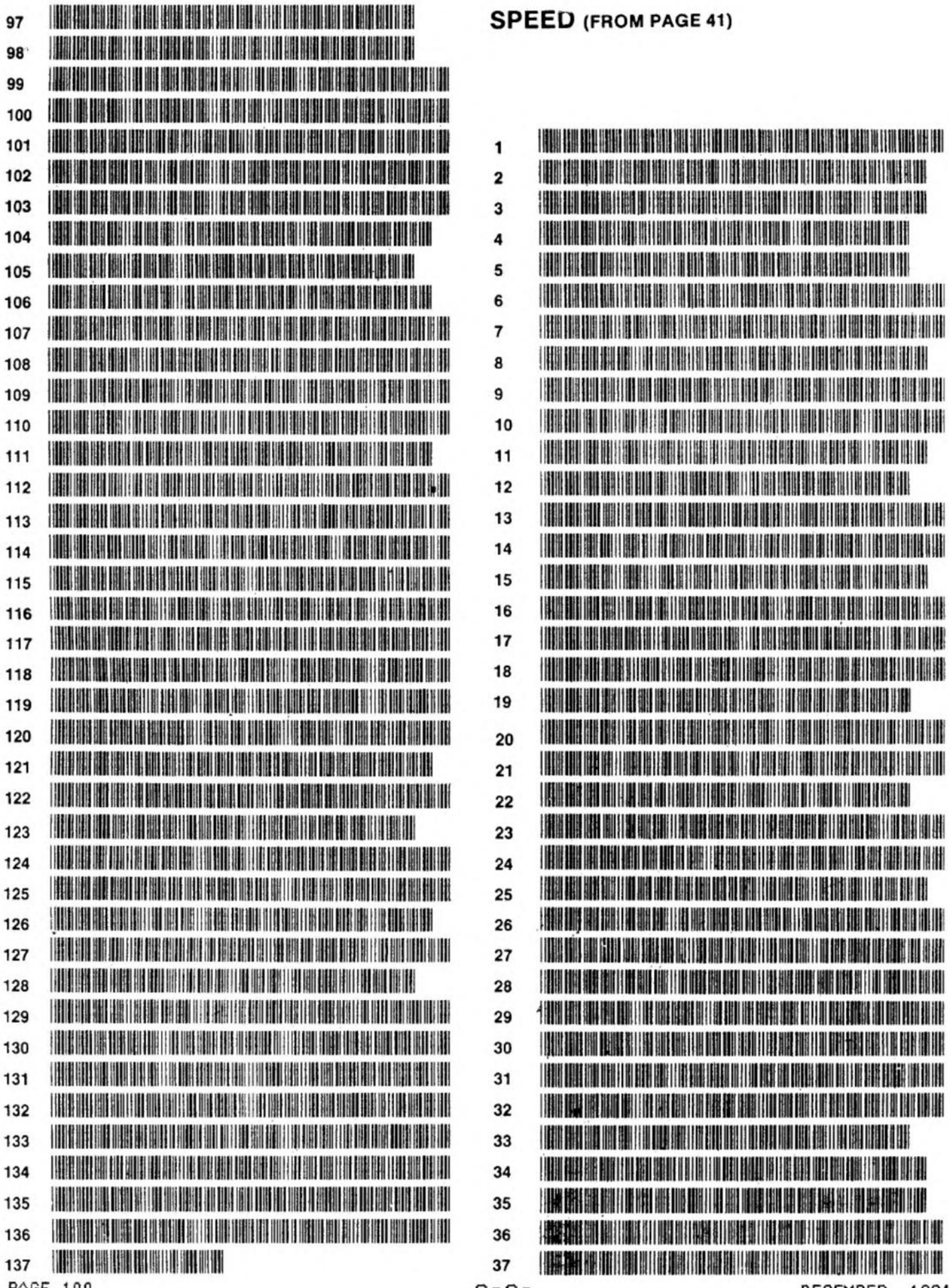


CIPHER (FROM PAGE 14)





SPEED (FROM PAGE 41)



38
39
40
41
42
43
44
45
46
47
48
48
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

TIME (FROM PAGE 42)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

"It says 'Mother'!"

AUSTRALIAN CoCo
The only other magazine
You need for your TRS 80
Color Computer

THE RAINBOW
Registered by Australia Post-
Registration No. QBG 4009
Graham Morphett, P.O.Box 1742, Southport. QLD. 4215.

POSTAGE
PAID
AUSTRALIA