For your TANDY Color Computer

ANNIVERSARY ISSUE

PNG K5.95
NZ $3.95
$4.50

AUSTRALIAN RAINBOW

Registered by Australian Post — Publication No. QBG 4009

August, 1986 No.61

SEE YOU AT

CoCoConf '86

August 30 - 31

GOLDLINK
on
VIATEL

GAMES
TUTORIALS
- much more!!

AUSTRALIAN RAINBOW

CONTENTS

RAINBOW Info

How To Read Rainbow

Please note that all the BASIC program listings you find in THE RAINBOW are formatted for a 32-character screen — so they show up just as they do on your CoCo screen. One easy way to check on the accuracy of your typing is to compare what character "goes under" what. If the characters match — and your line endings come out the same — you have a pretty good way of knowing that your typing is accurate.

We also have "key boxes" to show you the *minimum* system a program needs. But, *do* read the text before you start typing.

Finally, the little cassette symbol on the table of contents and at the beginning of articles indicates that the program is available through our RAINBOW ON TAPE service. An order form for this service is on the insert card bound in the magazine.

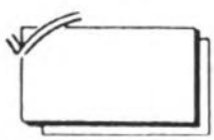
What's A CoCo

CoCo is an affectionate name that was first given to the Tandy Color Computer by its many fans, users and owners.

However, when we use the term CoCo, we refer to both the Tandy Color Computer and the TDP System-100 Computer. It is easier than using both of the "given" names throughout THE RAINBOW.

In most cases, when a specific computer is mentioned, the application is for that specific computer. However, since the TDP System-100 and Tandy Color are, for all purposes, the same computer in a different case, these terms are almost always interchangeable.

The Rainbow Check Plus



The small box you see accompanying a program listing in THE RAINBOW is a "check sum" system, which is designed to help you type in programs accurately.

Rainbow Check PLUS counts the number and values of characters you type in. You can then compare the number you get to those printed in THE RAINBOW. On longer programs, some benchmark lines are given. When you reach the end of one of those lines with your typing, simply check to see if the numbers match.

To use *Rainbow Check PLUS*, type in the program and CSAVE it for later use, then type in the command RUN and press ENTER. Once the program has run, type NEW and ENTER to remove it from the area where the program you're typing in will go.

Now, while keying in a listing from THE RAINBOW, whenever you press the down-arrow key, your CoCo gives the check sum based on the length and content of the program in memory. This is to check against the numbers printed in THE RAINBOW. If your number is different, check the listing carefully to be sure you typed in the correct BASIC program code. For more details on this helpful utility, refer to H. Allen Curtis' article on Page 21 of the February 1984 RAINBOW.

Since *Rainbow Check PLUS* counts spaces and punctuation, be sure to type in the listing exactly the way it's given in the magazine.

```
10 CLS:X=256*PEEK(35)+178
20 CLEAR 25,X-1
30 X=256*PEEK (35)+178
40 FOR Z=X TO X+77
50 READ Y:W=W+Y:PRINT Z,Y;W
60 POKE Z,Y:NEXT
70 IFW=7985THENB0ELSEPRINT
   "DATA ERROR":STOP
80 EXEC X:END
90 DATA 182, 1, 106, 167, 140, 60, 134
100 DATA 126, 183, 1, 106, 190, 1, 107
110 DATA 175, 140, 50, 48, 140, 4, 191
120 DATA 1, 107, 57, 129, 10, 38, 38
130 DATA 52, 22, 79, 158, 25, 230, 129
140 DATA 39, 12, 171, 128, 171, 128
150 DATA 230, 132, 38, 250, 48, 1, 32
160 DATA 240, 183, 2, 222, 48, 140, 14
170 DATA 159, 166, 166, 132, 28, 254
180 DATA 189, 173, 198, 53, 22,-126, 0
190 DATA 0, 135, 255, 134, 40, 55
200 DATA 51, 52, 41, 0
```

Using Machine Language

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

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

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

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

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

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

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

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

The Crew

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

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

PRINT #-2

Well, that's just great! It's the Rainbow's fifth birthday, and I only realised it yesterday somewhere in between making tapes and typing in Laurie O'Sheas' latest article.

Which is not surprising. I've been stuck behind a Viatel Computer for days? weeks? setting up most of the Goldlink database to go onto Viatel database. Uhg! Talk about a strain. I come in Monday, I type Viatel. I come in Tuesday, I type Viatel. I come in Wednesday ... etc and it goes on. But I manage to get out of Viatel from time to time.

So what do I get out of this? Sore fingers, a desk that's swamped with The Laymans Guide to Viatel (a 300-page step-by-step instruction manual to just understand & operate Viatel), letters, programs, 5-6 masters that have had to be re-made and 240 page of loose Viatel paper. (The odd amount of paper you need so you can find a particular page to edit.)

So what does that say about Australian Rainbow? Lots. It says that we've come a far way from the photocopy that was made in Lonnie Falk's (Editor of American Rainbow) bedroom. I mean, Greg first took over the job of editing the first Australian Rainbow. He gave the Rainbow a format, edited the already edited American Rainbow and made it a very popular magazine here downunder.

Then came his death. Untimely and unnecessary.

But along came somebody known as Graham who took over and created a totally new format along with creating Australian CoCo. He took over the job of being Editor of Australian Rainbow, and now that job is mine!

So what does an editor do all day? Sit back with your feet on the table and tell everyone else what to do? This one doesn't (can't). This editor dares to go down in 'the dungeon' from time to time to get a few back issues for somebody. (The dungeon, as everyone knows, is the area underneath 'the office'. It is a dark, cold & clammy place for storage. It is also a good setting for an adventure, although somebody already took up on that idea.)

Rumours had it that we were moving. We were. But we didn't. So we haven't. Therefore we're still here. (Are you still with me ... ??)

Lets go on with something everyone can understand (me for one).

CoCoConf is at the end of this month and Graham has taken on the honourable job of telling all of us what happens when and where (sure ...). Look for his words of wisdom later in the magazine, and then come up to CoCoConf despite what he says!

COCOCONF'86

CoCoConf is on this month, and we're all very excited about it!

There are more people this year than last, there are more tuts this year than last, and we'll have more computers for you to use at Conf than last year.

The big news for the MS DOS fraternity is that Paul Humphreys has consented to give a talk on the Networking of MS DOS computers - a subject close to the hearts of business and education users.

On the CoCo front, the conference presents some of the big names in computing - people like Ken Allen (Tandy Computer Buyer), Bob Delbourgo, John Redmond, Johanna Vagg, Ross Eldridge and many others who are experts in their fields, and the tutorials they present will be "leading edge" stuff.

Organise yourself to come up for the weekend. I know when one stays in one place for any length of time, it is easy to think of other places as being too hard to get to, but the reality is that the Gold Coast is not hard to get to, especially if you live on the eastern seaboard. The trouble you take to get here will be worth it.

If you are coming, we need to know by the 15th August - we need time to organise the caterers, the rooms, the seating and so on, so let us know NOW!

Many of you have asked about accommodation on the Gold Coast, and the following is a quick summary of some places which MAY have some space on the weekend of the 30th. Keep in mind that this is a school holiday weekend in NSW, so most places will be fairly full.

Motels:

Bahamas Motel	075-36-1824
Cnr Marine Pde & Hill St., Coolangatta	
Banora Point Motor Inn	075-54-2222
Cnr Pacific Hwy & Terranora Rd Banora Point	
Beachcomber Motor Lodge	075-36-3033
122 Griffith St., Coolangatta	
Cook's Endeavour Motel	075-36-5399
26 Francis St., Tweed Heads	
Coolangatta Motel	075-36-6244
95 Golden Four Dr., N Kiera	
Fairlight Motor Inn	075-36-2633
91 Pacific Hwy., Tweed Heads	

Caravan Parks:

The Homestead Caravan Park	066-74-1824
Pacific Hwy., Chinderah	
North Star Caravan Park	066-76-1234
Coast Rd., Hastings Point (This one is further away, but nice)	
River Retreat Caravan Park	075-36-1400
Drydock Rd., Tweed Heads	
Tweed Heads Caravan Park	075-36-5682
112 Drydock Rd., Tweed Heads	

Hotels:

Conrads (The Casino)	075-92-1133
Greenmount (Great!)	075-36-1222

CoCoConf'86.
Tutorials

Basic BASIC	Johanna Vagg.
Advanced BASIC	Mike Turk & Alex Hartmann.
FORTH	John Redmond & John Foxon.
OS-9	Graeme Nichols, Ron Wright, & Jack Fricker.
68000	Ron Wright, Jerome Slappy, & Jackie Cockinos.
MS DOS	Brian Dougan, Barry Cawley, Paul Fulloon.
Education	Ross Eldridge, Bob Horne, & Bob Delbourgo.
Games	Michael Horne, Andrew White, Nicholas Merantes, & Tony Evans.
Viatel	Ron Wright.
The Future	Mike Turk & Ken Allen (Tandy).
MC 10 Computing	Jim Rogers & friends.

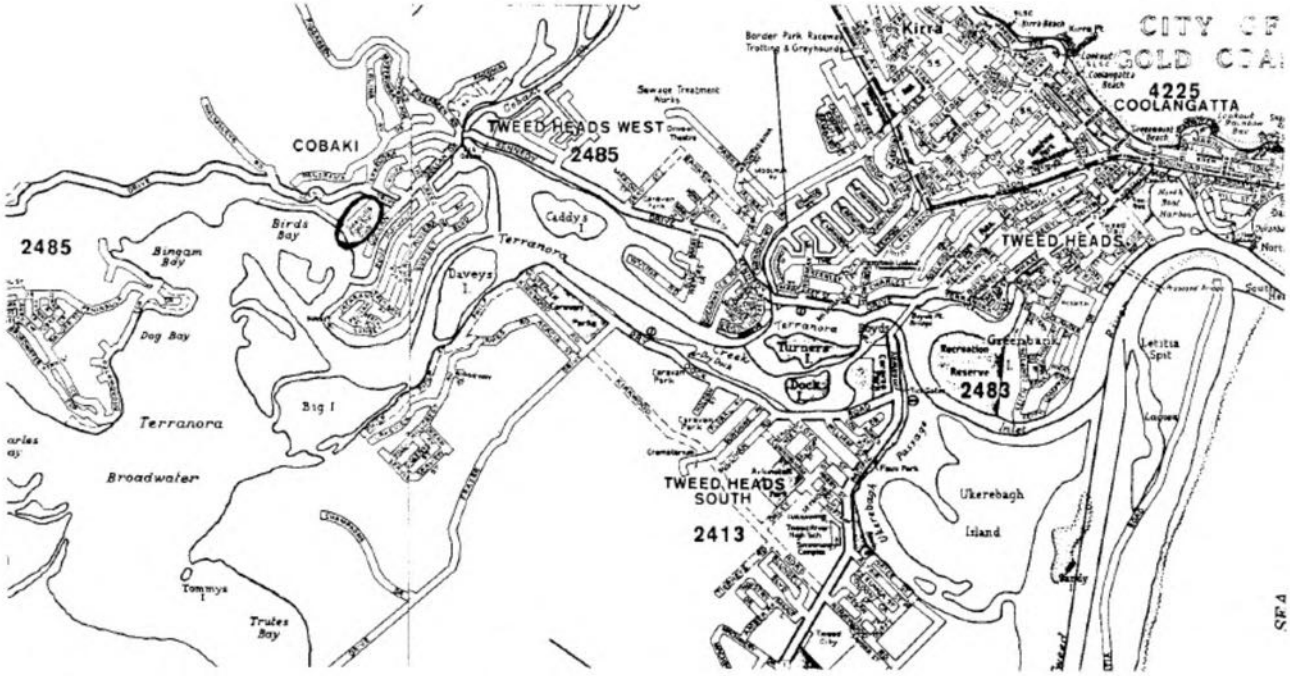
Proposed
Weekend TimeTable.

Sat AM	Sat PM	Sun AM	Sun PM
Basic BASIC (J Vagg)	Advanced BASIC (A Hartmann)	Hardware (G Fiala)	Games
Advanced BASIC (M Turk)	Viatel (R Wright)	FORTH (Redmond et al)	68000 Computers
OS-9 (J Fricker)	OS-9 (G Nichols)	OS-9 (R Wright)	
MS DOS (B Dougan)	MS DOS (B Cawley)	MS DOS (P Fulloon)	The Future (M Turk & K Allen)
Education (R Eldridge)	Education (B Horne)	Education (B Delbourgo)	
	MC 10 (J Rogers)	MC 10 (J Rogers)	

Remember that the southern end of the Gold Coast is quite a distance from the northern end, so lodgings in Surfers Paradise will mean a journey of about 30 minutes to get to CoCoConf.

For your information, I've included a map to show you where Seagulls is, and a timetable of events. Further news and information can be obtained from Goldlink on Viatel *642# as the time grows nearer.

Graham.



COCOCONF '86

WHAT'S HAPPENING:- Tutorials on Advanced BASIC, Basic BASIC, Educational use of computers, OS9, MS DOS/GW BASIC, FORTH, The CoCoConnection HARDWARE mods include:- high K upgrades (128,256,512,1mb) AND THAT'S JUST SATURDAY!! Saturday night we have our dinner and prize session. (this is included in your registration fee) SUNDAY continues with MORE tutorials plus the opportunity to browse/buy the large range of software and hardware available for the CoCo and T1000. There will be lots of bargains!

SPEAK UP!:- Now is your chance to suggest your ideas for any tutorials we may not have mentioned. (participants only).

LOCATION:-
SEAGULLS RUGBY LEAGUE CLUB
TWEED HEADS.

DATE:- Sat 30th & Sun 31st August 1986.

REGISTER NOW!!
We can only accept a limited number of people this year. DON'T MISS OUT! on a top weekend of FUN, FRIENDSHIP and LEARNING.

Name: .....

Address: .....

Phone: .....

No. People attending: .....

$39.95 per person/1st family member
$20.00 per additional family member
$9.95 dep. balance by 15/8/86
Cost includes:- tutorials, dinner Sat. night, morning and afternoon tea.

Tutorials likely to attend: .....

.....

Please find enclosed:
chq/money order/bankcard/visa/mastercard

Card No. .....

Signature: .....

GOLDLINK

*642#

I guess I'm a little biased - I'm certainly emotionally involved - heck Alex and I have been LIVING Viatel for the last month - so if you think I'm out of my tree, you could be pretty close.

But I think not.

We've dedicated the time because we very firmly believe that in a short time most computer owners will be using their computer to do much more that it is at present.

The on line data bases in general are becomming something for other media to reckon with.

It is little more that a year ago that we revealed the Stars system, and in the time since it began, we've seen that system grow into something very promising.

Viatel is in the same boat.

12 months ago, when it started, naturally, it was all index pages, and very little data.

As it grew, people started to use it - and for very good reason - it began to meet a need.

In fact in the first year it met the needs of 16000 paying customers!

Not bad is it, for a new venture to get 16000 customers or over 1000 new customers per month in their first year of training!

We joined Viatel, despite the fact that most of you don't own a modem suitable for the system, simply because we believe we can provide a cost effective service which will meet your computing needs well into the 1990's.

A question I often ask computer salesmen is who buys your computer, and why.

The reason I ask this question is that despite what you tell me on your survey forms each year, I don't believe that the home computer has found a niche in the home yet.

I give you that some use them to play games, I give you that many of you have made them a hobby - and a great one at that - and I can see them being used to assist children in their schooling - I can even see people genuinely using them to gainfully assist themselves in what they do at work.

But as a unit which has a job to do in the home like a car or a fridge - until now, that has been harder to see.

After the hardware has been purchased, Viatel is cheap and very easy to run.

It provides a very broad range of tasks - mainly in the area of information provision.

And it is getting bigger every day.

In our Goldlink magazine on 642, we have price lists from the major suppliers of Tandy hard and software, including Tandy themselves.

We also have a magazine for Tandy owners which has a lot of useful pieces of information - peeks and pokes - getting started - Users' Groups and a notice board for them - and a Bulletin Board where you can ask for advice and help, or just give cheek to Martha.

Then we have a Florist (!), a leisure magazine with fishing hints and tips - fishing club news, info for those of you interested in geneology, touring information, info for those interested in trains, or amateur radio, a special bulletin board on which you can voice your opinion on the TV you are seeing, and there is much more!

There are a zillion reasons to move to Viatel - see it at CoCoConf, or ask your local Tandy store to show you, then come on in to the Goldlink world on Viatel!

REVIEWS

Software

XWord, XSpell and *XMerge* are Xtra Good

by Larry Goldwasser

XWord, XSpell, XMerge is a complete word processing system for OS-9 users. The X series consists of four free-standing programs for use under the OS-9 operating system on the Color Computer.

The first program is *XWord*. It contains both *XEd*, a text editor, and *XPrint*, a text formatter and printer driver. The major advantage to having these programs separate is the ability to use *XEd* to process OS-9 procedure files. I have never liked the editor supplied with OS-9 and I believe that others will want to use *XEd* too.

XEd must be installed in the current execution directory via the copy command. If you use only one drive (always a problem with OS-9) you must first load Copy and then proceed with a single drive copy.

In addition to *XEd*, a screen driver must be installed. The screen drivers supplied are *XCodes*, the normal CoCo screen, .OPAK for *O-PAK*, .XS for *XScreen*, and .XP for *Wordpak*. All of the drivers automatically adjust for the screen size you are using.

To start using *XEd*, type XED followed by either the filename you want to edit or the OS-9 memory option. At startup there is approximately 4½ K. You are greeted by a blank screen with a status line on top. The status line gives tabs, overwrite or insert mode, wordwrap status and cursor position.

With *XEd*, carriage returns are visible allowing the user to see where lines will end. *XEd* uses the CLEAR key as a control key and commands are entered as some variation of the control key plus at least one other key. The arrow keys are used to move the cursor non-destructively around the document.

The authors chose to stay with the OS-9 standard Clear A for insertion and Clear D for deletion instead of remapping the keyboard. For regular OS-9 users this should cause no problems. Most of the commands are accessible through the Clear C command mode entry. Command mode commands are: ?, H, T, B, F, R, V, A, X, U, K, O, D, I, W, Q, Z, P, S, C, M, L, N, G and E. Obviously, there is not room in a review to cover all these, but I will try to cover the most important ones.

Command		Function
?	—	list available command mode commands
H	—	list edit mode commands and explanations
T	—	move cursor to top of text (in buffer only)
B	—	move cursor to bottom of text (in buffer only)
F	—	find a string
R	—	find and replace a string with another string
V	—	find and verify a string before replacing it
A	—	find and replace a string again
X	—	set tab
U	—	unset tab
K	—	kill all tabs
O	—	execute an OS-9 command and return
I	—	toggles overwrite mode on and off
W	—	toggles wordwrap mode on and off
Q	—	quit editing
Z	—	abort editing (does not save work)
P	—	quit editing and print
M	—	get more text to edit (from file)
G	—	get file (file insertion)
E	—	define user programable keys ('K', 'N', 'O' and 'U')

This is not a complete list of the available commands but rather the ones I used the most. There is a whole set of block commands which I have ignored but are very useful. One note I should make is that the Q command automatically saves current work.

XPrint is the text formatter and printer driver supplied with the *XWord* system. It consists of three files: XP, Index and INIT.?. The question mark stands for the abbreviation of your printer. *XPrint* includes several printer drivers as well as the information necessary to change one of these to your printer.

When you have finished modifying the file, it must be copied into the execution directory as INIT.XP and the attribute of this file must be changed to allow it to be executable. This is done by typing `ATTR /d0/cmds/INIT.XP E`.

If you use more than one printer, you have the option of calling the init file wanted on the *XPrint* command line. You cannot do this if you call *XPrint* from *XEd* with the Clear C P command.

Most of the commands *XPrint* uses are entered on a format line. A format line starts with a period and calls the formatting options overriding the format options specified in the init file. These format options can be included in the command line or they can be in the text file itself (preceded by a period which must be the first character in the line).

This may seem a little complicated, but that's the price for the ability to change virtually any formatting option. These options include:

Margin (may be different for odd or even number pages)
- TM — top
- BM — bottom
- LM — left
- RM — right
- PL — page length (must be the same for even and odd pages)

Line Format (may be different for odd or even pages)
- L — left justified
- C — centered
- R — flush right
- J — justified (except those ending in carriage returns)
- A — all lines justified
- LS — line spacing
- PG — end page and force new page
- OP — force end page and start at top of next odd page
- EP — force end page and start at top of next even page
- PP — page pause after printing

SP — force specified number of blank lines to be printed
NL — force specified number of lines to be printed in a block
FI — include file specified in printing

The list of available options is quite lengthy and if there are any omissions, I couldn't find them. Both header and footer insertions are supported and the page number may be included in either. Page numbers may be in either Arabic or Roman numerals. Tabbing is fully supported and this option includes the headers and footers.

Printer commands include font changes, print quality changes, underline, double strike, size of characters and spaces, width changes, superscript and subscript, italics and so forth.

If something needs to be changed in the printing of the text you may imbed a special instruction within a back-slash (\) character. This printer code is not limited to the printer presets specified in the init file, but may be any special printer code chosen by using \z(printer code)\.

XPrint also uses the program module *Index* to store, in a file, words and the current page number the user wants to save for later printing as an index page. This file prints three inches wide allowing two columns per page. There are many options which I have omitted.

All the options mentioned worked well and exactly as described. I had to make extensive changes to the closest init file to my printer. I did this without difficulty by consulting the printer code summary from the printer users manual.

The documentation supplied with *XWord* is excellent. The printing and binding make the documentation easy to use. The index and table of contents simplify looking up a specific topic. In short, the seventy-five pages give the user all the information needed. I felt the section on *XEd* was easier to use than the section on *XPrint*.

At first I was puzzled about where to use the format lines in *XPrint*. The examples given helped me to use the program, but still left me with questions. I had to do some experimentation to find out how some sections really worked.

The major complaint I have is not really a program fault. I do not have *O-PAK*, *XScreen*, or *Wordpak 80*. This forced me to use the *XCodes* 31-character screen. The cursor is very busy in *XEd* and if you watch the screen it is annoying. One thing which would make the program easier to use is a repeat key, at least for cursor movement.

In summary, this is a very effective and complete word processing system. I would recommend *XWord* to anyone who uses OS-9 on the Color Computer.

The third program I reviewed is *XSpell*, a spelling checker for text files. It can also detect obvious typographical errors which would result in nonsense words. *XSpell* is called from the execution directory. The user may switch disks if necessary to get the file to be checked in the drive.

The program then prompts for the name of the file to be checked. Give the name of the file including any pathnames necessary. *XSpell* opens the file and asks whether you want to consider as a word any group of characters enclosed by spaces or carriage returns (A) or only consider reasonable words (S). Using the 'S' selection eliminates any word or group with numbers, etc.

Generally, the user would choose selection 'A'. *XSpell* reads the entire file and constructs a list of all the words in the document. After digesting the file, the program asks for the name of the dictionary to use. This dictionary can be DICT (20,000 words), or DICT40 (40,000 words), the two dictionaries supplied with the program, or it could be a dictionary of your own creation.

After the dictionary file is opened, the program asks if you intend to update the dictionary with any new words it encounters. You also are given the option of starting a dictionary file with new words. These options are important if you plan to use the speller with technical reports or legal forms. Each time the program encounters a new word it requests a disposition on the word. The choices are: D, add all words to the dictionary; A, add this word to the dictionary; I, ignore this word and go to the next; M, mark this word as being incorrect; X, mark all words as incorrect; and Q, quit and return to DOS. Options A and D are not valid if you have previously selected not to add words to the dictionary.

After all unknown words have been processed, *XSpell* asks if you want to write a new text file. If yes, the program asks whether corrected or marked. Selecting Marked causes the entire text file to be rewritten with the questionable words marked by three asterisks trailing the word. If you select Correct, the text file is rewritten and as the program encounters each of the words you have selected as wrong, it prompts you to correct it. If the word was actually correct, you may choose to ignore the word and go on or mark the word to be looked at later.

One of the nicer features of *XSpell* is the ability to create its own dictionary. This can be created in one of two formats, either full format or compressed. The full format is simply a text file with words in it followed by a carriage return. The compressed mode offers the advantage of reduced disk space requirements and easier, quicker handling. Its main disadvantage is the relative difficulty for editing.

The obvious solution is to keep two dictionaries, one in full format and the other an exact copy except in compressed form. In either form the dictionary must be sorted in alphanumeric order making the easiest method of creating the program *XSpell* itself, which has a module designed to do this specific task.

When the user starts the program, he is asked if the new words are to be saved into a new dictionary or added to the old dictionary. This process creates a new dictionary or enlarges the current one. An *XSpell* dictionary may span several disks; the program prompts the user when to switch disks.

I found *XSpell* easy to use, but somewhat disappointing. Other spellers I have used offer the user the closest correct spellings it has on file. The user may select one of these spellings to be substituted or the user may correct the word from the keyboard. *XSpell* gives no such alternatives. The user must enter the correct spelling from the keyboard without benefit of the program's own dictionary. This forced me to have two dictionaries, one on disk and one in my hand.

One other problem I encountered with the *XSpell* program is its inability to distinguish imbedded control commands unless they are separated from the text by spaces. When I failed to use both leading and trailing spaces, the program presented these codes as words and prompted me for a disposition. Neither of these problems was very serious, but both were annoying.

XMerge is an upgrade to the *XPrint* module included in the *XWord* package. *XMerge* supports all of the formatting and printing commands available with *XPrint* and adds a few new ones of its own. They include: RP, Repeat text file X number of times; MF, Merge file name; RV, Read variables; PV, Prompt for variables; and SV, Set variable equal to string.

XMerge, like *XPrint*, must first be copied into the current execution directory. The program is then called with any command line options desired. *XMerge*, through the use of its special merge commands, allows the user to take variables from one file and combine them with the text from another file as it is printed.

This is the mechanism by which those personalized computer generated letters are made. This is also the way a mailing list is merged into a form letter. *XMerge* makes this process very easy. The user simply makes a form letter as a text file and includes, within square brackets, a variable name such as Name. The user must also make a merge file with a list of names. When *XMerge* encounters the bracketed variable, it looks at the merge file and selects the first name and inserts it. On the next form letter it takes the next name and so on. The user may have many such variables and the merged variable may be any combination of letters, numbers, or symbols up to eighty characters. All must be on one line, ended by a carriage return. The variable must be declared in a format line and the merge file must have been opened by the MF= command.

XMerge also allows the user to stop at each variable and prompt for keyboard input to be inserted in the text file. This is the .PV option. Something to remember: All files to be handled must be in ASCII.

The documentation for *XMerge* is complete and easy to understand. The examples given by the authors are clear and make the program much easier to use and understand.

As a general summary, the X series is a fine program group. If it has one weak link, it is *XSpell* which is not as full-featured as the other modules. I would rate *XSpell* as good and the other modules as excellent. If you use OS-9 on the Color Computer, you could not make a finer purchase than *XWord, XSpell, XMerge*. I recommend it highly.

Software

Zork I is Out of This World

— Bruce Rothermel

Ready for a trip to a place where the only limitations are those created by your mind? A dark, magical place where wonderful and sometimes horrible things happen? A place inhabited by elves, gnomes and other mysterious creatures? Then come to the Great Underground Empire of Zork.

Transportation to this place is via your imagination, the Color Computer and *Zork I*. What is *Zork I*? Infocom, the creator of Zork, refers to it as an interactive fantasy.

For those not familiar with interactive fantasies, they are stories in which you play the main character. Your own thinking and imagination determine the principal character's actions and guide the story. You are presented with a series of locations, items, characters and events. You interact with the story in a variety of ways: moving from place to place, becoming familiar with your surroundings, exploring locations and reading descriptions.

An important element of interactive fiction is puzzle solving. Encountering a locked door or a ferocious beast is a challenge to be conquered using certain items found by careful exploring as you travel.

The fun of *Zork I* is in surmounting these obstacles, finding fabulous treasures, avoiding being eaten by exotic creatures and solving diabolical puzzles in the Underground Empire.

Instructions are entered from the keyboard. The program tells where you are, reveals anything obvious you should know about the situation (the subtleties are for you to discover) and waits for a response.

A typical situation screen might read: "You are on a small, rocky beach on the continuation of the frigid river past the falls. The beach is narrow due to the presence of the white cliffs. The river crosses over the falls to the east and a narrow path continues to the southwest."

What you do next controls how the game interacts with you. You could choose to dig for treasure on the beach, continue on the path or sit down and have a beer. Sometimes what you want to do is not in the program, and you are given a message like "I don't know the word beer" or "that sentence isn't one I recognize."

However, it is amazing what the program will respond to. Once, when I was frustrated with a particular situation, I jokingly typed in SCREAM. Zork came back with a very satisfying "Aarrggghhhh!"

Conversations with *Zork I* are often bizarre. If you decide to put on the green calico hat, the response could be, "the munchkins giggle, but remain unconvinced that you're a witch." *Zork I* is also very flexible in the ways a command can be worded. Light Lamp, Turn on the Lamp, Turn the Lamp on, Activate the Lamp and Light the Brass Lantern all result in turning the lamp on.

There is more than one solution to this game. Measure your progress at any time by asking for the score. It increases as you solve puzzles, perform certain acts, visit certain locations and gather the various treasures hidden in the crevices of the Great Underground Kingdom. You also get points for putting treasures in the trophy case. You lose points if a thief steals the treasures or you are killed.

At any time, you have the option of saving your current position. This is highly recommended as there are times when rash moves result in sudden death. However, reincarnation is as simple as reloading the game to a position before your foolhardy move.

Saving is also necessary because *Zork I* is a long, complicated game. To master *Zork I* takes a lot of time, experimentation with different strategies and some plain dumb luck. Infocom will provide (sell, that is) an official hint book and map of the lands of Zork for those hopelessly caught in a quagmire.

The hint book is a jewel. The clues are revealed by rubbing the answers with a magic pen. Hints are often given in stages, in hopes the Adventurer will use his head rather than the book to solve the riddles.

Zork I is packed in a box that has enough teasers and goodies on it to thrill the hearts of Adventurers of any age. It is a book-type package containing the program disk, a storybook with the history of the Underground Empire, an operations guide and a nifty map written in an encrypted language.

Originally written for other computers, *Zork I* does not take advantage of all the CoCo's capabilities: no graphics, no sound, no speech. You have to use your imagination to visualize the game.

Zork I is a well thought-out, thoroughly debugged joy of a program. It requires a 64K Color Computer with at least one disk drive. If you are yearning for an Adventure trip to somewhere a little different, *Zork I* is for you.

Easy Gradebook and *Easy Testwriter* have Potential

By Dennis Church

Easy Gradebook attempts to ease the complexity of recording and averaging grades for the classroom teacher. It is supplied on either tape or disk and facilitates saving and loading to either format.

In general, *Easy Gradebook* allows the entry of students' names and grades to compile a class list, the saving of that file to disk or tape, and reporting to either screen or printer. The class list file may be loaded, viewed and ordered alphabetically or by class average.

Actually there are two grade averaging programs. One accepts grade input in the form of 1, 2, 3, 4 and 5 for F, D, C, B and A. A second program accepts percentage grades.

Each of these assumes certain levels for letter grade equivalents to numerical averages. To change them, the author suggests either editing the program (he gives the line numbers to edit) or returning the program with a letter of explanation. He offers to do the editing at no charge.

Easy Gradebook is obviously a more developed program than *%Grade*. *Easy Gradebook* offers Baud rate selection (9600 or 600), repeated onscreen memory updates and the ability to select only failing averages (called smokups) for reporting.

Both *Easy Gradebook* and *%Grade* present menus of 10 items to the user:

1) Student Averages presents the names and averages to either the screen or printer.

2) Add A Student is the routine for creating the student list, and can also add more students to the current list.

3) Edit A Student Record allows changing the student name, changing any previous grade or adding grades.

4) Order sorts the class lists either alphabetically or by average.

5) Print allows printing the class list, grades and averages to the screen or printer. Single students may be selected for this feature as well.

6) Delete A Student does just that.

7) Print Subject Average reports the combined average grade of a particular class.

8) Redo A Student's Grades is designed to give the entire class a new set of grades, which is useful for averaging grades of the same class for a different subject or starting a new grade period with the same students.

9) Load File/Restart/Main Menu. It loads class lists, restarts the program (necessary if memory already contains a class list) and gives access to a main menu from which disk users can start one of three programs: *Easy Gradebook*, *%Grade* or *Easy Testwriter.*

10) Save File saves class lists and averages to either tape or disk.

The overall performance of the grade averaging programs is acceptable, especially for only $15. But they show a lack of polish and flexibility that teachers really need in software. Beyond the main menus, answering queries is sometimes accomplished by spelling the entire word, other times only the initial is necessary.

Some answers are indiscernable without listing the program. For example, choices for Baud rate are "1 or 87," but the program expects a '1' or '2' as the answer.

Editing a student's grade requires selecting the grade by number, but the screen formatting does not make that clear. Also, when editing a grade, the screen presents the grades only, without the student's name to confirm you are altering grades for the right student.

The three menu items that can lead to printer output use different prompts, which makes learning the program unnecessarily complicated. Startup of *Easy Gradebook* reports memory size, but it flashes by too quickly to read. When entering grades, however, it is reported after each entry, which is unnecessary since it does not change after each grade entry.

Loading a file to *Easy Gradebook* requires answering six or seven different queries. While the option to see the directory is useful, it is not made available on all disk access options.

For some reason, printing the students' grades produces in duplicate, two across, both to screen and printer. A better format is produced by a different print option.

Printouts that are longer than a page will not skip the 11-inch perforation. There *will* be long printouts, too, because from three to five lines are devoted to each student, depending on whether you choose to see just the averages or the grades, too.

On the positive side, all print options give the choice of screen or printer. Response to prompts is speedy and improper answers to prompts do not hang up the program. Viewing students' averages on the screen makes use of the arrow keys to progress forward or backward through the list. Grades entered as numbers are converted to letters automatically.

The ability to use percentage grades actually requires a second, modified version of the *Easy Gradebook* program. Both programs are included for the same price. Some of the quirks of *Easy Gradebook* are not present in *%Grade*, but it operates basically the same way. Disk users can select either program on startup so it would be easy to use both kinds of grade systems, which you may want to do for different subjects.

Documentation is five sheets of typing paper in a plastic binder. It is a sparse explanation of all the menu items, some advisories about using the program and the details of changing the program to accommodate a different grading scale. There is, however, an additional source of documentation: a VHS format video cassette available for $10. The tape shows the screen while the software's author explains and demonstrates each option. The demonstration lasts fifteen minutes and is clear and informative. In addition, the tape also demonstrates the use of a second program available from this author, *Easy Testwriter.*

I can recommend *Easy Gradebook* if the buyer is willing to accept the program's flaws in exchange for its reasonable price. The potential buyer should realize it is not easy to add grades to an existing class list. It is not feasible to enter class grades until you want the final average.

I want to encourage the program author to add polish and flexibility. Then his program would appeal to a wider variety of teacher needs.

I cannot recommend, however, a second program, *Easy Testwriter*, which is designed to produce printed multiple-choice tests. It is available, along with *Easy Gradebook*, for $25 or $15 separately. Disk users with both programs may select either from an opening menu.

Easy Testwriter designs a multiple choice test which may be output to the printer and saved to disk. It also prints an answer key for the teacher. An interesting and potentially useful feature is the ability to combine previously saved test

questions into a final exam.

Menu selection is straightforward and clear. Entering test questions, and correct and incorrect answers is clear. Viewing the questions after creation or loading is easy and quick. Editing the question is a valuable option, but the prompts are not at all clear when editing.

Easy Testwriter is flawed in the following ways. No provision is made for tests longer than one page. Skipping for page perforation must be taken care of outside the program. The Baud rate option, which the gradebook program has, is not present here.

The documentation says that true/false questions may be created, and I attempted three different two-answer tests. The program logic takes care of randomly placing the answers, but my tests always came out with the correct answer first, except for one question. This is not good pedagogy. Kids become more interested in the answer pattern than the question material after a few answers.

In creating a multiple choice test, the program performed well until I exercised the edit option. Then I got answers that appeared in the wrong question.

When attempting to merge tests to compose a final, some question lines were blank while other questions appeared as answers. Viewing before printing was no help since this did not show up on the screen version of the questions and answers.

As is the case with *Easy Gradebook*, this program has great potential if its flaws can be fixed.

Software

The Witness — A Classic Whodunit

— John McCormick

Had enough of dungeons and spaceships? Need something that your spouse/parent can enjoy on the CoCo? Then try *The Witness*.

You have just 12 hours to solve a murder that may not even have happened yet, or face the displeasure of your captain.

The clues? A matchbook, a newspaper clipping, a telegram and a possible suicide note. The suspects? A sinister Oriental butler, a very independent rich girl, a business man, his dead wife's lover and others. Who did it? For that matter, who was the victim? If you're not careful, it could be you!

Infocom is well-known for involved interactive fiction programs which have been available for more expensive computers for several years. Now they have brought their experience to our favorite machine.

This program is flawlessly packaged with great documentation. One little extra I really appreciated was the resealable box that can hold all the little pieces of evidence that would probably soon be lost.

The Witness is a 1930s-style whodunit. You and the faithful Sergeant Duffy have to interview suspects, analyze evidence and accuse the perpetrator within the 12-hour time limit.

At any point, you have the option to save the work you have done so far. This is a good idea because, if you get bumped off, it is necessary to start over unless your position was saved at an earlier point.

This is not just another maze search. In *Witness*, you must logically solve a problem using interviews and police methods.

The action can be livened up with unlikely moves on your part. They will provoke some interesting responses from the other characters.

I predict that even seasoned mystery readers will not solve this problem easily, and jumping to a conclusion early in the game will probably have you pounding a beat in the boondocks.

The game is played by making choices and giving directions to the computer. Your choices at any juncture are numerous and so are the number of possible story lines. Your location (such as driveway entrance) and the time (e.g., 8:02 p.m.) are continuously displayed and updated along the top of the screen.

Software

Formatting BASIC Listings with *Listaid*

— James Ventling

Listaid is a BASIC program that creates a machine language routine to format BASIC listings. It breaks up a line of BASIC so that each new statement starts on a new line and is indented. The minimum requirement is 4K Color BASIC 1.1. OK, I've seen a few of these before, let's try it.

The first thing it wants is a start location. There isn't a default value and none is given in the documentation.

The top of RAM for my machine is 32767 so I'll try 32700. Ah ha, the program didn't like that. It told me that number was too high. I'll try 32650. That works.

The program now wants to save the program. I choose disk and name it *Listaid*. Now I'll LOADM it and see if it works. Wait a minute. The instructions say I need to clear memory before executing the program. Let's see, CLEAR 200, 32649. Now to EXEC and list some BASIC code:

```
5 A1=0
 :B1=0
  :C1=0
   :D1=0
```

That worked fine. Let's try some more:

```
10 IF A1=1 THEN A1=2 ELSE IF B1=
1 THEN B1=2 ELSE C1=1
```

Oh, *Listaid* doesn't recognize the ELSE, only colons. Let's try something else:

```
20 AP$="VALUES CONTAINED IN VARIABLES
: A1, B1, AND C1 ARE TEST DATA"
```

Hmm, I didn't type all those spaces in the middle of Line 20. *Listaid* doesn't seem to know the difference between a colon that separates statements and a colon within a string.

I'd better check the EDIT mode. Good, the EDIT mode is unaffected by *Listaid*. With EDIT, I can see that Line 20 doesn't really contain all those spaces.

I wonder how to turn *Listaid* off. The instructions don't say anything about it. I think I'll try EXEC again. Bad news, typing EXEC a second time causes the computer to lock up and Reset doesn't help. That could be a big problem if you forget and do it again. You could lose all your code.

Now for the real test. Does it work with different printers? Great! It does work. Evidently the program must use ASCII.

Listaid works well onscreen or with any printer. The $10 price is very reasonable, too.

Software Bonanza Package: What You Need is What You Get

— David Gerald

A package deal of popular programs has been announced by Spectrum Projects that allows you to create your own Software Bonanza Package. You choose 12 programs from the following: *CoCo Checker, Multi-Pak Crak, CoCo Screen Dump, Disk Utility 2.1, Spectrum Font Generator, Tape/Disk Utility, Fast Dupe II, 64K Disk Utility, CoCo Calendar, Schematic Drafting Processor, OS-9 Solution, Graphicom, EZ Base, Black Jack Royale 2.0* and *Spectrum Dos 1.0*. This last program, *Spectrum Dos 1.0*, fixes a number of bugs inside the CoCo's ROMs, adds 24 new commands and gives the user a high-density screen of 32, 51 or 64 characters per line.

New commands are:
DOS — works just like the one in 1.1 Disk BASIC.
ERROR — Similar to an ON ERROR GOTO trapping routine.
FLEX — Boot up FLEX without the need of running a special boot loader.
RUNM — Single statement LOAD and EXEC of machine language programs.
PPEEK — Prints the 16-bit value of a specified memory location.
PPOKE — Stores a 16-bit value in a specified memory location.
AUTO — Automatically generate line numbers when entering a BASIC program.
INVERT — Reverse screen color.
NORMAL — Return to normal screen.
WAIT — A memory saving timer routine.
LMOVE — Copy and delete BASIC program lines.
RATE — Sets the drive seek rate for any or all drives.
TRACKS — Sets the number of tracks for any or all drives.
HELP — Lists all these commands onto a Hi-Res screen.
OLD — Restore a program that has been erased by NEW
FKEY — Define up to nine programmable keys.
LCOPY — Duplicate a BASIC line of code.
BREAK — Disable the BASIC key.
MEMO — A full text screen editor and screen dump to the printer.
FLIP — Invert the text screen.
EXIT — Return from Hi-Res to normal text screen.
ECHO — All output sent to the screen also goes to the printer.

New Features are:
35/40 track drives can be used.
Auto-disk search for all drives.
One button text screen dump.
One button loading of a BASIC program.
Lowercase Readable (commands can be in lower case).
Auto-key repeat.
New cursor (any printable character).
New prompt (anything you want).
Reset protected.

Fixed commands are:
DIR — Prints side-by-side directory on the screen as well as free granules.
DSKINI — Prints messages to let you know what it's doing.

All of the programs are individually packaged and contain ample documentation. Spectrum Products advertises this package for $99.95. That represents quite a bargain since the total package of 12 programs purchased individually would cost over $300. This is a good chance to increase your CoCo software library by choosing programs that fit your needs. Consult past issues of THE RAINBOW for further details and reviews of most of these programs.

Action and Challenge Define *Shock Trooper*

— Barbara Combes

Just when you think the game writers have run out of scenarios, along comes a program that restores your faith in the future of space games.

Such a game is *Shock Trooper*, created by Mark Data, a long-time pacesetter in innovative programming. The challenges are seemingly endless and the difficulty level is enough to frustrate even veteran game players. It is complete with sound effects and the nice assortment of color graphics that we've come to expect in top-of-the-line programs.

In *Shock Trooper*, you are the squad commander and have received word that aliens are planning to conquer Earth. Your mission is to infiltrate the alien base and destroy it. You also must escape with parts from their new TRG-5 space saucer and return them to Earth scientists for analysis.

Four highly-trained shocktroopers are placed under your command, each supplied with the latest in attack equipment — electro guns, inviso devices and porta-bombs.

The alien base consists of mazes containing perplexing enemy defenses. Among these defenses are lasers, which emit powerful energy bolts; rotating lasers, which can be stunned but not destroyed; zaproid robots, which enter through an opening in the ceiling; and forcefields.

Your weapons are useless in the filter grids of the tunnel sections. The most important areas of the base are protected by rows of ceiling-mounted emitters that drop radioactive particles.

One of your most important weapons is the inviso device. It makes you invisible and protects you from laser bolts, radiation particles and zaproids. However, anytime you use the device or another weapon, your radiation level is increased and is indicated by a bar meter at the top of the screen. You die if the level reaches a certain amount.

The aliens have captured many of your troopers and imprisoned them in special brain-draining chambers throughout the base. You must disable each chamber to prevent the aliens from obtaining vital secrets.

The game consists of 14 screens that become increasingly fascinating and more difficult to negotiate. This is a quality game in every respect. Obviously time and care have gone into it. If you're looking for a challenge, I recommend *Shock Trooper*.

Advanced BASIC Programming Aid and *Super Programming Aid* are Great Values

By James Ventling

All of us have wished at one time or another that Tandy had added certain commands and capabilities. Anyone who programs in BASIC knows the CoCo is limited in its built-in functions.

When the CoCo first appeared with only 4K memory and Color BASIC, lines with bugs had to be completely retyped. Line editing appeared with Extended BASIC and life was made easier.

With more memory, programs grew larger and harder to manage. Some programmers tried using word processors to write their programs but had to use ASCII files and couldn't test run a program while using the processor.

From time to time, utilities have appeared to try to take care of one shortcoming or another, but having all the various utilities and programming aids contained in a single program has not been accomplished — until now.

Bangert Software Systems has put together a group of programming aids intended to fill the void. *Advanced Programming Aid* is a compact 2,752-byte, relocatable machine language program. It can be used with either a disk- or tape-based system and works with any ROM version. You'll need Extended BASIC and 16K memory or more, (it does not take advantage of the extra memory in a 64K system).

Using *Programming Aid* is very simple. If you're not using Hi-Res graphics you may want to first PCLEAR 1, particularly if you only have 16K. Type RUN"APA" (or CLOAD and RUN for tape). The next thing you see is the banner message and you are ready to use *Programming Aid*.

Programming Aid does not get in your way; you can use the CoCo the same as before. It doesn't require or impose any special restrictions or formats. You could ignore it and never know it was there — but you won't want to when you discover how useful it is.

The commands and functions are entered by using two keystrokes. First the control key (down arrow) is pressed, then the key for the command of your choice. I consider it a real convenience to be able to use one hand to enter commands rather than have to put down whatever I'm holding in order to press several keys simultaneously. This is just one example of the friendliness of this program.

Let's look at the functions this program has to offer.

Automatic Line Numbering — When you select this command a line number appears on the screen waiting for you to type in a line of BASIC. When that line is entered the next line number automatically appears. You have the option of specifying the start and increment values for line numbers, though the program will not let you run over existing lines. If you reach an existing line, the automatic line numbering stops. Each time the automatic numbering is interrupted, the program remembers where it left off and begins there again unless otherwise instructed.

Automatic Loading of Menu (Disk only) — If you have a menu program written in BASIC, you can load and run it with just two keystrokes. A check is made to be sure there is no program in memory that would be erased. There is a very simple menu program included for demonstration but it can be replaced with any BASIC program of your choice.

Keyboard Clicker — Each time you press a key it makes a clicking sound. This is an option I find very helpful when typing in code from a magazine listing. The click sound lets you know if a key is missed or hit accidentally without having to look up at the screen. If you don't want the click option you can turn it off.

Suspend — This is a very exciting feature. This command lets you suspend, or hide the program you are working on. Your program will seem to have disappeared. Now you can check the listing of another program, type in a new program, run another program, or anything else you like, all without interfering with the hidden program. The RESTORE command brings your original program out of hiding. This is also an easy way to merge programs. While the original program is hidden, you can load and edit an additional program. Any BASIC code left in memory is appended to the end of the original program when it is brought out of hiding.

Copy — The Copy command copies a line or a group of lines in your program. You specify the line or range of lines to be duplicated and where in the program you want the copy to go. Line numbers are automatically changed on the copy. A check is made to be sure you don't copy over or erase existing lines.

Move — Similar to the Copy command, the Move command copies a line or group of lines to a new location in the program. However, the original group of lines is deleted.

Find — If you've ever had to search through a long listing for something, then you will appreciate this command. Find allows you to search through a program for the occurrence of any string up to 17 characters long. The search string is remembered, so after finding one occurrence, you can enter the Find command to find the next occurrence without retyping the characters you are searching for.

Termination — You can eliminate *Programming Aid* without affecting your BASIC program (to execute another machine language program for example). Though the manual doesn't mention this, while *Programming Aid* is active, its functions are carried over into the running of your BASIC program (such as Key Click). More about this later.

Program Scrolling — There are times when I want to look at a program line by line. Tandy didn't provide any way to do this. *Programming Aid* not only allows you to scroll through a listing one line at a time, but you can view the lines in forward or reverse order. In addition, the scrolling starts with the line last worked on. With the Scroll command you may never use the old LIST again.

Repeating Keys — You are able to repeat any key (including the backspace) by just pressing that key and then holding down the CLEAR key. I found this to be especially useful when editing a line.

Basic Program Formatting — When turned on, this option takes a line containing multiple statements and lists each of those statements on a separate line, indented several spaces. This works for both the screen and printer and is even active as you are typing in a line. As soon as you type a colon the cursor jumps to the next line. I find it useful when printing a hard copy to share with others, and beginners may find it helps when debugging a program.

Clear Key Disable — You have the option of turning off the usual CLEAR key function. Accidently clearing the screen can be most annoying, especially if you are typing in a long line of BASIC. This is fairly easy to do since the

CLEAR and the ENTER are side by side. With the CLEAR disabled you need never suffer this misery again.

Current Line Edit — This command automatically puts you in the Edit mode for the line last worked with. This saves time and typing since neither the usual Edit command nor the line number need be entered.

Command Keys — This is not a single function but a whole range of commands. You do have an option of whether or not to load the command keys table. It takes up approximately 280 additional bytes but it is worth it. By pressing the control key (down arrow) and then another key, a whole string of characters is automatically typed in for you. For instance, the letter 'A' gives you Audio O, 'B' gives Backup, 'C' gives Circle(, and so on. The command key table included contains 26 of these command strings (one for each letter on the keyboard). These command keys are easily remembered after a short time and save a tremendous amount of typing. A disadvantage is having all the keys predefined for you. For example, the 'Z' key gives CLOAD". If you have a disk based system as I do, you would probably want to replace the CLOAD" string with something more useful. This problem is overcome with *Advanced Programming Aid*'s sister program, *Super Programming Aid*.

Super Programming Aid contains all of the above plus a program to create your own command key tables and a new loader which asks which table you want to use with the programming aid. Not only can you make a command key table tailored to your own specific needs and habits, but you can make a number of different tables for different purposes and situations.

Menus and prompts make it easy to build or modify a command key table. You can define as many as 36 command keys and each key can be assigned a string up to 250 characters long. If you like, a command string can end with a carriage return (the ENTER key) so that a command will automatically execute.

Though not mentioned in the documentation, I found the features of *Programming Aid* (when active) are carried over into your BASIC program. When the BASIC program is running, you will find that Key Click, Key Repeat, Clear Key Disable and the Command Keys are in force and operating. This opens up some interesting possibilities. You could even create a command key table for program responses.

Of course, these programs are copyrighted and you can not sell or distribute any portion of them.

Advanced Programming Aid and *Super Programming Aid* are very modestly priced and the value is excellent. This kind of quality software needs to be encouraged. If you like the program, tell all your friends but please don't give away illegal copies. At this price, everyone can afford to buy their own. Let's support the development of fine software.

The manual is clear and detailed. It includes a table of contents, technical information and a command summary page. Each function is fully explained and examples are given.

Once you have read the manual you will probably never need to refer to it again. Onscreen prompts, status messages and error messages keep you from getting lost or confused. Also included with the program is a customer-comment survey and a postcard for asking any questions you might have.

Though my description may seem long-winded, the programs are very simple and easy to use. I could not find any bugs or problems. Everything works with convenience and style. This is the kind of utility that should be included with every CoCo sold. For usefulness and value, *Advanced Programming Aid* and *Super Programming Aid* have my vote for best buy of the year.

Software

Snap Study System Records Notes and Ideas

— Bill Tottingham

Cozy Software's *Snap Study System* keeps track of your mental ramblings in a rather elaborate file consisting of boxes, main files, subfiles and items. You have the options to save your ideas to disk, print a hard copy, and change or delete as needed.

From our sample file we learn that the subject box can be divided in up to eight main files. Each one of the main files can, in turn, have up to eight subfiles, which can have up to eight items. All this information can easily be entered or examined by using the arrow keys in a manner like many spreadsheet programs.

For example, if I wanted to keep track of jobs to do, I could first divide the Jobs box into up to eight parts. There could be one part for home jobs, one for at work jobs, one for moonlighting jobs and so forth. These are the main files. Now I can take the Home main file and divide it into eight more sections, such as in what parts of the home these jobs are located. And then, under these subfiles, I can enter up to eight items, such as what needs to be done. I can continue this for all of the main and subfiles if desired.

To view this at a later date, just select the Jobs file and scan through it.

It may sound like this program will take weeks to learn, but this is not the case. *Snap Study System* is extremely simple to understand. The program is written in BASIC, comes on a disk and requires 32K and one drive.

The documentation is a 13-page booklet that we found wanting. For example, if you follow the directions in the startup procedures, you won't be able to make use of the example files. Also, if you press the keys as you read along, a prompt asking if you want memory cleared appears a step or two before the documentation tells you about it. These are minor errors, and the rest of the documentation and the program itself are quite easy to understand.

My main complaint involves the Print Report instructions. Here you are advised that you are able to enter printer control codes for your particular printer, but no explanation follows. I was able to figure it out, but I thought it could have been more carefully written.

I believe the program could have allowed you to change the drive default and the printer Baud rate while operating, and I found it somewhat disquieting to use BASIC's KILL command to remove empty files from the disk. However, in all fairness, I must add that removing unused or empty files is not necessary.

Snap Study File ran well and I was unable to crash it. But the program seemed to stall occasionally after aborting a disk save, and the keyboard response is often slow when leaving the viewing screen to the menu.

To sum up, *Snap Study File* is an easy-to-master filer/outliner, but $19.95 may be a little steep for a program with these limitations.

OTERM is a Pleasing Telecommunications Environment

— Cray Augsburg

And to think I believed I would be without an OS-9 terminal package. As SIGop of the RAINBOW Color SIG, I often require some method of downloading OS-9 programs. In the past, I have downloaded files using a terminal program running under Disk BASIC. Then I loaded OS-9 and Xcopied the files to OS-9. If I had only known then what I know now.

OTERM is a terminal program running under the OS-9 operating system. More precisely, it is a very pleasing telecommunications environment. It is as though you are sitting in the captain's chair with universal control at your fingertips. The only annoyance with program operation I was able to find is that *OTERM* uses OS-9's CLEAR key as the control key. Most other programs I have used operate with the down-arrow key as the control key. Once I was used to the difference, the world was before me.

On entering *OTERM*, you are presented with an options screen. This lets you alter the various communications parameters to suit the system you want to contact. These options include, along with the expected parameters, the ability to clean the incoming data (control word-wrap) and to strip linefeeds from incoming data before sending it to a file or the printer.

Once you have chosen the parameters, you may save them to a file for future use. You may also load a previously saved options file. The next time you go into *OTERM*, you may select an options file from the same command line used to call *OTERM*. This can come in very handy when you have several sets of parameters you use regularly.

Just as *OTERM* allows creating options files, it allows creating function key definitions files. There are 256 bytes of string space which may be divided among function keys one through nine in any manner wanted. You can set up one key to cause the auto-dial modem to dial and set up another to issue your username for logging in. For the sake of security, don't assign one key to send your password. Who knows what your friends might do with that information.

From the function keys menu, which is accessible from the command mode, you may add, kill, or display the setting for a given key. You can also save and load function key files to or from the disk. This allows changing the function key settings while online, giving an unlimited number of function keys. These files, just as the options files, may be loaded from the same command line by which you enter *OTERM*.

Perhaps one of the most beneficial areas of *OTERM* is the command mode. Just press CONTROL-ALT (CLEAR-'@') and you are ready to issue a single-key command abbreviation. If you do not remember all of the command abbreviations, just press 'M' and a command summary appears on the screen. The only fault I find with this routine is that you must remember the abbreviation you were looking for, go back to communications and use CONTROL-ALT to get back into the command mode to issue the command. This only occurs if 'M' is used to view the summary.

The command mode allows you to issue shell commands to OS-9, go to the function key menu and generally control other aspects of your communications. You may even tell *OTERM* to send a copy of the screen to the line printer. Keep in mind, though, that its power is possible only because of the operating system under which it is running. The same features would be difficult, if not impossible, to include in a terminal program under Radio Shack Disk BASIC.

File transfers are easily accomplished while using *OTERM*. The program supports both the buffer-capture method and downloading using the Xmodem protocol. Also, the capture buffer allows you to use the DC2/DC4 transfer procedure for automatic buffer control. Manual buffer control, however, may be used.

You can, of course, print, save, load and transfer the contents of the capture buffer. An interesting option, however, lets you get the screen. If you issue the G command for the buffer, whatever appears on the screen is appended to the contents of the buffer. This can be very useful for taking down bits and pieces of information from different places on a communications network. It can also be a time saver if you happen to forget to open the buffer before reading something online. Just issue the command and the information will be in the buffer.

Xmodem uploading and downloading is quite simple to use. Just instruct the remote computer to prepare itself, go to the command mode and issue the Up or Down command. You are prompted for whether the file is in binary or ASCII format. Then you are asked for a filename (pathname) under which *OTERM* is to store the file. Then transfer begins. As the buffer is filled, *OTERM* saves the old contents out to disk, thus allowing very large files to be transferred. You even have an abort option for Xmodem transfers, which is a feature most other terminal programs don't have.

If you get tired of the old 32-column screen, *OTERM* may be configured to work with the Hi-Res program of *O-Pak* from Frank Hogg. This modification is well documented, very simple to perform and gives quite acceptable results. Also, for those who own *Wordpack II* from PBJ, if you have installed the OS-9 drivers, *OTERM* will use the 80-column screen.

My only gripe with the *OTERM* package is the documentation. The expert may have few problems, but the novice may become frustrated trying to wade through the manual. In general, the manual is well-written. However, it does not give the user any way to tie all the information it contains together. A complete command summary would be very helpful in mentally compiling the information. As an experienced user of telecommunications, I found it quite frustrating that I was afraid to boot up the package without

reading through the manual four times. It is true, however, that the program tends to pull the information together once it is running.

All in all, *OTERM* is an excellent package. Though it requires a Multi-Pak Interface and an RS-232 card, it includes more than enough features at a reasonable price. In my little black book, *OTERM* has four stars by its name.

Software

New and Improved: OS-9 Version 2.00

— Larry Goldwasser

OS-9 version 2.00 is an update package for earlier versions of OS-9. It consists of two disks (an updated system disk and a new boot/config disk) and an 81-page booklet describing the changes made to the original system disk as well as the new utilities.

The most obvious change in the package is the addition of the *Config* utility. This program allows you to create a new system disk with only the drivers and modules you choose. To use *Config*, boot OS-9, then change disks inserting the boot/config disk in /d0. At the OS-9 prompt, type `/d0/cmds/config`. The utility then displays a menu and asks if you want to use one or two drives. After answering, you are given a choice of devices to be included on the new system disk. The list of choices is quite extensive and includes some surprises:

/term 32	standard TV 32-column display
/term 80	optional 80-column video display
/d0	floppy disk Drive 0
/d1	floppy disk Drive 1
/d2	floppy disk Drive 2
/d3	floppy disk Drive 3
/h0-15	hard disk Drive 0, 15 meg
/h0-35	hard disk Drive 0, 35 meg
/h1-15	hard disk Drive 1, 15 meg
/h1-35	hard disk Drive 1, 35 meg
/p	printer using standard RS-232 port
/t1	terminal using standard RS-232 port
/t2	terminal using optional RS-232 pack
/t3	terminal using optional RS-232 pack
/m1	modem
/m2	modem
/ssc	speech/sound cartridge

After selecting the devices, the user must also select the I/O routines and the power line frequency (60 hertz U.S.A. or 50 hertz Europe). The user also chooses the command set for the new system disk: (N)o commands, (B)ASIC commands, (F)ull command set, (I)ndividually select commands, or (?) receive help on command choices.

After making the appropriate choices, the user is prompted to insert a formatted disk in Drive /d1 for a two-drive system or Drive /d0 for a one-drive system and the process completes itself except for disk swaps necessary on a one-drive system. I was a little disappointed to find the new system disk was not bootable from Disk BASIC, but that can be remedied using the *Cobbler* utility.

Another new feature is optional DOS Help. This consists of two files. The first file is the *Help* program itself. It must be copied from the boot/config disk CMDS directory to the /d0/cmds directory of the system disk. The *Help* program uses the data file CMDS.HP which must be copied from the boot/config disk to the /d0/sys directory of the system disk.

Once the *Help* files are in place, the user simply enters at the OS-9 command line `HELP` plus the command he wants help with. The program supplies information about the command and the necessary syntax.

The third new utility is *Iniz*. It is intended to initialize a port or other device at the start of an OS-9 session, forcing the allocation of buffer space before other programs use it. Forcing allocation early in the session prevents memory fragmentation which can rob the system of valuable RAM.

The program resides in the /d0/cmds directory. It is called, preferably from the Startup file, by *Iniz* device-names.

The fourth new utility, *Tuneport*, is my favorite. This program sets the loop counter for the serial port enabling the user to work at a higher Baud rate. *Tuneport* resides in the /d0/cmds directory. To call it, the user types `TUNEPORT /p` or `TUNEPORT /t1` to set the delay times for either the printer or the terminal.

The utility presents the current Baud rate, sends a test message to the printer or terminal and asks the user for a new value. If the message was successfully transmitted, the user replies with a carriage return. Otherwise, the user must supply a new value to try. This continues until the proper delay time has been established and the program ends with a carriage return.

If the user wants the change to be permanent, he must run *Cobbler*.

Several utilities have been changed in version 2.00. Most of the changes were made to accommodate the 80-column screen.

There have been other changes too. All keys are self-repeating. When a key is pressed for more than one second, it automatically repeats.

The '@' key now acts as the 'ALT' key. This adds 128 to the ASCII value of a key pressed.

The Getstat function now gives values for several key combinations that were missing.

FORMAT has been changed to allow almost unprompted formatting of disks.

And finally, OS9GEN has been changed to allow those with single drives to use it. In previous versions only *Cobbler* was available without two drives.

I received OS-9 version 2.00 as an addendum, not as a complete package. I don't know if it is available any other way. The documentation is quite good. The changed modules are examined in detail as are the new facilities.

Overall, I had trouble with only one instruction. The manual says to insert the config/boot disk in /d0 with the execution directory set to /d0/cmds and to type `CONFIG`. When I did this I got an error message. Typing `/d0/cmds/config` worked well.

If you don't own a previous version of OS-9, I recommend you try to get a complete package. The documentation is not meant to be a complete guide to the OS-9 operating system and the manual supplied assumes the user already knows the basics of OS-9 use.

I like OS-9 version 2.00 very much. I wish the authors had included an option to make bootable disks with the *Config* program. The *Tuneport* program worked like a champ and my printer now hums along at 9600 Baud. All of the programs written for earlier versions of OS-9 ran perfectly.

I recommend OS-9 version 2.00 highly. It is refreshing to find something new and improved.

VICIOUS VIC

by Jay R. Hoggins

VICIOUS Vic is a game of chance and strategy. It is not a shoot-em-up arcade-type game.

Your player is placed on a grid. Also randomly placed on the grid are electric fences, acid pots and, of course, the Vic fellows. Accumulate points in this game by destroying the Vicious Vic characters.

The first screen has eight Vics on it. If you are successful in clearing that screen, a new screen is generated with two more Vics than the previous screen had. Your turn is over when one of the Vics gets you or you bump into a pot or a fence.

Each time you move your player one step in any of the eight possible directions, each Vic moves one step towards you. You may take as much time as needed between each move. The Vics will not move until you do and no penalties to your score are assessed.

If you place your right hand on the keyboard in the normal touch typing position, your middle finger is on the 'K' key. This game allows you to move the player as if it is always at the 'K' key position. In order to move the player one step to the left, press the key directly to the left of the 'K'(the 'J' key). In order to move the player diagonally up to the right, press the key which is diagonally up to the right of the 'K'(the 'O' key), and so forth. To remind you which keys to use, the lower right corner of the game screen shows the player surrounded by the keys that may be pressed for the corresponding directions of movement.

Since the Vics are not terribly intelligent, they can be brought to an untimely end by moving your player so as to cause one or more Vics to collide with a pot or a fence. Do this to clear the screen of Vics while being careful not to step into a pot or a fence yourself.

Each time a Vic is electrocuted by a fence, you receive 11 points multiplied by the number of the screen you are on. Each time a Vic meets his end by stepping into an acid pot, you get 12 points multiplied by the screen number. If you are very clever, you can cause a Vic to bump into another Vic. This will net you 13 points multiplied by the screen number.

To maximize scoring, take the time to carefully plan your moves. This is particularly true on the higher numbered sreens since the number of points you receive is multiplied by the screen number.

If a Vic is electrocuted, both the Vic and the fence disappear. But if a Vic steps into an acid pot, only the Vic is distroyed and the acid pot survives to claim another victim. Take advantage of this by aligning your player behind an acid pot. Then you can move back and forth, causing the Vics to step into the pot one at a time. Be careful, though. One may step up behind you!

40	189	840	8
154	85	1070	37
158 ...	172	1190 ...	170
161	98	1350	10
165 ...	188	1510 ...	154
169 ...	128	1680	57
243 ...	244	1930	58
250 ...	238	2200 ...	165
400	0	2360	64
610 ...	215	2520 ...	229
672 ...	191	2670	11
730 ...	242	2769 ...	107
		END	244

The Listing: VIC

```
1 '*****************************
2 '                VICIOUS VIC
3 '  (C) 1985 BY JAY R. HOGGINS
5 '           1747 PATRICIA WAY
6 '        SALT LAKE CITY, UTAH
7 '                      84116
8 '*****************************
9 '
10 'INITIALIZATION
11 CLEAR500,&H7DFF:CLEAR1000
12 DATA 158,186,48,137,17,235,95
,79,111,128,139,1,129,4,38,248,4
8,136,28,203,1,193,40,38,238,57,
158,186,79,95,111,128,195,0,1,16
,131,17,160,38,245,57
13 FOR AD=&H7E00 TO &H7E19:READQ
:POKE AD,Q:NEXTAD
14 FOR AD=&H7F00 TO &H7F0F:READ
Q:POKE AD,Q:NEXTAD
15 DEF USR0=&H7F00:DEF USR2=&H7E
00:DIM NA(20),P(64),F(12),FA(20)
,HA(20),MA(20),VA(20),C$(37),EX(
20),VS(20),HH(20),HI(20)
30 GOTO153
38 '
39 'PRINT CHARACTERS TO HIRES SC
REEN
40 X$=STR$(XB):Y$=STR$(YA):W$="B
M"+X$+","+Y$:DRAW W$:FOR LP=1 TO
 LEN(N$):L$=MID$(N$,LP,1):L=ASC(
L$)
50 IF L>=65 THEN L=L-54:GOTO90
60 IF L=32 THEN L=10:GOTO90
70 IF L=8 THEN L=37:GOTO90
80 IF L<=57 THEN L=L-48
90 DRAW C$(L):NEXTLP:RETURN
99 '
100 'PUTS FENCE AROUND GRID
110 Y=2:FOR X=3 TO 243 STEP 10:P
UT(X,Y)-(X+7,Y+4),FA,PSET:NEXTX
120 X=243:FORY=8 TO 134STEP6:PUT
(X,Y)-(X+7,Y+4),FA,PSET:NEXTY
130 Y=134:FORX=243 TO 3 STEP-10:
PUT(X,Y)-(X+7,Y+4),FA,PSET:NEXTX
140 X=3:FOR Y=134 TO 2STEP-6:PUT
(X,Y)-(X+7,Y+4),FA,PSET:NEXTY
150 RETURN
151 '
152 'TITLE SCREEN
153 DATA 3,0,1,5,1,0,1,1,1,41,1,
55,1,50,1,15,1,50,1,5,1,55,1,40,
1,55,1,40,1,50,1,14,1,5,1,54,1,0
,1,14,1,0,2,5,1,55,1,40,1,55,1,4
0,6,0,1,5,1,0,1,1,1,41,1,55,1,50
,1,55,1,54,1,5,1,55,1,41,1,55,2,
50,1,14,1,15,1,55,1,0,1,14,1,0,2
,5,1,55
154 DATA 1,41,1,55,1,50,38,0,1,1
,1,40,1,5,1,0,1,15,1,0,1,50,1,5,
1,0,1,54,1,1,1,40,2,50,2,14,1,5,
1,0,1,5,1,0,1,14,1,0,1,54,1,1,1,
40,1,14,6,0,1,1,1,40,1,5,1,0,1,1
5,1,0,1,50,1,5,1,0,1,54,1,1,1,40
,2,50,2,14,2,0,1,5,1,0,1,14,1,0
155 DATA 1,54,1,1,1,40,1,14,39,0
,1,50,1,14,1,0,1,15,1,0,1,50,2,0
,1,54,1,1,1,40,2,50,1,14,1,15,1,
54,1,0,1,1,1,40,1,50,1,0,1,54,1,
1,1,40,8,0,1,50,1,14,1,0,1,15,1,
0,1,50,2,0,1,54,1,1,1,40,2,50,1,
14,1,5,1,55,1,0,1,1,1,40,1,50,1,
0,1,54
156 DATA 1,1,1,40,40,0,1,14,1,50
,1,0,1,15,1,0,1,50,1,5,1,0,1,54,
1,1,1,40,2,50,1,14,1,0,1,5,2,0,1
,51,1,40,1,0,1,54,1,1,1,40,1,14,
7,0,1,14,1,50,1,0,1,15,1,0,1,50,
1,5,1,0,1,54,1,1,1,40,2,50,2,14,
1,5,2,0,1,51,1,40,1,0,1,54,1,1,1
,40
157 DATA 1,14,39,0,1,5,1,40,1,1,
1,55,1,50,1,55,1,54,1,5,1,55,1,4
1,1,55,1,50,1,55,1,54,1,15,1,55,
2,0,1,15,1,0,1,5,1,55,1,41,1,55,
1,50,7,0,1,5,1,40,1,1,1,55,1,50,
1,15,1,50,1,5,1,55,1,40,1,55,1,4
0,1,15,1,50,1,5,1,54,2,0,1,15,1,
0,1,5,1,55
158 DATA 1,40,1,55,1,40,294,0,1,
4,23,0,1,4,7,0,1,5,1,40,22,0,1,5
4,7,0,1,1,1,50,21,0,1,1,1,50,8,0
,1,54,21,0,1,5,1,40,8,0,1,15,21,
0,1,5,9,0,1,5,1,40,20,0,1,15,9,0
,1,1,1,40,20,0,1,14,9,0,1,1,1,50
,20,0,1,54,9,0,1,1,1,50,20,0,1,5
4,10,0
159 DATA 1,54,20,0,1,50,10,0,1,5
4,19,0,1,1,1,50,10,0,1,54,8,0,1,
B,1,FF,1,FE,1,80,7,0,1,1,1,50,10
,0,1,54,8,0,1,3D,1,FF,1,FD,1,E0,
7,0,1,1,1,50,10,0,1,55,7,0,1,3,1
,FE,1,7F,1,F3,1,FE,7,0,1,1,1,50,
10,0,1,55,7,0,1,3F,1,FF,1,80,1,F
,1,FF,1,C0
160 DATA 6,0,1,1,1,40,10,0,1,55,
7,0,2,FF,1,FD,2,FF,1,F0,6,0,1,5,
1,40,10,0,1,15,6,0,1,1,2,FF,1,FD
,2,FF,1,FC,6,0,1,5,1,40,10,0,1,1
5,6,0,1,7,2,FF,1,FD,2,FF,1,FE,6,
0,1,5,1,40,10,0,1,15,6,0,1,F,2,F
F,1,FD,3,FF,1,80,5,0,1,5,1,40,10
,0,1,15
161 DATA 6,0,1,1F,2,FF,1,FD,3,FF
,1,C0,5,0,1,5,1,40,10,0,1,15,1,4
0,5,0,1,3F,2,FF,1,FD,1,DF,2,FF,1
,E0,5,0,1,15,1,40,10,0,1,5,1,40,
5,0,1,7F,2,FF,1,FD,1,DF,2,FF,1,F
0,5,0,1,15,11,0,1,5,1,40,5,0,3,F
F,1,FD,1,DF,2,FF,1,F8,5,0,1,15,1
1,0,1,5,1,40
162 DATA 4,0,1,1,3,FF,1,FD,1,DF,
2,FF,1,FC,5,0,1,15,11,0,1,5,1,50
,4,0,1,7,3,FF,1,FD,1,DF,2,FF,1,F
E,5,0,1,55,11,0,1,5,1,50,4,0,1,F
,3,FF,1,FD,4,FF,1,80,4,0,1,55,11
,0,1,5,1,50,4,0,1,1F,3,FF,1,FD,4
,FF,1,C0,4,0,1,55,11,0,1,5,1,50,
4,0,1,3F
163 DATA 3,FF,1,FD,4,FF,1,E0,4,0
,1,55,11,0,1,5,1,50,4,0,1,7F,3,F
F,1,DD,4,FF,1,F0,4,0,1,55,11,0,1
,5,1,50,4,0,1,7F,3,FF,1,DD,4,FF,
1,F0,4,0,1,55,11,0,1,1,1,54,4,0,
4,FF,1,DD,4,FF,1,F8,3,0,1,1,1,54
,11,0,1,1,1,54,3,0,1,1,4,FF,1,DD
,4,FF,1,FC
164 DATA 3,0,1,1,1,54,11,0,1,1,1
,55,3,0,1,1,4,FF,1,DD,4,FF,1,FC,
3,0,1,5,1,54,11,0,1,1,1,55,1,50,
2,0,1,3,4,FF,1,FD,4,FF,1,FE,3,0,
1,55,1,54,11,0,1,1,3,55,1,5A,1,A
B,4,FF,1,FD,4,FF,1,FE,1,AA,1,D5,
2,55,1,54,11,0,1,1,3,55,1,5A,1,2
7,4,FF,1,FD
165 DATA 5,FF,1,22,1,D5,2,55,1,5
4,11,0,1,1,3,55,1,58,1,8F,4,FF,1
,FD,1,DF,4,FF,1,88,1,D5,2,55,1,5
4,12,0,3,55,1,5A,1,3F,4,FF,1,FD,
1,DF,4,FF,1,E2,1,D5,2,55,1,50,12
,0,3,55,1,58,1,9F,4,FF,1,FD,1,DF
,4,FF,1,C8,1,D5,2,55,1,50,12,0,3
,55,1,5A,1,3F,4,FF,1,FD
166 DATA 1,DF,4,FF,1,E2,1,D5,2,5
5,1,50,12,0,3,55,1,58,1,BF,4,FF,
1,FD,1,DF,4,FF,1,E8,1,D5,2,55,1,
50,12,0,1,15,2,55,1,5A,1,7F,4,FF
,1,FD,5,FF,1,F2,1,D5,2,55,1,40,1
2,0,1,15,2,55,1,58,5,FF,1,FD,5,F
F,1,F8,1,D5,2,55,1,40,12,0,1,15,
2,55,1,5A,5,FF,1,FD,5,FF
167 DATA 1,FA,1,D5,2,55,1,40,12,
0,1,5,2,55,1,58,5,FF,1,DD,5,FF,1
,F8,1,D5,2,55,13,0,1,1,2,55,1,5A
,5,FF,1,DD,5,FF,1,FA,1,D5,1,55,1
,54,13,0,1,1,2,55,1,58,5,FF,1,DD
,5,FF,1,F8,1,D5,1,55,1,50,14,0,2
,55,1,5B,5,FF,1,DD,5,FF,1,FE,1,D
5,1,55,1,40,14,0,1,15
168 DATA 1,55,1,5B,5,FF,1,DD,5,F
F,1,FE,1,D5,1,55,17,0,1,1,5,FF,1
,FD,5,FF,1,FC,19,0,1,1,5,FF,1,FD
,5,FF,1,FC,19,0,1,3,5,FF,1,FD,5,
FF,1,FE,19,0,1,3,5,FF,1,FD,1,DF,
4,FF,1,FE,19,0,1,3,5,FF,1,FD,1,D
F,4,FF,1,FE,19,0,1,7,5,FF,1,FD,1
,DF,5,FF,19,0,1,7
```

```
169 DATA 5,FF,1,FD,1,DF,5,FF,19,
0,1,7,5,FF,1,FD,1,DF,5,FF,19,0,1
,7,5,FF,1,FD,6,FF,19,0,1,7,5,FF,
1,FD,6,FF,19,0,1,7,5,FF,1,FD,6,F
F,51,0,1,7,12,FF,19,0,1,F,12,FF,
1,80,18,0,1,F,12,FF,1,80,18,0,1,
F,12,FF,1,80,180,0,1,55,1,3,1,55
,3,0,1,15,1,40,1,D5
170 DATA 1,40,22,0,1,55,1,41,1,5
5,3,0,1,15,1,50,1,55,1,40,22,0,3
,55,3,0,1,15,2,55,1,40,22,0,1,15
,1,55,1,54,3,0,1,5,2,55,23,0,1,1
5,1,55,1,54,3,0,1,5,2,55,23,0,1,
5,1,55,1,50,3,0,1,1,1,55,1,54,24
,0,1,55,5,0,1,15,1,40,620,0,1600
,FF
200 PMODE4,1:SCREEN1,1:PCLS(0):A
D=PEEK(186)*256+PEEK(187)+&H02E0
201 FOR E=1 TO 861:READC,B$:FOR
AD=AD TO AD+C-1:POKE AD,VAL("&H"
+B$):NEXT AD,E
239 '
240 'INITIALIZE SCORES
241 N$(1)="JAY":N$(2)="BRADLEY":
N$(3)="BARBRA":N$(4)="BARBRA":N$
(5)="ANY NAME":N$(6)="WHAT NAME"
:N$(7)="THAT NAME":S(1)=2847:S(2
)=1698:S(3)=3145:S(4)=4542
242 '
243 'INITIALIZE STRINGS
244 A$(1)="THESE INSTRUCTIONS AR
E ABBRE- VIATED, FOR MORE COMPLE
TE INSTRUCTIONS PLEASE READ THE
ARTICLE PROVIDED IN THE RAINBOW.
"
245 A$(2)="the object of the gam
e-"
246 A$(3)="YOU ARE TRAPPED IN A
FIELD, SURROUNDED BY ELECTRIC FE
NCES.  A NUMBER OF CRAZY CHARACT
ERS (NICKNAMED VIC) ARE OUT TO G
ET YOU.  YOU MUST TRY TO DESTROY
 THEM BEFORE THEY DESTROY YOU.
WITHIN THE GRID ARE RANDOMLY PLA
CED ELECTRIC FENCES AND"
247 A$(4)="ACID POTS.  BOTH OF T
HESE ARE FATAL TO VIC.  HOWEVER
THEY ARE FATAL TO YOU TOO!  YOU
MUST MOVE WITHIN THE GRID"
248 A$(5)="AVOIDING THE POTS AND
 FENCES WHILE LURING VIC INTO TH
EM.  one note- IF VIC TOUCHES A
FENCE, BOTH THE FENCE AND VIC WI
LL BE DESTROYED.  BUT IF VIC TOU
CHES AN ACID POT, ONLY VIC WILL
BE DESTROYED."
249 A$(6)="how to play the game-
"
250 A$(7)="FOR EVERY STEP THAT Y
OU TAKE, EACH VIC WILL TAKE ONE
STEP TOWARDS YOU.  YOU MAY MOVE
IN ONE OF EIGHT DIRECTIONS.  THE
 KEYS ON THE KEYBOARD SURROUND-
ING THE K KEY MOVE YOU IN ONE OF
 THE EIGHT DIRECTIONS.  FOR"
260 A$(8)="INSTANCE, THE I KEY I
S DIRECT- LY ABOVE THE K KEY AND
 WILL MOVE YOU ONE STEP STRAIGHT
 UP THE SCREEN.  THE M KEY IS DO
WN AND TO THE LEFT DIAGONALLY FR
OM THE K KEY.  IT WILL CAUSE YOU
 TO MOVE DIAGONALLY DOWN TO THE
LEFT."
270 C$(0)="BM+0,+1D2BM+1,+1R1BM+
1,-1U2BM-1,-1L1BM+5,+0"
280 C$(1)="BM+1,+1D0BM+0,+3R2L1U
4BM+4,+0"
290 C$(2)="R3D2L3D2R3BM+3,-4"
300 C$(3)="R3D2L3R3D2L3BM+6,-4"
310 C$(4)="D2R3D2U4BM+3,+0"
320 C$(5)="R3L3D2R3D2L3BM+6,-4"
330 C$(6)="D2R3D2L3U4R3BM+3,+0"
340 C$(7)="R3D4BM+3,-4"
350 C$(8)="R3D2L3U2D4R3U4BM+3,+0
"
360 C$(9)="R3D4U2L3U2BM+6,+0"
370 C$(10)="BM+6,+0"
380 C$(11)="BM+0,+1D3U2R3D2U3BM-
1,-1L1BM+5,+0"
390 C$(12)="D4R2BM+1,-1U2D1L2R2U
1BM-1,-1L2BM+6,+0"
400 C$(13)="D4R3U1BM+0,-2U1L3BM+
6,+0"
410 C$(14)="R2BM+1,+1D2BM-1,+1L2
U4BM+6,+0"
420 C$(15)="D4R3L3U2R3L3U2R3BM+3
,+0"
430 C$(16)="D4U2R3L3U2R3BM+3,+0"
440 C$(17)="D4R3U2L1BM-2,-2R3BM+
3,+0"
450 C$(18)="D4U2R3D2U4BM+3,+0"
460 C$(19)="R3L1D4R1L3R1U4BM+5,+
0"
470 C$(20)="R3L1D4L2U1BM+6,-3"
480 C$(21)="D4U2R1BM+1,+1R0BM+0,
-2R0BM+1,-1R0BM+0,+4R0BM+3,-4"
490 C$(22)="D4R3BM+3,-4"
500 C$(23)="D4U3R3D3U4BM+3,+0"
510 C$(24)="D4U3R1BM+1,+1R1D2U4B
M+3,+0"
520 C$(25)="D4R3U4L3BM+6,+0"
530 C$(26)="D4U2R3U2L3BM+6,+0"
540 C$(27)="BM+0,+1D2BM+1,+1R1U1
R1D1U3BM-1,-1L2BM+6,+0"
550 C$(28)="D4U2R3BM+0,+2L0BM-1,
-1L0BM+1,-1U2L3BM+6,+0"
560 C$(29)="R3L3D2R3D2L3BM+6,-4"
570 C$(30)="R3L1D4L1U4BM+5,+0"
580 C$(31)="D4R3U4BM+3,+0"
590 C$(32)="D3BM+1,+0D1R1U1BM+1,
+0U3BM+3,+0"
600 C$(33)="D4U1R3D1U4BM+3,+0"
610 C$(34)="R0BM+1,+1D2R1U2BM-2,
+3R0BM+3,+0R0BM+0,-4R0BM+3,+0"
620 C$(35)="D1BM+1,+1D2R1U2BM+1,
-1U1BM+3,+0"
630 C$(36)="R3BM-1,+1L0BM-1,+1L0
BM-1,+1D1R3BM+3,-4"
640 C$(37)="C0D4R1U4R1D4R1U4BM+2
,0C1"
648 '
649 'PLAY THEME SONG
650 PLAY"T202L4GP200GP200GP200L8
.E-P20003L16B-P200":PLAY"O2L4GP2
00L8.E-P20003L16B-P20002L2GP200"
:PLAY"O3L4DP200DP200DP200L8.E-P2
00L16B-P200":PLAY"O2L4G-P200L8.E
-P20003L16B-P20002L2GP200"
658 '
659 'PRINT COPYRIGHT & WARNING
670 DRAW"C0":YA=144:XB=20:N$="CO
PYRIGHT 1985 BY JAY R HOGGINS":G
OSUB40:YA=152:XB=20:N$="WARNING"
:GOSUB40:YA=160:XB=20:N$="THE PR
OGRAMMER GENERAL HAS":GOSUB40:YA
=168:XB=20:N$="DETERMINED THAT T
HIS PROGRAM MAY":GOSUB40:YA=176:
XB=20:N$="BE ADDICTIVE"
671 GOSUB40:YA=184:XB=20:N$="USE
 AT YOUR OWN RISK":GOSUB40:DRAW"
C1"
672 FOR D=1 TO 2000:NEXTD
679 'CLEAR SCREEN
680 CLS(RND(8)):PRINT@192,""
688 '
689 'SET UP GRAPHICS
690 PMODE4,1:PCLS:PSET(97,56,1):
PSET(103,56,1):PSET(99,57,1):PSE
T(101,57,1):PSET(99,59,1):PSET(1
01,59,1):PSET(97,60,1):PSET(103,
60,1):GET(97,56)-(104,60),EX,G
700 FOR X=100 TO 102 STEP2:FORY=
40 TO 44 STEP4:PSET(X,Y,1):NEXTY
,X:FORX=98 TO 104 STEP6:FORY=41
TO 43:PSET(X,Y,1):NEXTY,X:GET(98
,40)-(104,44),HH,G:FOR X=98 TO 1
04 STEP2:FOR Y=41 TO 43 STEP2:PS
ET(X,Y,1):NEXTY,X:GET(98,40)-(10
4,44),HI,G
710 FOR X=98 TO 104:FOR Y=88 TO
90:PSET(X,Y,1):NEXTY,X:FOR X=100
 TO 102:FOR Y=87 TO 91 STEP 4:PS
ET(X,Y,1):NEXTY,X:PSET(97,87,1):
PSET(105,87,1):FOR X=100 TO 102:
FORY=88 TO 90 STEP2:PSET(X,Y,0):
NEXTY,X:GET(97,87)-(105,91),VS,G
720 PCLS:FOR X=100 TO 102 STEP 2
:FOR Y=40 TO 44 STEP 4:PSET(X,Y,
1):NEXTY,X:FOR X=98 TO 104 STEP
2:FOR Y=41 TO43:PSET(X,Y,1):NEXT
Y,X
730 XP=97:FOR X=XP TO XP+6 STEP6
:FORY=56TO60:PSET(X,Y,1):PSET(X+
1,Y,1):NEXTY,X:Y=56:PSET(XP+2,Y+
1,1):PSET(XP+3,Y+1,1):PSET(XP+4,
Y+1,1):PSET(XP+5,Y+1,1)
740 PSET(101,72,1):PSET(99,73,1)
:PSET(101,73,1):PSET(103,73,1):P
SET(101,74,1):PSET(101,75,1):PSE
```

```
T(99,76,1):PSET(103,76,1)
750 FOR X=98 TO 104:FOR Y=88 TO
90:PSET(X,Y,1):NEXTY,X:FOR X=100
 TO 102:FOR Y=87 TO 91 STEP 4:PS
ET(X,Y,1):NEXT Y,X:PSET(97,87,1)
:PSET(105,87,1):FOR X=100 TO 102
:PSET(X,88,0):NEXTX
760 GET(97,56)-(104,60),FA,G
770 GET(98,40)-(104,44),HA,G
780 GET(99,72)-(103,76),MA,G
790 GET(97,87)-(105,91),VA,G
800 GET(97,97)-(105,102),NA,G
808 '
809 'ASK IF PLAYER WANTS INSTRUC
TIONS
810 PRINT@192,"    DO YOU WANT I
NSTRUCTIONS?          PRESS Y
OR N"
820 Z$=INKEY$:A=RND(10):IFZ$=""T
HEN820ELSE1FZ$<>"Y"THEN1120
828 '
829 'PRINT INSTRUCTIONS ROUTINE
830 W$=A$(1):GOSUB850
840 FOR R=2 TO 8:W$=A$(R):GOSUB8
60:NEXTR:GOSUB1040:GOTO1120
850 CLS(RND(8)):X=2:PRINT@3," VI
CIOUS VIC INSTRUCTIONS ";
860 IF X>15THEN1020
870 IF LEFT$(W$,1)=" "THENW$=RIG
HT$(W$,LEN(W$)-1):GOTO870
880 IF LEN(W$)<=30 THEN A$=W$:GO
TO930
890 FOR Q=31 TO 1 STEP-1
900 A$=MID$(W$,Q,1)
910 IF A$=" "THEN920 ELSENEXTQ
920 Q=Q-1:A$=LEFT$(W$,Q)
930 B$=A$
940 IF LEN(A$)>29 THEN 970 ELSE
FOR P=LEN(A$) TO 29
950 B$=B$+" "
960 NEXTP
970 PRINT@X*32-31,B$;
980 X=X+1
990 IF LEN(W$)<=30 THEN RETURN
1000 W$=RIGHT$(W$,LEN(W$)-Q)
1010 GOTO860
1020 GOSUB1040
1030 GOTO850
1040 PRINT@482," PRESS ANY KEY T
O CONTINUE ";
1050 FOR D=1 TO 50
1060 IF INKEY$=""THEN NEXTD ELSE
 1110
1070 PRINT@482,"
          ";
1080 FOR D=1 TO 50
1090 IF INKEY$=""THEN NEXTD ELSE
 1110
1100 GOTO1040
1110 RETURN
1118 '
1119 'SET UP SCREEN FOR SHOWING
HIGH SCORES
1120 NL=8:MN=8:PCLS:SCREEN1,1
1128 '
1129 'SCORE SORTING ROUTINE
1130 Q=1:FOR T=1TO5:FOR C=T TO 6
:IF S(7)<=S(C) THEN S(7)=S(C):N$
(7)=N$(C):Q=C
1140 NEXTC:FORRB=Q TO T STEP -1:
S(RB)=S(RB-1):N$(RB)=N$(RB-1):NE
XTRB:S(T)=S(7):N$(T)=N$(7):S(7)=
0:NEXTT:GOSUB100
1148 '
1149 'PRINT HIGH SCORES
1150 N$="HIGH SCORES":YA=40:XB=8
2:GOSUB40
1160 N$=N$(1):YA=50:XB=70:GOSUB4
0
1170 N$=STR$(S(1)):XB=124+6*(6-L
EN(STR$(S(1)))):YA=50:GOSUB40
1180 N$=N$(2):YA=60:XB=70:GOSUB4
0
1190 N$=STR$(S(2)):XB=124+6*(6-L
EN(STR$(S(2)))):YA=60:GOSUB40
1200 N$=N$(3):YA=70:XB=70:GOSUB4
0
1210 N$=STR$(S(3)):XB=124+6*(6-L
EN(STR$(S(3)))):YA=70:GOSUB40
1220 N$=N$(4):YA=80:XB=70:GOSUB4
0
1230 N$=STR$(S(4)):XB=124+6*(6-L
EN(STR$(S(4)))):YA=80:GOSUB40
1240 N$=N$(5):YA=90:XB=70:GOSUB4
0
```

```
1250 N$=STR$(S(5)):XB=124+6*(6-L
EN(STR$(S(5)))):YA=90:GOSUB40
1260 S(6)=0
1268 '
1269 'ASKS FOR PLAYERS NAME
1270 N$="PLEASE TYPE IN YOUR NAM
E":YA=110:XB=28:GOSUB40:SOUND100
,1
1280 LA=0:N$="":NA$="":XB=185
1290 Z$=INKEY$:IF Z$=""THEN1290
1300 PLAY"O4T50BO3":IF Z$=CHR$(8
) THEN 1370
1310 IF Z$=CHR$(13) THEN 1390
1320 IF ASC(Z$)<65 OR ASC(Z$)>90
 THEN 1290
1330 N$=Z$:YA=110:GOSUB40:XB=XB+
6
1340 NA$=NA$+Z$:LA=LA+1
1350 IF LA=8 THEN 1390
1360 GOTO1290
1370 LA=LA-1:IF LA<0 THEN LA=0
1375 IF LEN(NA$)<=0 THEN 1440
1380 NA$=LEFT$(NA$,LEN(NA$)-1):X
B=XB-6:N$=Z$:GOSUB40:GOTO1290
1388 '
1389 'SET UP GAME PLAYING SCREEN
1390 PCLS
1400 IF NA$="" THEN NA$=" "
1410 N$(6)=NA$:N$=NA$:YA=148:XB=
10:GOSUB40
1420 N$=STR$(S(6)):XB=76+6*(6-LE
N(STR$(S(6)))):YA=148:GOSUB40
1430 N$="SCREEN":XB=10:YA=159:GO
SUB40
1440 N$=STR$((MN/2)-3):YA=159:XB
=100:GOSUB40
1450 N$="NUMBER LEFT":YA=170:XB=
10:GOSUB40
1460 N$=STR$(NL):YA=170:XB=76+6*
(6-LEN(N$)):GOSUB40
1470 PUT(126,144)-(132,148),HA,P
SET
1480 PUT(127,154)-(134,158),FA,P
SET
1490 PUT(127,164)-(131,168),MA,P
SET
1500 PUT(125,174)-(133,178),VA,P
SET
1510 N$="ACID":YA=144:XB=144:GOS
UB40
1520 N$="FENCE":YA=154:XB=144:GO
SUB40
1530 N$="YOU":YA=164:XB=144:GOSU
B40
1540 N$="VIC":YA=174:XB=144:GOSU
B40
1550 PUT(203,158)-(207,162),MA,P
SET
1560 DRAW"BM136,146R4BM136,156R4
BM136,166R4BM136,176R4"
1570 N$="U I O":YA=148:XB=191:GO
SUB40
1580 N$="J":YA=158:XB=191:GOSUB4
0
1590 N$="L":YA=158:XB=215:GOSUB4
0
1600 N$="M":YA=168:XB=191:GOSUB4
0
1610 DRAW"BM206,170D2L2R3U2"
1620 DRAW"BM218,170D1L1U1"
1630 DRAW"BM186,144R36D34L36U34"
1632 G=USR0(NL)
1638 GOSUB100
1639 '
1640 'PUT RANDOM FENCES ON GRID
1650 FOR F=1 TO 52
1660 X=RND(23)
1670 Y=RND(21)
1680 X=(X*10)+3
1690 Y=(Y*6)+2
1700 IF PPOINT(X,Y)<>0 THEN 1660
1710 PUT(X,Y)-(X+7,Y+4),FA,PSET
1720 NEXTF
1729 '
1730 'PUT RANDOM HOLES ON GRID
1740 FOR H=1 TO 10
1750 X=RND(23):Y=RND(21)
1760 X=(X*10)+4:Y=(Y*6)+2
1770 IF PPOINT(X-1,Y)<>0 THEN 17
50
1780 IF PPOINT(X+2,Y+2)<>0 THEN
1750
1790 PUT(X,Y)-(X+6,Y+4),HA,PSET
1800 NEXTH
1809 '
1810 'PUT RANDOM VIC'S ON GRID
1820 FORD=1 TO 32:P(D)=0:NEXTD
1830 FOR M=1 TO MN
```

```
1840 X=RND(23):Y=RND(21)
1850 X=(X*10)+3:Y=(Y*6)+2
1860 IF PPOINT(X,Y)<>0 THEN 1840
1870 IF PPOINT(X+1,Y+1)<>0 THEN
1840
1880 PUT(X,Y)-(X+8,Y+5),VA,PSET
1890 P((M*2)-1)=X:P(M*2)=Y
1900 NEXTM
1909 '
1910 'PLACE AND FLASH THE PLAYER
1920 X=RND(23):Y=RND(21)
1930 X=(X*10)+5:Y=(Y*6)+2
1940 IF PPOINT(X-2,Y)<>0 THEN 19
20:'CATCHES MAX'S AND FENCES
1950 IF PPOINT(X-1,Y+2)<>0 THEN
1920:'CATCHES HOLES
1960 PUT(X,Y)-(X+4,Y+4),MA,PSET
1970 FOR D=1 TO 20
1980 Z$=INKEY$:IF Z$<>""THEN2050
1990 NEXTD
2000 PUT(X,Y)-(X+4,Y+4),NA,PSET
2010 FOR D=1 TO 20
2020 Z$=INKEY$:IF Z$<>""THEN2050
2030 NEXTD
2040 GOTO1960
2050 PUT(X,Y)-(X+4,Y+4),MA,PSET
2060 IF Z$="U"THEN2150
2070 IF Z$="I"THEN2160
2080 IF Z$="O"THEN2170
2090 IF Z$="J"THEN2180
2100 IF Z$="L"THEN2190
2110 IF Z$="M"THEN2200
2120 IF Z$=","THEN2210
2130 IF Z$="."THEN2220
2140 GOTO1960
2150 XN=X-10:YN=Y-6:GOTO2230
2160 XN=X:YN=Y-6:GOTO2230
2170 XN=X+10:YN=Y-6:GOTO2230
2180 XN=X-10:YN=Y:GOTO2230
2190 XN=X+10:YN=Y:GOTO2230
2200 XN=X-10:YN=Y+6:GOTO2230
2210 XN=X:YN=Y+6:GOTO2230
2220 XN=X+10:YN=Y+6:GOTO2230
2230 IF PPOINT(XN-2,YN)=5 AND PP
OINT(XN-1,YN)=0 THEN 2280:'CHECK
 FOR VIC
2240 IF PPOINT(XN-2,YN)=5 AND PP
OINT(XN-1,YN)=5 THEN 2380:'CHECK
 FOR FENCE
2250 IF PPOINT(XN-1,YN+2)=5 THEN
 2430:'CHECK FOR HOLE
2260 PUT(X,Y)-(X+4,Y+4),NA,PSET
2270 X=XN:Y=YN:PUT(X,Y)-(X+4,Y+4
),MA,PSET:GOTO2470
2279 '
2280 'YOU HIT A VIC
2290 PUT(X,Y)-(X+8,Y+5),NA,PSET:
XN=XN-2
2300 PUT(XN,YN)-(XN+8,YN+5),VS,P
SET
2310 PLAY"T402L4GP200GP200GP200L
8.E-P20003L16B-P20002L4GP200L8.E
-P20003L16B-P20002L2G"
2320 PUT(XN,YN)-(XN+8,YN+5),VA,P
SET:FORD=1 TO 100:NEXTD
2330 PUT(XN,YN)-(XN+8,YN+5),VS,P
SET:FOR D=1 TO 100:NEXTD
2340 PUT(XN,YN)-(XN+8,YN+5),VA,P
SET:FOR D=1 TO 100:NEXTD
2350 PUT(XN,YN)-(XN+8,YN+5),VS,P
SET:FOR D=1 TO 100:NEXTD
2360 PUT(XN,YN)-(XN+8,YN+5),VA,P
SET:FOR D=1 TO 200:NEXTD
2370 GOTO1120
2379 '
2380 ' YOU HIT A FENCE
2390 PUT(X,Y)-(X+4,Y+4),NA,PSET
2400 PUT(XN-2,YN)-(XN+5,YN+4),EX
,PSET
2410 PLAY"T255AGAGAGAG":PUT(XN-2
,YN)-(XN+5,YN+4),FA,PSET:PLAY"AG
AGAGA":PUT(XN-2,YN)-(XN+5,YN+4),
EX,PSET:PLAY"GAGAGAGAGAGAG":PUT(
XN-2,YN)-(XN+5,YN+4),FA,PSET
2420 GOTO1120
2429 '
2430 ' YOU HIT A HOLE
2440 PUT(X,Y)-(X+4,Y+4),NA,PSET
2450 PLAY"T255ABCDEFG":PUT(XN-1,
YN)-(XN+5,YN+4),HH,PSET:PLAY"ABC
DEFG":PUT(XN-1,YN)-(XN+5,YN+4),H
I,PSET:PLAY"ABCDEFG":PUT(XN-1,YN
)-(XN+5,YN+4),HA,PSET
2460 GOTO1120
2470 'VIC MOVE ROUTINE (MN=NUMBE
R OF ROBOTS)
2480 FOR DQ=1 TO MN*2 STEP 2
2490 IF P(DQ)=0 THEN 2780
2500 IF NL=0 THEN GOTO2830
2510 XR=P(DQ):YR=P(DQ+1)
2520 QX=X-XR-2
2528 '
2529 'SET UP MOVE DIRECTIONS
2530 IF QX<0 THEN H=-10
2540 IF QX=0 THEN H=0
2550 IF QX>0 THEN H=10
2560 QY=Y-YR
2570 IF QY<0 THEN V=-6
2580 IF QY=0 THEN V=0
2590 IF QY>0 THEN V=6
2600 QX=XR+H:QY=YR+V
2608 '
2609 'CHECK TO SEE IF A VIC GOT
YOU!
2610 IF QX=X-2 AND QY=Y THEN PUT
(QX,QY)-(QX+8,QY+5),VA,PSET:X=XR
:Y=YR:XN=QX+2:YN=QY:GOTO2290
2618 '
2619 'SEE IF VIC HIT A VIC
2620 IF PPOINT(QX,QY)=5 AND PPOI
NT(QX,QY+1)=0 THEN 2630 ELSE 267
0
2630 PLAY"T3001CGCGCGCT203"
2640 PUT(XR,YR)-(XR+8,YR+5),NA,P
SET:NL=NL-1:S(6)=S(6)+10*(1.3)*(
(MN/2)-3):P(DQ)=0:P(DQ+1)=0
2650 IF NL=0 THEN GOTO2830
2660 GOTO2780
2668 '
2669 'VIC HITS A FENCE?
2670 IF PPOINT(QX,QY)=5 AND PPOI
NT(QX,QY+1)=5 THEN 2680 ELSE 272
0
2680 PUT(XR,YR)-(XR+8,YR+5),NA,P
SET:NL=NL-1:S(6)=S(6)+10*(1.1)*(
(MN/2)-3):P(DQ)=0:P(DQ+1)=0
2690 PLAY"T255AGAGAGAG":PUT(QX,Q
Y)-(QX+7,QY+4),FA,PSET:PLAY"AGAG
AGA":PUT(QX,QY)-(QX+7,QY+4),EX,P
SET:PLAY"GAGAGAGAGAGAG":PUT(QX,Q
Y)-(QX+7,QY+4),FA,PSET:PUT(QX,QY
)-(QX+8,QY+5),NA,PSET
2700 IF NL=0 THEN GOTO2830
2710 GOTO2780
2718 '
2719 'VIC HITS A HOLE?
2720 IF PPOINT(QX+3,QY)=5 AND PP
OINT(QX+3,QY+1)=5 THEN 2730 ELSE
 2770
2730 PUT(XR,YR)-(XR+8,YR+5),NA,P
SET:S(6)=S(6)+10*(1.2)*((MN/2)-3
):NL=NL-1:P(DQ)=0:P(DQ+1)=0
2740 PLAY"T255ABCDEFG":PUT(QX+1,
QY)-(QX+7,QY+4),HH,PSET:PLAY"ABC
DEFG":PUT(QX+1,QY)-(QX+7,QY+4),H
I,PSET:PLAY"ABCDEFGT2":PUT(QX+1,
QY)-(QX+7,QY+4),HA,PSET
2750 IF NL=0 THEN GOTO2830
2760 GOTO2780
2768 '
2769 'MOVE VIC WITHOUT HITTING A
NYTHING
2770 PUT(XR,YR)-(XR+8,YR+5),NA,P
SET:PUT(QX,QY)-(QX+8,QY+5),VA,PS
ET:P(DQ)=QX:P(DQ+1)=QY
2780 NEXT DQ
2788 '
2789 'UPDATE SCORE
2790 AC=USR2(NL):N$=STR$(S(6)):X
B=76+6*(6-LEN(STR$(S(6)))):YA=14
8:GOSUB40
2792 '
2793 'UPDATE SCREEN NO.
2794 N$=STR$((MN/2)-3):YA=159:XB
=100:GOSUB40
2798 '
2799 'UPDATE NUMBER OF VICS LEFT
2800 N$=STR$(NL):YA=170:XB=76+6*
(6-LEN(STR$(NL))):GOSUB40
2810 GOTO1960
2819 '
2820 'YOU CLEARED THE SCREEN
2830 AC=USR2(NL):N$=STR$(S(6)):Y
A=148:XB=76+6*(6-LEN(STR$(S(6)))
):GOSUB40:MN=MN+2:NL=MN
2835 N$=STR$((MN/2)-3):YA=159:XB
=100:GOSUB40
2837 N$=STR$(NL):YA=170:XB=76+6*
(6-LEN(STR$(NL))):GOSUB40
2840 FOR D=220 TO 230:PMODE3,1:S
CREEN1,1:SOUNDD,2:PMODE4,1:SCREE
N1,1:FORDA=1TO10:NEXTDA:NEXTD:FO
R D=1 TO 100:NEXTD
2850 GOTO1632
```

If you have problem tenants and they've gotten out of hand, you need to call . . .

The Evictor

By Paul Jensen

When the landlords can't get non-paying (or other) tenants out of the building they call you . . . the Evictor.

Your job is to push the tenants out the building's windows. (No more Mr. Nice Guy!) To do this, you (the white block) move the joystick left and right to push the tenants (which are blue) off the floor they are on. For every floor they fall, you get 50 points. However, these tenants have no wish to leave their homes, so they get out the old mega-laser-ray-cannon rifles (which, remarkably, only kill evictors) and shoot everywhere, hoping to get you.

These tenants know they are no match for you, therefore, they try to escape by descending to the ground floor. If one of the tenants reaches it, he unlocks the front door, lets everyone else out and you lose the game.

Being the disrespectful creatures they are, they prefer to descend by burning a hole in the floor to get to the one below. Whenever this occurs, it costs you 25 points, plus the 50 points lost for not being able to pitch them out of a higher window.

Needless to say, you prefer to take the elevator, which is located in the middle of the building. To use it, just walk onto it and point the joystick up or down.

At the top of the screen, a bonus clock ticks away. After you have disposed of all 10 tenants, you are given 10 points for each bonus point, and are then transported to yet another building to do some landlord's dirty work.

When you lose the three evictors, the game ends. If your score is in the top 10, you are asked for your initials. The 10 best scores (stored in the file TOPTEN/EVC) are saved with each game. The whole score save routine is located at Line 50000 and I would be happy if anyone else would like to use it with their own games.

To load the game, just type it in and save it on a disk. Then RUN it. It asks if your computer can use the speed-up POKE. Answer 'Y' or 'N' accordingly. Then it asks if you want to initiate the TOPTEN/EVC file. If this is the first run, answer with YES. This step only has to be taken once. What it does is erase the high scores on the disk.

After these two questions are answered, a title screen appears. Then, after a key is pressed, the building shows up, and some comments scroll at the bottom of the screen. (The graphics in *Evictor* are in SET/RESET format. I wanted to make the game in machine language but I haven't yet found a good ML random number routine. Does anyone know of one?).

Press a key again. You'll hear a few bars of *Basin Street Blues*, and you're off to the races.

I think this game could be used on a cassette system if the high score saving function was removed.

I hope you enjoy *Evictor*, and may you never get zapped by an ornery tenant.

The listing: EVICTOR


```
1 REM ************************
2 REM *                      *
3 REM *    E V I C T O R     *
4 REM *                      *
5 REM *    BY PAUL JENSEN    *
6 REM *    FEBRUARY 1986     *
7 REM *                      *
8 REM *    BASIN STREET      *
9 REM *    BLUES WRITTEN BY  *
10 REM*    SPENCER WILLIAMS  *
11 REM*    ARRANGED BY JOHN  *
12 REM*    EDMONDSON         *
13 REM*                      *
14 REM************************
100 CLEAR1500
110 CLS:INPUT"CAN YOUR COMPUTER
HANDLE THE    SPEED-UP POKE";A$:
IF LEFT$(A$,1)="Y" THEN HI=65495
:LO=65494 ELSE HI=32768:LO=32768
 ' POKE WHERE IT WON'T HURT ANYT
HING
120 INPUT"INITIATE 'TOPTEN/EVC'
FILE? TYPE'YES' IF YOU WANT TO";
A$:IF A$="YES" THEN GOSUB50000
130 CLS0:ZZ=1:NN$="EVICTOR"
140 FORT=1030TO1284:POKET-1,128:
POKET,207:POKET+4,128:POKET+5,17
5:CC=CC+1:IFCC=32 THEN CC=0:POKE
T-1,ASC(MID$(NN$,ZZ,1))AND191:ZZ
=ZZ+1
150 NEXT
160 POKET-1,128:POKET+4,128:PRIN
T@40,"BY PAUL JENSEN";:PRINT@72,
"FEBRUARY 1986";:PRINT@136,"PRES
S ANY KEY.";
170 IFINKEY$=""THEN170
180 POKEHI,0
190 D$=STRING$(28,32)+"*** EVICT
OR ***   BY PAUL JENSEN   BOX 10
35  FOREST, ONTARIO, CANADA N0N
1J0   DEVELOPED IN FEBRUARY OF 1
986 FOR THE RAINBOW.  USE THE RI
GHT JOYSTICK TO PLAY. PRESS ANY
KEY TO START GAME."+STRING$(32,3
2):L=1058:EL$=STRING$(2,220)
200 MG(1)=&HCB:MG(2)=&HCF:MG(3)=
&HC7:EL=79:BL$=STRING$(2,128):LL
=3
210 CLS0
220 CO=0:FORT=66TO482STEP64:PRIN
T@T,STRING$(28,140);:NEXT:PRINT@
480,STRING$(2,239);:POKE1534,239
:POKE1535,239:FORT=79 TO 482STEP
64:PRINT@T,STRING$(2,128);:NEXT:
PRINT@EL,EL$;:BN=50:IF P1=0 THEN
 GOTO720
230 FORT=1TO10
240 D=RND(512)+1023:IF PEEK(D+32
)<>140 OR PEEK(D+1)=175 OR PEEK(
D-1)=175 OR D>=1442 THEN 240 ELS
E D(T)=D:POKE D,175:NEXT
250 GOSUB1050
260 TN=TN+1:IF TN=11 THEN TN=1
270 IF D(TN)=0 THEN 260
280 POKEL,207
290 JS=JOYSTK(0)
300 IF JS<5 THEN DI=-1 ELSE IF J
S>58 THEN DI=1 ELSE DI=0
310 POKEL,MG(DI+2)
320 IF PEEK(L+32+DI)=140 OR PEEK
```

```
(L+32+DI)=&HDC THEN POKEL,128:L=
L+DI
33Ø IF PEEK(L+DI)=175 THEN 66Ø
34Ø POKEL,&HCF
35Ø IF PEEK(L+32)<>&HDC THEN 38Ø
ELSE JS=JOYSTK(1)
36Ø IF JS>58 AND EL<>463 THEN PR
INT@EL,BL$;:POKEL,128:L=L+64:EL=
EL+64:PRINT@EL,EL$;:POKEL,2Ø7
37Ø IF JS<5 AND EL<>79 THEN PRIN
T@EL,BL$;:POKEL,128:EL=EL-64:L=L
-64:PRINT@EL,EL$;:POKEL,2Ø7
38Ø REM
39Ø REM *** MOVE TENANTS ***
4ØØ T=D(TN)
41Ø DI=RND(3)-2
42Ø IF PEEK(T+32+DI)=128 THEN DI
=Ø
43Ø POKET,128:T=T+DI:POKET,175
44Ø REM *** LANDLORD RAY ***
45Ø IF RND(5)<>1 THEN 54Ø
46Ø IF RND(2)=1 THEN DI=-1 ELSE
DI=1
47Ø RY=T
48Ø RY=RY+DI:IF PEEK(RY)=128 THE
N POKERY,188
49Ø IF ZP=Ø THEN IF RY=L THEN ZP
=1 ELSE ZP=Ø
5ØØ IF PEEK(RY+32)=14Ø THEN 48Ø
51Ø IF PEEK(RY)=188 THEN POKERY,
128
52Ø RY=RY-DI:IF RY<>T THEN 51Ø
53Ø IF ZP THEN 74Ø
54Ø REM *** BURN HOLE IN FLOOR
55Ø IF T>=1442 THEN 1Ø6Ø
56Ø IF RND(15)<>1 THEN 61Ø
57Ø POKET+32,128:POKET,128:T=T+3
2:POKET,175:FORDL=1TO1ØØ:NEXTDL:
POKET,128:T=T+32:POKET,175
58Ø SC=SC-25:IF SC<Ø THEN SC=Ø
59Ø GOSUB73Ø
6ØØ POKET-32,14Ø
61Ø REM
62Ø BC=BC+1:IF BC=5 THEN BC=Ø:BN
=BN-1:IF BN<Ø THEN BN=Ø:
63Ø PRINT@1Ø,USING"BONUS ###";BN
;
64Ø D(TN)=T
65Ø GOTO26Ø
66Ø TE=L+DI:FORT=1TO 1Ø:IF D(T)=
TE THEN 67Ø ELSE NEXT:GOTO26Ø
67Ø POKE TE,128:TE=TE+DI:POKETE,
175
68Ø IF PEEK(TE+32)<>128 THEN D(T
)=TE:GOTO 26Ø
69Ø IF PEEK(TE+64)=14Ø THEN POKE
TE+32,128:TE=TE+32:POKETE,175:FO
RDL=1TO1ØØ:NEXTDL:POKETE,128:TE=
TE+32:POKETE,175:D(T)=TE
7ØØ POKETE,128:TE=TE+32:POKETE,1
75:SC=SC+25:GOSUB73Ø:IF PEEK(TE+
32)<>239 THEN 7ØØ ELSE POKETE,12
8:D(T)=Ø:SC=SC+25:GOSUB73Ø:CO=CO
+1:IF CO=1Ø THEN 1Ø4Ø
71Ø GOTO26Ø
72Ø FORT=1 TO LEN(D$)-31:PRINT@4
83,MID$(D$,T,26);:FORDL=1TO15:IF
 INKEY$="" THEN NEXTDL,T:GOTO72Ø
 ELSE PRINT@483,STRING$(26,128);
:GOSUB73Ø:GOTO23Ø
73Ø PRINT@483,USING"SCORE ##,###
  LANDLORDS # ";SC;LL;:RETURN
74Ø FORDL=1TO5Ø:POKEL,128:POKEL,
2Ø7:NEXT
75Ø FORT=1TO1Ø:PRINT@483,"  ****
* YOU BURN!! *****  ";:FORDL=1TO
1ØØ:NEXTDL::PRINT@483,STRING$(26
,128);:FORDL=1TO1ØØ:NEXTDL:NEXT
76Ø ZP=Ø
77Ø LL=LL-1:IF LL>-1 THEN GOSUB7
3Ø:GOTO26Ø
78Ø PRINT@235,"GAME OVER";
79Ø PLAY"V15T3L4CDCL1D#
8ØØ FORDL=1TO1ØØØ:NEXTDL
81Ø POKELO,Ø
82Ø FORDL=1TO5ØØ:NEXTDL
83Ø CLS:PRINT"YOUR SCORE WAS"SC
84Ø OPEN"I",#1,"TOPTEN.EVC"
85Ø FORT=1TO1Ø:INPUT #1,I$(T),S(
T):NEXT:CLOSE#1
86Ø FORT=1TO1Ø:IF S(T)>SC THEN N
EXT:GOTO96Ø
87Ø PRINT"YOUR SCORE PLACES #";T
88Ø A$="***"
89Ø PRINT"ENTER INITIALS:"
9ØØ FORG=1TO3
91Ø PRINT@34Ø,A$;
92Ø Z$=INKEY$:IFZ$=""THEN92Ø
93Ø MID$(A$,G,1)=Z$:NEXTG
94Ø PRINT@34Ø,A$;
95Ø FORF=1ØTO T STEP-1:I$(F)=I$(
F-1):S(F)=S(F-1):NEXT:S(T)=SC:I$
(T)=A$
96Ø PRINT@48Ø,"PRESS <ENTER> TO
SEE HI SCORES";
97Ø IFINKEY$<>CHR$(13)THEN97Ø
98Ø CLS:PRINT"  *** BEST SCORES
TO DATE ***"
99Ø PRINT
1ØØØ FORT=1TO1Ø:PRINTUSING"## %
% ###,###";T;I$(T);S(T):NEXT:PRI
NT:PRINT"PRESS <ENTER>"
1Ø1Ø IFINKEY$<>CHR$(13)THEN1Ø1Ø
1Ø2Ø OPEN"O",#1,"TOPTEN.EVC"
1Ø3Ø FORT=1TO1Ø:WRITE#1,I$(T),S(
T):NEXT:CLOSE#1:RUN19Ø
1Ø4Ø FORF=BN TO ØSTEP-1:SC=SC+1Ø
:PRINT@1Ø,USING"BONUS ###";F;:GO
SUB73Ø:PLAY"T5ØCEG":NEXTF:P1=1:L
=1Ø58:EL=79:GOTO21Ø
1Ø5Ø PLAY"O2T4L8.FL16FL8.DL16DL4
E-L8EL4.FP2P4L4FL8.B-L16B-L8.A-L
16GL8FL4.GP2P4L4FL4.B-L8B-L4A-A-
GL8GL8G-L2G-P8L8B-L8.A-L16GL8.FL
16DL8E-L16EL8FL4D-L4.O-B-O+P4":R
ETURN
1Ø6Ø PLAY"V31":FORS=1TO1Ø:POKED(
S),128:PLAY"T1ØCE-GV-V-V-":NEXTS
:GOTO78Ø
5ØØØØ REM ***** INITIATE FILE **
***
5ØØ1Ø PRINT"INITIATING TOPTEN/EV
C...":OPEN"O",#1,"TOPTEN.EVC":FO
RT=1TO1Ø:WRITE#1,"***",Ø:NEXT:CL
OSE#1:RETURN
```

Submitting Material To Rainbow

This article is about how you should send in your masterpiece - the program you have been working on for the past few months!

Well it's all very simple.

You can save it on tape or disk.

If you are saving it on disk, save it twice under a different name, ie

```
SAVE"filenam1"
SAVE"filenam2"
```

Now read on after the following paragraph.

If you are saving it to tape:

On a 30 minute tape (I prefer C-30's because they don't break too easily) you save the program normally two times, ie

```
CSAVE"filename"
CSAVE"filename"
```

and then you should save it in ASCII mode, ie

```
CSAVE"filename",A
```

This is in case the first two don't make it, we've got a third copy. Mind you, some programs saved three time usually load in on the third copy.

Ok, so there's your program. Next we'd also like to read about your great work, ie what does it do, what it requires, how it works, etc. You know all that sort of info.

To do that, you can use a word processor of some sort to type on all this information.

The type of word processor we prefer (because we have it) is:

1. Telewriter-64 (or Telepatched-64)
2. Pen-Pal
3. Scripsit
4. VIP

Now if, by some chance, that you don't have these commonly used word processors, we suggest the following:

Use the bottom program to write your text with. Writing instructions on paper is ok, but it slows things down a little and delays the time it gets out of the "in" box to the magazine.

The aim of the program is so that the text is saved in some sort of data file (or text file).

Your help will help me make my day and somehow help you.

```
0 GOTO10
1 '** TEXT CREATION FILE *******
3 SAVE"PROG":END
10 CLEAR5000
20 M=1
30 '*****************************
40 REM M=1 IF YOU'RE SAVING YOUR
  DATA TO DISK
50 '*****************************
60 REM M=-1 IF YOU'RE SAVING
  YOUR DATA TO TAPE
70 '*****************************
80 OPEN"O",#M,"TEXT"
90 CLS:PRINT"ENTER YOUR TEXT. TY
PE AN '@'    WHEN FINISHED.":PRI
NT
100 LINEINPUT"> ";A$
110 IF A$="@"THEN140
120 PRINT#M,A$
130 GOTO100
140 CLOSE#M
```

Display graphics files without all the hassle

The Great Pi

As a child I always wanted to draw pictures, but my best efforts looked worse than a 2-year old's finger painting. In my mind I could see the pictures I wanted to draw, but could never get my hands to transfer them to paper.

Then I got a color computer! What a great tool for a would-be artist. It was an answer to my problem. Of course I have to edit each picture over and over, but at least I get my mental pictures on the screen for others to see and enjoy.

I try to collect as many pictures as I can and have 20 disks full of file pictures. I enjoy the artistic works of others and use them as an inspiration for my own creations. I had a problem when I viewed them, though. It was necessary to load each picture into memory to see it. To help me do this, I wrote "Picture Show".

PICTURE SHOW is easy to use. It is menu driven and does practically everything for you. The program starts with a BASIC loader. This loader has the pokes to set the colors. The title page comes up and if the color set is wrong, just press reset. If the colors are right, press ENTER. The pokes restart the program in memory and allow you to set the colors.

Once the colors are correct, you have the option to read the instructions. You may choose to view the pictures individually or automatically and which drive to read. If you choose to see the pictures individually, it presents a menu of the files on disk. Enter the number of the file you want to see and press ENTER. The file is then loaded into view. If the file is a CoCoMax file, use the up and down-arrow keys to see both pictures. Press ENTER to return to the menu.

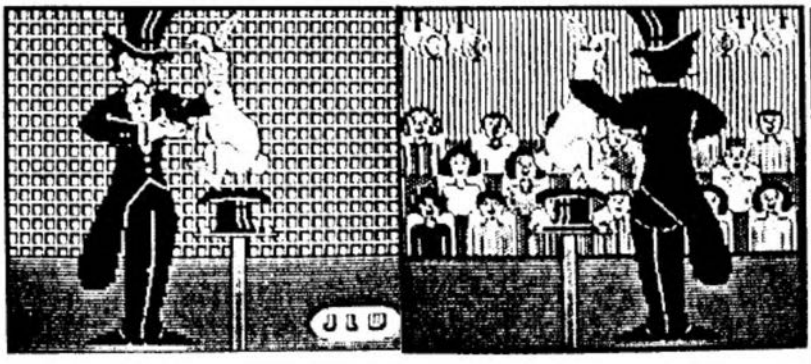
If you choose to see the pictures automatically, Picture Show loads all the files on disk and, if they are CoCoMax files, scrolls the page by itslf. If you choose automatic, make sure the only BIN files are pictures, because the program loads any BIN file on the disk.

Editor's Note: To demonstrate Picture Show's operation, four picture files will be included on this month's Rainbow on Tape/Disk immediately following the Picture Show program listing. When transferring these files to disk, they must be given extensions of BIN, MAX or PIC, followed by the ML Addresses listed below:-

```
SAVEM"MAGICIAN/BIN",&H0E00,&H3DFF,
&HA027
SAVEM"WIZARD/BIN",&H0E00,&H25FF,
&HA027
SAVEM"LATECOCO/BIN",&H0E00,&H25FF,
&HA027
SAVEM"MERLIN/BIN",&H0E00,&H25FF,
&H0000
```

Also be sure to add the extension of "MSP" to the "SHOW" program when transferring to disk.

To generate the underscore(_) in the following listing, use the SHIFT and up-arrow keys. The backslash(\) is generated by pressing the SHIFT and CLEAR keys.



The above is a **CoCo Max** *file. Use the up- and down-arrow keys to scroll the pages.*

RAINBOW ON TAPE filename: MAGICIAN

RAINBOW ON TAPE filename: MERLIN

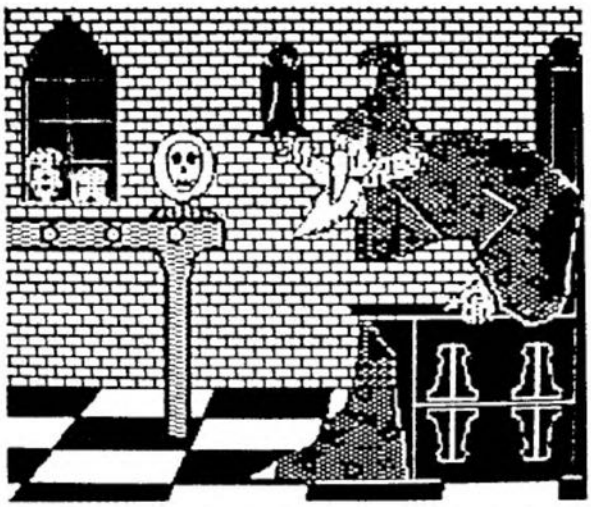
icture Show

By Jeff White

200	26	960	12
360	216	1070	90
510	227	1170	183
660	74	1280	188
810	8	END	37

Listing 1: LOADER

```
10 'PICTURE SHOW
20 'BY JEFF WHITE
30 '(C) 1986
40 '1304 FOUR SEASONS BLVD.
50 'TAMPA, FLA. 33613
60 '(813) 971-4451
70 'LOADER PROGRAM WITH AUTO
    RESTART AND PICTURE DATA FOR
    PICTURE SHOW
80 CLS3:PRINT@235,"one moment";
90 POKE1262,32
100 CLEAR3000,&H7F42:GOTO1400
110 'PICTURE DATA
120 S=6:E=&H1D:IF PEEK(&HC000)=&
H44 THEN D=1:S=S+8:E=E+8
130 POKE&H7FFC,S:POKE&H7FFD,0:PO
KE&H7FFE,E:POKE&H7FFF,&HFF
140 FORI=&H7F42 TO&H7FF7:READ H$
:POKE I,VAL("&H"+H$):NEXT
150 DATA EC,8D,0,B8,83,0,1F,ED,8
D,0,AD,A3,8D,0,AB,43,50,5C
160 DATA ED,8D,0,8C,1A,50,7F,FF,
DF,9E,33,30,6,10,8E,80,0,86
170 DATA 8,A7,8C,3A,86,6,A7,8C,3
4,A6,80,80,30,48,48,48,59,6A
180 DATA 8C,29,27,E,6A,8C,25,26,
F4,E7,A0,C6,8,E7,8C,1C,20,EB
190 DATA 86,6,A7,8C,14,A6,80,26,
A,A6,4,81,22,26,C,30,5,A6
200 DATA 80,80,30,48,48,20,D9,0,
0,8E,80,0,10,AE,8C,50,A6
210 DATA 80,A7,8C,47,6F,8C,45,A6
,80,A1,8C,3F,26,F,E6,80,A6,80,A7
220 DATA A4,8D,15,8D,22,5A,26,F7
,20,4,A7,A4,8D,A,8D,17,27,E2,7F
230 DATA FF,DE,1C,AF,39,10,AC,8C
,1E,24,4,31,A8,20,39,31,A9,E8
240 DATA 21,39,6D,8C,F,26,B,10,A
C,8C,E,26,3,6C,8C,4,1A,4,39
250 READZ:EXEC&H7F42
260 GOTO1340
270 DATA1
280 "m?@40?@13`000?@73`3d10D4004
100@710D50@4410Cd4PD0000>3@\73Pd
290 ";1`h=2`L>3@\73Pd;1`h=2`L>3@
\73Pd;1`h=2`L>3@\73Pd;1`h=2`L>3@
300 "000?@C3o@80?@io`000?ooo?cal
O01m0AE00jKR85VH`ISIV4001PI645@m
310 "0EAAdI6ATM7@O@4Ad57A`000>kM
^gO^gK]gk]fkMnkM^gO^gK]gk]fkMnkM
320 "^gO^gK]gk]fkMnkM^gO^gK]gk]f
kMnkM0000ooo3gMg3gmoOol?MgL?Ogmo
330 "ooo@80?@io`0001DEm0AED0L7@5
AE0>jk3^0OPN6PiZN06njkk[\>hOSin?
340 "WhN7inm0Cim0KW0000k]fkMnkM^
WC]fK]cjm^iLNcL^WO^gK]gk]fkMnkM^
350 "gO^d:A<g<b<K>c<S5k0`;]gk]d0
003oo`aeM@aMKGCo37Ee35e]M?oom0P0
360 "m2?oo?glo?@=oocmo?co0000m0I
E0OkoH0EEEHFhkQ^8ioPHKX1^^^bik;W
370 "XX>JOWijNOWinnO@4n?WPi^3P00
00k]fkMnh=@8nHm0@H6AVHSd001nkM^g
380 "O^gK]gk]fkMnh0N<g=cLcMK@f9L
000^gO^g@000?oo6?Ok?Okg6?lHmo\mo
390 "_LHoood203d1Ood1Okooood1?kd
4_10of1Pm0EWHg1_If1PIfMQn@41o`00
400 "041001PH@40InGT0D5EE0>jk[SQ
^>Vhkk[_^^njkkR00QhLWIbL71oT10@7
410 "Y^@0>0^00003^gK]gk\4AR4a4QP
HF1TA<2101@nkM^gO^gK]gk]fkMnh019
420 "VKVo?d19_a002kMnkM0000oooSG
NognegSon=MkoOkGN?ooo@80?@4o`=?^
430 ";d^9:2PYbO71`od4_1LQF0`4IWI
fITaLN>31acoooomo?co00005AD5IFDE
```



RAINBOW ON TAPE filename: WIZARD

RAINBOW ON TAPE filename: LATECOCO

```
440 "5>7on0EEEE@1k[\0QhNPIR1V8fP
k[S\^^VjJS^7QioWinO@5oWinOW10003
450 "^gK]gk]fkMniM>gM^GC]gk]fkMn
kM^gO^gK]gk]fkMnh5ha<2HK<c<S4ch`
460 "85^gO^g@000?ooookmoOgnooooo
_groOkooood203d1?10LN9RH61VH6AXL
470 "`00m1;o0OUQH6ITm0QU16AVH?T1
0O10003d1UD0oolHOEED0;_XP1SSh05V
480 "HPJPk[{X^H4@LG5alO711LClO7a
`m0E1T0000>kM^gO^gK]gk]fkMnkM^gO
490 "^gK]gk]fkMnkM^gO^gK]gk]fkMn
kM^gO^gK]gk]fkMnkM0000oom`moO`mo
500 "M`og3gmo3emWOooo@80?ooo`m0o
\;0`?S1`<W2o000m1;oh81`H<;7Shod1
510 "In?clMR<0n0h?10003d1eF5PA1E
E@GQY@D1VIV1VH4IOAT5I@FDU15AAO@@
520 "E@000>kM^gO^gK]gk]fkMnkM^gO
^gK]gk]fkMnkM^gO^gK]gk]fkMnkM^gO
530 "^gK]gk]fkMnkM0000oomMcMGIgM
eMol5OGl?Ogd7ooo@80?ool<@Ok6a<C
540 "6o=S<ah006m1;o0<OVI200Qo@7o
hLWInK61Pko0000m0@:0P120_@42P8?0
550 "P;d10X23`82m0@:0P120_@42P8?
0P;d10X23`82m0@:0P120PX:20P03082
560 "m0@:0P120_@42P8?0P;d10X23`8
2m0@:0P120_@42P8?0000ool4gMgLgMg
570 "Lohae_Mc]MHcooo@80?@4ocPCm0
BF5QH6U^100?@Bo`3o<30bm0bc<?`00?
580 "10003d17X2o`82m0Aj0_120_@4N
P;o0P;d17X2o`82m0Aj0_120_@4NP;o0
590 "P9jNWQh0>3d2P10H7Qj0_120_@4
NP;o0P;d17X2o`82m0Aj0_120_@4NP;o
600 "0000ool@moL`mOHGoa3moCgmoAg
ooo@80?@4oaRC51A`L71PB13P0PKoooo
610 "PcmKFa\Kgm_K6g<30ool0cl=32`
_d2__inOcloo10003d17X2o`82NWYjN0
620 "3100;d17X2o`82m0Aj0_120_@4N
P;o0P;d17X2o_@G0?120_@4NP;o0P;d1
630 "7X2o`82m0Aj0_120WYjN7P0h000
0?ookgMkno_gKom_mo_knoO_oood203d
640 "1?18`lG5`L7=`LWAiP00ooon0><
e=C4ah@45LGM`l?oo3<d<3?@;?aj4hGh
650 "00?10003d17X2o`82NWYj>P8300
00@61P0<00001PL70~\000Ng]iN0;o00
660 "1jNWQ0m0X0h83n0O@;08020?@4N
03h001hN71`0>000?@4@?@=0?ooo_@5o
670 "okoo_@5ookooo@80?@4oaR;``^9b
>Qh>1R8b>7Sooan1n?@4S?R`V8`7033o
680 "o`3c@43d34j>21\hN?od101PH60
0h000m0A`0?00<0P60@0003S11^oOgem
690 "_LW_kn?\K21L000MW2`8=3P1?0?
@40o@40?P7m0D0A4Cd1002m1d00@8000
700 "3ooc1gMg1gMcOo<7MgL7Mg=oooo
0P0oooo3d1?61a>Af=9D5Q?@63oon68?
710 "eQH61oHf1VHO`00ool0o\70c<k>
c<7oc<K6`179c<3b0P?om0`0m0D10000
720 "0H51?0<0002lgmoN^Goono\gah0
0O_ooonOVIXE30=oolo?ooocloco`m0\
730 "?00<300@B004210P@8420m0D0oo
mom0OoOo@8oo@80?@4oan?Ch1?7ho?cl
740 "n?3aoooolOcePL3PLS2C1HS`00o
omh6hS8m0C>SPhN?ShNWXk6h7<00?100
750 "008210@8D52Q8Cd1PP000P820Wn
m0@0^7Sh<37`1?[j1_9P00;JT0a<\?3`
760 "1330m0O``o@9?c`0<33`1?10003
Ph?@4071`0000<3Q6PO@=0?oo7Vel7Fe
770 "]7_lNKFdMKFdNoood203d1?o0Wk
2iW8k7Tj6aWh30oooh0XmFUPH7Q\K6aY
780 "100?oo0?o1`<c>c<W3ol?1b<c?c
\c1o`00o`000210@83d103d108?0P8J6
790 "SXjO__d1@0j0]o2P_@4NP8o0P9j
N_@:010@61Pho??c3l?looool`023`82
800 "2QXJ6P8O0P8j>SXJ0Q120PX:0P:
2HaX2m0X0oolgem[Mf]LgocOGf]gJecO
810 "ooo@80?@4ob0?TC0`<S:b\94>02
3oog10laD50L4=0DTA1P00ool0S1o?C`
820 "l?m0Ho3dm?cln?3aoo00000PXJ0
W120_@4NP;o0P;d17X2o`826QXjNP;o0
830 "P;d17X2o`82m0Aj0P?d3@0`3o3?
?<3oo`02o`82m0Aj0_120_@4NP;o0P;d
840 "17X2o`82m0Aj0S120000oomoOoo
oogmoogmooooo0goooo@80?oooPR3iK
850 "Fa\O6aXH67010oooo68_2hW8j7P
h68S8hN?@Ho`0007YjNP;o0P;d17X0m`
860 "l?GSP2NP;o0P;d17X2o`82m0Aj0
_120_@4NP;o0P1PP83d1P0?7`10`@92F
870 "]Xj>P;o0P;d17X2o`8205mlMP7n
0P;d17X2o`82m0Aj0_120000oon>gMkO
880 "gmf>ohkMg]oOgHkooo@80?oooGl
0la@40<0?3gmlOGcloooo03m6APH6MWI
890 "fAUa0@?ooo?Wbi>c\k>OWlO3loo
oQj>c^in3`o`0007YjNP;o0P;d17X2hO
900 "koOam_L`7e01aOGem?4=h>3Vm^M
WP2o`82NWUkN`?30`41m0L03kno_3fm7
910 "CdlO7ajNSL00G]kMgL7m`L3NhGi
1oTNIam_Mg]h0_120_@4NP;o0P000?oo
920 "?mooOkoO?ologomo_mlooood203
d1?10oF=QHO@5ILD43?ooo`3O<3TL3TL
930 "C8C4OO<3oo`7h70a4h30?`o5h70
k6i_IT2?010o10001jNWX2o`82NWU1N`
940 "Of3Xg=bn_Zj^TOoco?moWn?]kUN
go0oomo03mak]?P1O@4ogR77o@4Ooooo
950 "co3m0CoOglo?mlNjOOgkno?m0S_
Kho_oooooWjm0P;d17X2o`80003ool0K
960 "flOKflOo`mgM`mOKgOoom0P0m0C
o7YbMW828V;WKWQ`00?oooclOWalO?an
970 "OWilO7cooo`1n61Sd1=WHgmQHm0
EI67h00?10001jNWP3ka`imL^fM6c[d[
980 ":ZFUZjPWl2h_;bhW;jn_9Bl_Xbl
_;k0_;jnWclm0CnWWod5?17nognm0[og
990 "ln0P_@4NP;o0P;d17X2o`80003o
oo@60@Ko17em3Geml?oom0P0m0Co7hod
1000 "1<o0Q`TC3T3Pml;o031<3?@4k0
c130cd1N`<?`00o`0007Xjf^9O`X;d17
X
1010 "2o`82m0Aj0_120_@4NP;o0P;d1
7X2o`82m0Aj0_120WYjN[[2ko;ho_@@o
g
1020 "nogo@9oocj0_@4NP;o0P;d17X2
o`80003oom3Gem3Gec3oOkno_kVlOooo
m
1030 "0P0oooo`9bVQXjN=VKnQPHC<7S
d4_1l<BH<f9Ra\O@5\k6IV<`68C0lo`0
0
1040 "07YjNP;o0P;d17X2o`82m0Aj0_
120_@4NP;o0P;d17X2o`82m0Aj0_120_
@
1050 "4NP;o0P9j>\[bl__jm0Glm0Kno
?cjnO?goooo_cno?`3d17X2o`82m0Aj0
_
1060 "120000oomooooo1nMom0[om0P0
ooooh@PoF5PH65mKFAS`00?d4_17hAP<
A
1070 "^Kclo@5no?fiT`Hh047o`0007Q
hN031003d17P0o000m0Ah0?`00?@4N03
1
1080 "003d17P0o000m0Ah0?`00?@4N0
31003d17P0o000m0Ah0?`00?@4N0341?
3
1090 "d1?S`c800m0Ah0?`00?@4N03hm
0@0m1?om0P0ooooooQR3aLG1`Hd=PLUg0
0
1100 "3d4_n0_h`<IRHf<c<k>CTm?3`n
OWkd1?10001mO7emO7ioNVeiM71h@V92
H
1110 "TLR8P=7@PP97@\93BAd20TM2PT
=OfioKWm^OfioKWm^0000>gM^GC]gKUd
k
1120 "MfiM>gM^GC]gKUdkMfiM>gM^GC
]gKUdkMfiM<6I^ARIfIUHgMVA@<7L000
3
1130 "ooo@?EOoom0P0oooo7h31Q8201
h>0TXCh007d4_10OAPHCDd]9RH6Rh^;d
M
1140 "7AH61V1PKo00004QXG4QXA4SXG
4QZ9TR9<8TM31T=B@b:30R?30f93@[<<
7
1150 ")`lJ?S\o>klk_o^onkokP000;]
gk]fkMnkM^gO^gK]gk]fkMnkH\gCTd;1
f
1160 "k]bjM^cA]W3PgCQbIe<38f?VK7
Q`0@=g0000oood3eGooo@80?@;oo@4Oo
@
1170 "Do`3_PX8D52@X:0TA4A>SXjO7a
11?3o1000293I0@4A8j518Jk?o^Ofho3
P
1180 "71nnci1^O>gbioK3h<`>k?kSn>
PBR1k_1^?1k`k_o^0000\4BO[:`<S1b\
K
1190 "8boP03^gK]gj0;?e]K6ao@4a]n
001\gK]d0>5QH63mH61PHL002M`000?o
o
1200 "m0mEoood203d8olOWalOOgod4?
10000WKQ::4e<oTeKB`3g;h>kok_n^On
c
1210 "1k_o'onkok_o^onkok_o^onkok
_o^o0;ok_o^0000^a26c\K6Q_@4I\<00
>
1220 "gL^7M^0?<E1@713@594NL00;]g
kP5``1C4`<3>`<S@h002M`000?oom0mE
o
1230 "ood203d>O10001OTm3;jmmk3BF
eQ<B>ADD6k_k^onho003Ronkok_o^onk
o
1240 "k_o^onkok_323nkok_o^0000^d
5HFAUIF5TI>ERI4B721@]g3X31QH6018
>
1250 "2WlPE31bkMnkM2<?4a<30cl79d
>4007L0003ooo@?EOoom0P0m3Wo0000X
W
1260 "=:FB40NPR97H^93HTMRhT=i?3^
3`hok_o^onkok_o^onkok_cP`bkok_o^
o
1270 "nkokP000;]gk]fkMnkM^gO^gK]
gk]fkMhHPof5PH7aPH6ESa`H=^gO^g@3
m
1280 "HV9PH61VIV3300Qg0000oood3e
Gooo@80?@io`00018B>1LB6T::2B7a8e
4
1290 "AOLPXGTU2h\NRXX?Mfl;Raj;21
<JTXP88d?SQj>SQhN?AdLP0002kMnkM^
g
1300 "O^gK]gk]fkMnkM^gO^7C\g;Qfk
MnkM^gO^gK]gkPe0o\;0`?W0`<W2o000
M
1310 "`000?@Coo@80?@i1000020P<?1
P\41PH10PX70PX20`420P@43`@510`21
0
1320 "@?0P821`820`P93@\90@8:1`84
00002`L>3@\73Pd;1`h=2`L>3@\73Pd;
1
1330 "`h=2`L>3@\73Pd;1`h=0`L61@<
73P430`852`L0000?@Cn?@400
1340 CLS3
1350 'RESTART DATA
```

```
1360 A=PEEK(116)*256+PEEK(117)-2
0:X=INT(A/256):Y=A-(X*256):POKE1
13,85:POKE114,X:POKE115,Y:FORI=A
 TO A+17:READ B: POKE I,B:NEXTI:
DATA 18,182,255,3,138,1,183,255,
3,189,173,33,189,172,239,126,173
,158
1370 PRINT@228,"LOADING--> pictu
re show";
1380 POKE1270,32
1390 RUN"SHOW.MSP"
1400 PCLEAR8:GOTO110
```

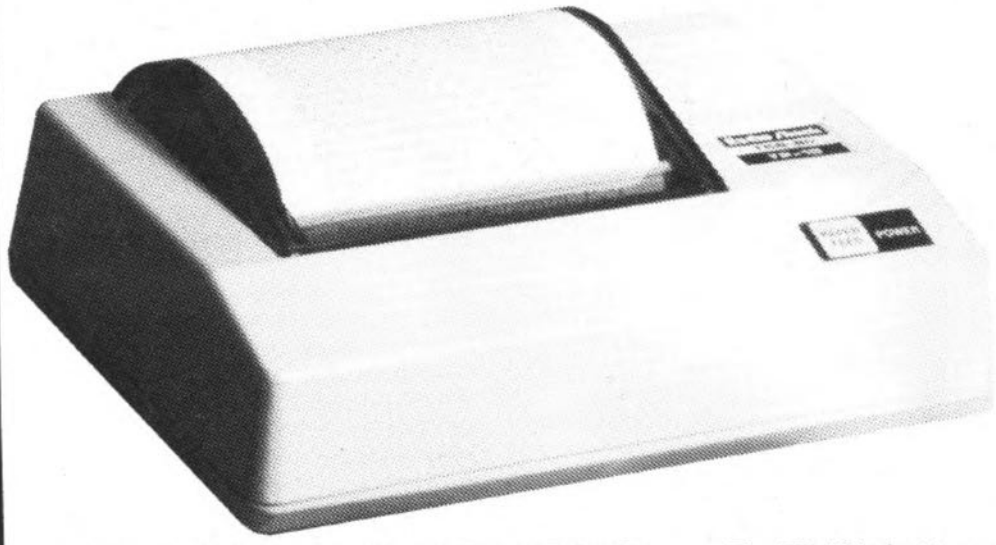
```
210 ......198    1100 .....214
400 .......88    1200 ......17
620 ......210    END ......42
860 ......113
```

Listing 2: SHOW

```
10 PMODE4,1:SCREEN1,1
20 A$=INKEY$:IF A$="" THEN 20
30 FORT=1TO4:PCOPY T TO T+4:NEXT
40 P=RND(-TIMER)
50 P=RND(4)-1
60 POKE178,P
70 A=0:B=0:C=255:D=191
80 FORT=1TO100
90 LINE(A,B)-(C,D),PSET,B
100 A=A+1:B=B+1:C=C-1:D=D-1
110 NEXT T
120 B=3:CLS(B)
130 PRINT"do you need instructio
ns  (y/n)?";
140 POKE1026,32:POKE1030,32:POKE
1035,32:POKE1048,32:POKE1049,32:
POKE1050,40:POKE1052,47:POKE1054
,41:POKE1055,63
150 A$=INKEY$:IF A$="" THEN 150
160 IF A$="Y" THEN 1080
170 CLEAR:DIM C$(11),PIC$(68),EX
T$(68)
180 B=3
190 CLS(B)
200 PRINT"    automatic or indiv
idual?"
210 POKE1024,32:POKE1025,32:POKE
1026,32:POKE1027,32:POKE1037,32:
POKE1040,32:POKE1052,32:POKE1053
,32:POKE1054,32:POKE1055,32:POKE
1051,63
220 AI$=INKEY$:IF AI$="" THEN 22
0
230 IF AI$="A" THEN A=1 ELSE A=2
240 PRINT@256,"          (ENTER
)= 0"
250 POKE1279,95
260 PRINT@224,"";:INPUT"ENTER DR
IVE NUMBER (0,1,2,3)";K
270 IF K<0 OR K>3 THEN 200
280 DRIVE K
290 B=3:CLS(B)
300 GOSUB550
310 PRINT@392,"enter the number"
;
320 PRINT@425,"of the picture";
330 POKE1425,32:POKE1421,32
340 POKE1448,32:POKE1451,32:POKE
1455,32:POKE1463,32
350 PRINT@456,"to be loaded";
360 POKE1482,32:POKE1485,32:POKE
1492,45:POKE1493,62
370 POKE1494,32:POKE1495,32
380 PRINT@488,"type (q) to quit"
;
390 POKE1516,32:POKE1517,60:POKE
1519,62:POKE1520,32:POKE1523,32
400 PRINT@470,"";:LINE INPUT"";F
$
410 FORT=1496TO1503:POKE T,62:NE
XT
420 IF F$="Q" THEN 890
430 F=VAL(F$)
440 IF F<1 OR F>C THEN 370
450 P$=PIC$(F)+"/"+EXT$(F)
460 PMODE4,1:PCLS:SCREEN1,1
470 LOADM P$
480 I=7:PMODE4,1
490 IF (PEEK(&H155) AND 8)=0 THE
N I=I+1:IF I=>19 THEN I=19:GOTO5
10
500 IF (PEEK(&H156) AND 8)=0 THE
N I=I-1:IF I<=7 THEN I=7:GOTO510
510 POKE &HBA,I+I:SCREEN1,1
520 IF INKEY$<>CHR$(13) THEN 490
530 GOSUB720
540 GOTO310
550 'GET FILE NAMES
560 FOR X = 3 TO 11
570 DSKI$ K,17,X,A$,B$
580 IF (LEFT$(A$,1)=CHR$(&HFF))
THEN 600
590 C$(X)=A$+LEFT$(B$,127):NEXT
X
600 POKE&HFF40,0:X=X+1:C=1
610 FOR Y = 3 TO X:FOR Z=0 TO 7
620 IF MID$(C$(Y),Z*32+9,3)="BIN
" OR MID$(C$(Y),Z*32+9,3)="MAX"
OR MID$(C$(Y),Z*32+9,3)="PIC" TH
EN 630 ELSE 680
630 PIC$(C)=MID$(C$(Y),Z*32+1,8)
640 EXT$(C)=MID$(C$(Y),Z*32+9,3)
650 O$=LEFT$(PIC$(C),1)
660 IF (O$=CHR$(0) OR O$=CHR$(&H
FF)) THEN 680
670 C=C+1
680 NEXT Z:NEXT Y
690 IF A=1 THEN GOSUB790
700 C=C-1
710 IF C=0 THEN 1260
720 MID=INT(C/2)+1
730 CLS(B):TAB=1
740 FOR D = 1 TO C
750 PRINT@TAB,USING"##";D;:PRINT
".--> ";PIC$(D);
760 TAB=TAB+32:IF D=MID THEN TAB
=16
770 NEXT D
780 RETURN
790 'AUTOMATIC DISPLAY
800 FOR D=1 TO C-1
810 POKE15870,111
820 IF C=1 THEN 1260
830 PMODE4,1:SCREEN1,1
840 P$=PIC$(D)+"/"+EXT$(D):LOADM
 P$
850 FORT=1TO800:NEXTT
860 IF PEEK(15870)<>111 THEN GOS
UB 990 ELSE PMODE4,1:SCREEN1,1
870 NEXTD
880 C=C-1
890 GOSUB720
900 PRINT@384,"DO YOU WISH TO DO
 ANOTHER DISK? "
910 PRINT@428,"(yes/NO)"
920 FORT=1TO300:NEXTT
930 PRINT@428,"(YES/no)"
940 FORT=1TO300:NEXTT
950 A$=INKEY$:IF A$="" THEN900
960 IF A$="Y" THEN 170
970 IF A$="N" THEN 980 ELSE 950
980 POKE113,0:EXEC40999
990 S=6:IF PEEK(&HC000)=&H44 THE
N S=S+8:U=S+24
1000 PMODE4,1:SCREEN1,1
1010 FORT=1TO500:NEXTT
1020 FOR V=S TO U:GOSUB1070:NEXT
V
1030 FORT=1TO500:NEXTT
1040 FORV=U TO S STEP-1:GOSUB107
0:NEXTV
1050 FORT=1TO500:NEXTT
1060 PMODE4,1:SCREEN1,1:RETURN
1070 POKE&HBA,V:FORT=1TO100:NEXT
T:SCREEN1,1:RETURN
1080 'INSTRUCTIONS
1090 CLS
1100 PRINT"          instructions
"
1110 PRINT:PRINT"GO GET YOU YOUR
 POPCORN AND YOURDIET COKE AND S
IT BACK AND ENJOYTHE SHOW.
1120 PRINT:PRINT"PICTURE SHOW IS
 A VERY EASY TO  USE PROGRAM. IT
 IS MENU DRIVEN  AND GIVES YOU 2
 WAYS TO SEE YOUR";
1130 PRINT"PICTURES."
1140 PRINT@448,"   press spaceba
r to continue   ";
1150 POKE1480,32:POKE1489,32:POK
E1492,32
1160 A$=INKEY$:IF A$="" THEN 116
0
1170 CLS
1180 PRINT"1. automatic---> LETS
 YOU SIT   BACK AND YOUR COMPUTE
R DOES THE REST. IT WILL EVEN SC
ROLL 2 PAGEMAX FILES BUT THEY MU
ST HAVE AN EXTENTION OF <BIN>, <
MAX>, OR   <PIC>.
1190 PRINT"2. individual--> LETS
 YOU PICK  WHICH PICTURE YOU WAN
T TO LOOK  AT. IT WILL ALSO SCRO
LL 2 PAGE  MAX FILES BY USING TH
E UP AND   DOWN ARROW KEYS, THEN
 PRESS THE ";
1200 PRINT"<ENTER> KEY TO GET BA
CK TO THE  MENU. "
1210 PRINT
1220 PRINT@480,"   press spaceba
r to continue ";
1230 POKE1512,32:POKE1521,32:POK
E1524,32
1240 A$=INKEY$:IF A$="" THEN 124
0
1250 GOTO170
1260 CLS(B):DIR
1270 PRINT:PRINT"THERE ARE NO <B
IN>, <MAX>, OR   <PIC> FILES ON
THIS DISK"
1280 FORT=1TO4000:NEXT
1290 GOTO170
```

Tandy ELECTRONICS

COLOR COMPUTER THERMAL PRINTER



50% OFF!

Reg **189.95**

99.95

Here's a great way to expand your Tandy Color Computer system and save lots of money! The **TP-10** will give you hard copies of notes, letters, drawings, recipes or whatever data you store in your computer. It's very compact, whisper-quiet, and prints both alphanumerics and graphics on 10.4cm thermal paper.

The TP-10 is fast too, printing 32 characters per line at 30 characters per second, and has an elongation mode for expanded print and repeat function for easier graphics programming. Color Computer-compatible serial interface only. With the money you save, invest in more Color Computer accessories! 26-1261

Portable Thermal Ribbon Printer

Tandy 200 Not Included

Save $200

Reg **499.00**

$299

If you require a portable printer at an unbeatable price — look no further! The **TRP-100** is designed for mobility, working on AC power (with included adapter) or batteries so it's easy to move around. The super-quiet TRP-100 prints up to 50 characters per second and bit-image graphics on plain or thermal paper. Has both parallel and Color Computer-compatible serial interfaces. 26-1275

Wordprocessing Paper*

Size (mm)	Design	Quantity	Cat.No.	Reg	Now
241 x 279	Blank White	1000 sheets	26-9316	39.95	
241 x 279	Quartz	250 sheets	26-9318	19.95	12.95
241 x 279	Ivory	250 Sheets	26-9319	19.95	12.95
241 x 279	Mist	250 Sheets	26-9320	19.95	12.95

*All Wordprocessing Paper has "INVISIBLE" vertical edge perforations.

Fanfold Printer Paper

Size (mm)	Design	Quantity	Cat.No.	Price
241 x 279	Blue H/S	1000 sheets	26-9317	39.95
381 x 279	Blue H/S	1000 sheets	26-9315	39.95

16K STD BASIC COLOR COMPUTER 2

Save $60

Reg **239.95**

179.95

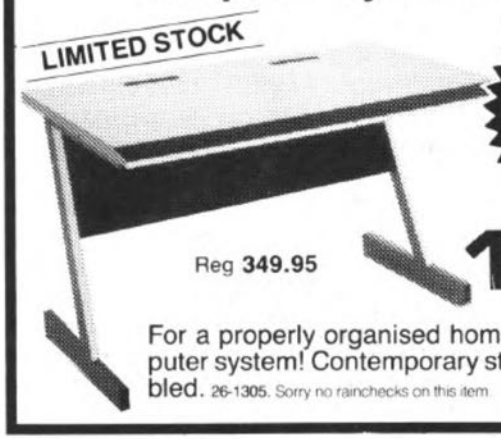
TV Not Included

A low-cost introduction to the exciting world of computers! Plug it into your TV and begin by creating eight color graphics with sound or music, next write your own programs. Or snap in Program Pak™ cartridges for hours of family fun playing games or educational software. It's ideal for children improving their schooling skills, students storing homework data or adults listing recipes or home budget details. 26-3134 No Rainchecks On This Item

16K Extended BASIC Color Computer 2. Similar to above but with extended color BASIC. Expand memory for advanced uses. 26-3136 Reg 349.95 . . . . . **Now 229.95**

64K Extended BASIC Color Computer 2. Designed for business or advanced home use! Access data bases, work out budgets, do word processing! Add an optional disk drive, printer and modem for a complete personal computer system. 26-3127 Reg 449.95 . . . . . **Now 299.95**

Computer System Desk



LIMITED STOCK

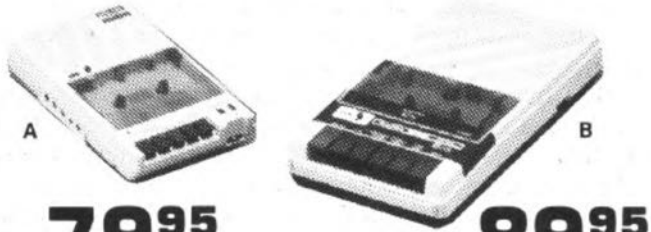
60% OFF!

Reg **349.95**

149.95

For a properly organised home or office computer system! Contemporary styling, unassembled. 26-1305. Sorry no rainchecks on this item

Computer Cassette Recorders



79.95

89.95

A. CCR82. Expand your computer, store data. 26-1209

B. CCR81. Load and record programs or games! Cue/review, pause, auto stop functions. Tape counter. 26-1208

Tandy ELECTRONICS

A DIVISION OF TANDY AUSTRALIA LIMITED INC. IN N.S.W.

Nearly 350 Stores Australia-Wide

WE SERVICE WHAT WE SELL!

Available from 350 Stores Australia-wide including Tandy Computer Centres

Independent Tandy Dealers may not be participating in this ad or have every item advertised. Prices may also vary at individual Dealer Stores

**Faster than keyboard entry —
more user-friendly than arrow keys
— able to leap whole screens in a single bound.
Yes, it's . . .**

CoCo Mouse!

by Steve Bjork

In the past 15 years I have seen the user interface for computers improve greatly. First I used punch cards to enter my programs into a computer. Next came the hard copy terminal, but it was like using a typewriter (very slow and noisy). Now finally, the home computer with its display and keyboard makes for faster, quieter and easier user interface.

True, we have come a long way since the punch card, but the keyboard is not always the best way to talk to a computer. An example is the game *Mega-Bug*; you can use the keyboard's arrow keys to control the game, but the joystick makes it easier to play the game.

The same is true for most programs written in BASIC. Moving a pointer on the screen to select an option from the menu is more user friendly than looking at the screen, then pressing keys to tell the computer what to do.

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

This is where *The Mouse* comes in. It uses a Color Computer mouse (or joystick) as a point-and-click user interface. By moving the mouse, a pointer (cursor) is moved around on the screen. When the cursor is over the option wanted, press the button on the mouse to select it. No more looking away from the screen to hunt for keys to enter your selection.

About a year ago I placed in public domain my first version of *The Mouse* Version 1.0. After receiving many requests for a few options to be added, Version 2.0 of *The Mouse* is ready for release.

The Mouse is an assembly language program that displays a moving cursor and prints upper- and lowercase text on the Hi-Res graphics screen. This Hi-Res text driver has a format of 32 columns by 24 lines with a scroll-protect window option. Bell and click sounds also have been added to round out the package.

Software Overview

The Mouse communicates with a BASIC program via a USR function and the PRINT command. The 60-hertz interrupt is used to draw the cursor on the screen and read the joystick's position and button status. *The Mouse* can poll the right or left joystick ports, *CoCo Max* Hi-Res input module, the Radio Shack *X-Pad* or the Hi-Res Joystick Convert for the cursor position and button status. The PRINT command is redirected from the standard green text screen to a Hi-Res screen text driver whenever the Hi-Res screen is displayed. The USR function has 14 commands that can be passed to it. They are:

USR(0) — This command turns off the cursor (pointer) on the screen. Make sure the cursor is off before using any BASIC graphics commands.

USR(1) — This command turns the cursor on. The cursor should only be on when a selection is to be made from the screen.

USR(2) — This function returns the X position of the cursor (and the joystick).

USR(3) — This function returns the Y position of the cursor (and joystick).

(Note: Because the Color Computer joystick port uses a six-bit DAC system, they only return a value of zero to 63. To get full-screen movement of the cursor, the zero to 63 from DAC is multiplied by two for zero to 126 across, or multiplied by three for zero to 189 up and down. The *CoCo Max* hardware, *X-Pad* and Hi-Res Joystick Interface options return a full 128 by 192 reading.)

USR(4) — This function is used to find out if the button has been pressed. A zero is returned if the button has not been pressed since the last time the command was used. A one is returned if it was pressed.

USR(5) — This function returns the up/down status of the button. A zero is passed if the button is not pressed or a one if the button is pressed.

USR(6) — This command is used to unlink (turn off) *The Mouse* system. This command *must* be used when ending the BASIC program. When any other command is used, *The Mouse* is automatically linked into the Color Computer system.

USR(7) — This command plays a bell sound.

USR(8) — This command plays a click sound.

USR(9) — This command selects the right joystick as the input device for cursor movement and button status. This is the standard input device for *The Mouse* and is automatically selected when *The Mouse* binary file is loaded into memory.

USR(10) — This command selects the left joystick for cursor position and button status.

USR(11) — This command selects the *CoCo Max* Hi-Res Input Module for cursor position and button status.

USR(12) — This command selects Radio Shack *X-Pad* for cursor position and button status.

USR(13) — This command selects Hi-Res Joystick Interface for cursor position and button status.

Tables 1 and 2 are for quick reference of the USR commands and the screen control code for the Hi-Res screen text drivers.

To make the binary file of *The Mouse*, type in Listing 1 and save it. Now run the program. If an error is encountered in one of data lines the program prints the line number and stops. After all the data lines have been converted, the program asks if the binary file should be saved to tape or disk.

The disk version of *The Mouse* binary file has a load address of zero and needs an offset address whenever it is loaded into memory. To load this file in the end of memory of a 16K system, a load offset address $3400 should be used (LOADM "MOUSE",&H3400).

The cassette version has a starting address of $3400 and ending address of $3FFF. If the file is loaded on a 32/64K system, a load offset of $4000 will put it at the top of memory ($7400 to $7FFF).

Table 1
USR commands

0 Turn off Hi-Res cursor.
1 Turn on and display Hi-Res cursor.
2 Get Joystick X position (0 to 127).
3 Get Joystick Y position (0 to 191).
4 Get button press. If button was pressed, senses the last use of this command and a -1 is returned. Else a 0 is returned.
5 Get button status. The number -1 is returned if the button is pressed down. Else a 0 is returned.
6 Disable *Mouse* software, unlink its hooks.
7 Play Bell sound.
8 Play Click sound.
9 Select right Joystick for input device.
10 Select left Joystick for input device.
11 Select *Coco Max* Hardware for input device.
12 Select Radio Shack *X-Pad* for input device.
13 Select Hi-Res Joystick Interface for input device.

Using the Mouse Software

Now that you have an idea of what the commands are, let's see how to use them. *Lines* (Listing 2) is a BASIC program that uses *The Mouse* to draw lines on the screen by selecting the start and end points. The program also has the option to exit or clear the screen.

Table 2
Screen Control Codes

0 Nil, do nothing.
1 X-position, Y-position — Set cursor position.
2 Select white on black characters.
3 Select black on white characters.
4 X-position, Y-position, X-length, Y-length — Set window position and size.
5 Move text cursor left
6 Move text cursor right.
7 Play bell sound.
8 Backspace.
9 Move cursor to next tab position.
10 Move down one line with scroll (line feed).
11 Move up one line.
12 Clear screen and home cursor (form-feed).
13 Carriage return.

14 to 31 and 128 to 255 are not defined at this time.
32 to 127 are printable characters.

Line 100 clears space for *The Mouse* and Line 120 loads it in. Note: this program is configured for loading in *The Mouse* from disk. If you are using this program on a casette-based system, delete Line 120 and remove the apostrophe (') from Line 140.

Line 150 defines USR function zero with *The Mouse* address. The next line tells BASIC to clear and display a 6K Hi-Res screen.

To make easy use of the scroll-protect windows, lines 170 through 180 define three types, full screen, top line and draw area. Information for window placement and size is pasted to *The Mouse* by printing a control code number four and four bytes of data. The format is: 4, X-position, Y-position, X-length, Y-length. The X-position is any of the first 31 columns (0-30). The Y-position is any of the first 23 lines (0-22). The X-length is from 1 to 32 (X-position) in size. The Y-length is from 1 to 24 (Y-position) in size.

After using any of the USR functions *The Mouse* is linked in the PRINT command and the 60-hertz interrupt and that's what the USR(0) in Line 190 does. Lines up to 260 print the instructions to the program. Line 270 waits for the user to press the button to continue. The GOSUB in Line 500 clears the work area.

After turning on the cursor, the computer waits for the user to press the button to place the starting point of the line. This is done in lines 290 through 350. Next comes the end point selection by pressing the button one more time after moving the cursor on the screen (lines 360 to 400).

The interrupting of the cursor points on the screen is done by the subroutine at lines 410 to 490. If the cursor point selected (by pressing the button) is on the work area, then the X and Y location is returned. If the Clear option is selected, the screen is erased to black. If the Exit option is selected, the program unlinks *The Mouse* from the system and ends the program in lines 520 through 530.

Whenever the screen is changed by drawing a line, clearing the screen or placing a dot, the cursor must be turned off and then turned back on after all changes are made. The reason for this is the cursor makes a copy of the area underneath before drawing on the screen. When it is turned off or moved, the old area is restored. The *Lines* program is good for demonstrating *The Mouse*, but not much else.

Now let's look at real application of the point-and-pick user interface, the *Disk Drive Timer*. This program (List-

ing 3) makes extensive use of *The Mouse* and icons.

After running the programs, you see four disks on the screen. Each disk represents disk Drive 0 to 3. To test the speed of a disk drive, first select the disk icon representing the drive by moving the cursor (arrow) over the disk and pressing the button. The computer tests to see if the disk drive is online with a disk in it. If there is no error, then that disk drive's icon is inverted to show it is the selected drive.

At the bottom of the screen is the command menu with three options. First is the Disk Speed History. This command builds a bar graph of the drive's speed over a long period of time. Press the button to exit this command.

Next is the Adjust Speed command. This command displays in real time the speed of the drive. Again, press the button to exit this command.

The last is the Exit command. This unlinks *The Mouse* and exits the program.

Now that you understand how to use *The Mouse*, let's see what you can write! Send those programs (along with documentation) to THE RAINBOW or upload them to RAINBOW'S CoCo SIG on Delphi.

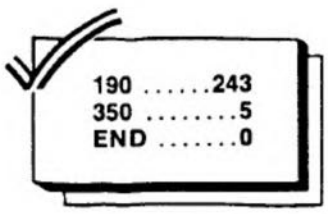
So much for this month, next time we'll look at how *The Mouse* works by taking a peek at its assembly language source code.

240 151	1250 88
1010 181	1310 163
1070 159	1360 2
1130 242	1420 126
1190 3	END 68

Listing 1: MOUSE

```
0 GOTO 2000
10 'THIS PROGRAM MAKES A MOUSE
20 'BINARY FILE
30 'COPYRIGHT 1986 BY SRB SOFTWA
RE
40 'THIS IS A PUBLIC DOMAIN
50 'SOFTWARE BY STEVE BJORK
100 CLEAR 50,&H33FF
110 MOUSE=&H3400
120 CLS:P=MOUSE:LN=1000
130 READ L$,C:S=0
140 PRINT @0,"WORKING ON LINE";L
N
150 FOR X=0 TO 63
160 V=VAL("&H"+MID$(L$,X*2+1,2))
170 POKE P,V:S=S+V:P=P+1
180 NEXT X
190 IF C<>S THEN PRINT "DATA ERR
OR IN LINE";LN:END
200 LN=LN+10:IF P<MOUSE+&HBFF TH
EN 130
210 CLS:PRINT"SAVE TO DISK OR CA
SETTE? (D,C)
220 I$=INKEY$:IF I$="D" THEN 300
 ELSE IF I$<>"C" THEN 220
230 PRINT "PRESS ENTER WHEN READ
Y TO SAVE  TO CASSETTE."
240 IF INKEY$<>CHR$(13) THEN 240
250 CSAVEM"MOUSE",MOUSE,MOUSE+&H
BFF,&HA027
260 PRINT"MOUSE BINARY FILE IS N
OW ON THE TAPE."
270 PRINT"THE LOAD ADDRESS IS $3
400-$3FFF"
280 END
290 'THESE LINES ARE FOR DISK
300 INPUT"WHAT DRIVE SHOULD THE
FILE BE   SAVED TO";D
310 IF D<0 OR D>3 THEN 300
320 SAVEM"MOUSE:"+CHR$(48+D),MOU
SE,MOUSE+&HBBF,0
330 OPEN"R",1,"MOUSE/BIN:"+CHR$(
48+D),1
340 FIELD #1, 1 AS R$
350 LSET R$=CHR$(0):PUT #1,4
360 LSET R$=CHR$(0):PUT #1,5
370 CLOSE
380 PRINT"THE MOUSE BINARY FILE
AS A LOAD ADDRESS OF 0"
390 PRINT"A LOAD OFFSET MUST ALW
AYS BE    USED!"
400 END
1000 DATA "16025A01003FFFFF1FFFF
F0FFFFF07FFFF03FFFF01FFFF00FFFF0
07FFF003FFF001FFF001FFF00FFFF30F
FFFF87FFFF87FFFF87FFF00000040000
06000007000",8307
1010 DATA "007800007C00007E00007
F00007F80007FC0007C0000460000060
0000300000300000000CFFFFFC7FFF
FC3FFFFC1FFFFC0FFFFC07FFFC03FFFC
01FFFC00FFF",6837
1020 DATA "C007FFC007FFC03FFFCC3
FFFFE1FFFFE1FFFFE1FFF00000010000
01800001C00001E00001F00001F80001
FC0001FE0001FF0001F0000118000018
00000C00000",5094
1030 DATA "C00000000F3FFFFF1FFF
FF0FFFFF07FFFF03FFFF01FFFF00FFFF
007FFF003FFF001FFF001FFF00FFFF30
FFFFF87FFFF87FFFF87FF00000004000
00600000700",9629
1040 DATA "0007800007C00007E0000
7F00007F80007FC0007C000046000006
00000300000300000000FCFFFFFC7FF
FFC3FFFFC1FFFFC0FFFFC07FFFC03FFF
C01FFFC00FF",6882
1050 DATA "FC007FFC007FFC03FFFCC
3FFFFE1FFFFE1FFFFE1FF00000001000
001800001C00001E00001F00001F8000
1FC0001FE0001FF0001F000011800001
800000C0000",6279
1060 DATA "0C00000000023E0169015
B015E0161016602A0046904B402B402B
402B402B402B4000000000000000000
0000000000000FFFF000000000000
00000001820",2501
1070 DATA "00000000000000000000
0000000000000000000000000000000
0000000000000000000000000000000
0000000000000000000000000000000
00000000000",0
1080 DATA "00000000000000000000
0000000000000000000000000000000
0000000000000000000000000000000
0000000000000000000000000000000
00000000000",0
1090 DATA "00000000000000000000
0000000000000000000000000000000
000003410308D01DBBC010D351027513
476FC010D338DFF3EED4117021FED523
08D01C0BF01",3064
1100 DATA "0DFC0168ED43308D041EB
F01686F5DBE0123F60120C1352620318
DFF5658AF4710BF0123A680A7A05A26F
9EC8DFF52ED49308D03A4AF8DFF48357
64D2624BDB3",7037
1110 DATA "ED1083000E221B3474338
DFEE4318DFEB458ECA531AB3504ADA43
5704F7EB4F4E65339E65239E6556F553
9E6543934011A50E651CA40E7518D023
581E6512BED",7464
1120 DATA "C54027E934411A50CA80E
751308DFCF3318DFEADEC52A080E0806
F5A81B02406C1742402635A34041F138
1C53401250140C6203D350125054353C
300019BBA1F",6433
1130 DATA "01E6E45757E78DFE5F308
5AF8DFE5A35028403C6603D33CB8610E
68DFE4D26379CBA252934528603E68DF
E3CC12024139CB7240F3402A684A7A0A
4C4A8C830A7",7607
1140 DATA "843502300133415C4A26E
1355230882033434A26CB35C1A78DFE0
6EC84EDA1A4C4A8C830E441E8C831ED8
1A684A7A0A442A8C832A784334330881
E6A8DFDE226",7729
1150 DATA "DA35C134011A50E651C4B
FE7518D02358134011A50E6512AF6C47
FE751318DFDE0AE588610E65A26299CB
A251D3412E6578603A750C12024089CB
72404A6A0A7",7149
1160 DATA "8430015C6A5026ED35123
088204A26D93581A750ECA1ED81E6A0E
78430881E6A5026F1358117FF9BAE41B
F010DEC43FD0168EC47FD01235F39C00
9E75F39338D",7218
1170 DATA "FD6FAE45A640AA5181402
70FAC522712341017FF7C3510A640260
7AF5217FE9C2006E64027026A40308C0
73410347F6ED8018D23ED45E654A6562
7075D26096A",5802
1180 DATA "5620095D2706E7558602A
7563B009D009D013D01100014A65F483
08CF0EC866E8B4F8D6DEC52634F2B428
EFF00A68823E6882034068 4F7A78823A
601E6033406",6121
1190 DATA "C4F7E7038A08A7018D238
6C03D3402A60184F7A7018D165435021
F023506A701E7033506E78820A788231
F2039340186FFA78820862D4A26FDCC0
27E1A50A788",6096
1200 DATA "205A26FDA6842B0712121
25C26F65A358134028EFF00A602C6FFE
70235045C53EA84A70253C403E75439A
65F8DE334408EFF00A68823E688201F0
284F7A78823",6816
1210 DATA "A601E6031F038A08A701C
4F73404E68DFC5E585858EAE0E7038D2
286033D3404A60184F7A7018D1558340
41F30E703A7011F20E78820A78823350
435C2CC4080",6350
1220 DATA "3402CA02E78820C802A68
42B04E0E42002EBE4350244810126E65
45439F6FF6054B6FF613406F6FF62C50
22607EC52EDE45F2010C5042704A653A
761C5082704",6946
```

```
1230 DATA "A652A7E4C401E75435863
4068EFF908D158D13C6C03DA7E48D0C4
4A7618D07438480A7543586A680C60A5
A26FD398D2A34011A5086E68D3B1F89C
4F7CA02F7FF",7670
1240 DATA "208D30C602F7FF204A4A8
11224E83501B6FF2384F7B7FF235F39B
6FF238A08B7FF23B6FF0184F7B7FF01B
6FF0384F7B7FF0339C6B45A26FD398DD
F860A1F895A",8294
1250 DATA "26FDF6FF20C8F0F7FF204
C814C25EE20BFC6088CC6027EAC46273
B8140263734O2B6FF2284E088E035022
62ABDB3E43440338DFB3C6F506C50E04
E820024F8EB",7796
1260 DATA "4EA6504AA14D24C9ED5B3
5409DA52706812C26C09D9F6E9DFB200
075004D004101740186019901A43404F
6FF22C8E0C4E0DA6F350427046E9DFAF
93262347633",6783
1270 DATA "8DFAEEE65D308CD258EC8
5308BA6E4AD84E65CCAE08605DD8835F
6A14D2503A64D4AA75B6F5D39A14E250
3A64E4AA75C6C5D3901100111010E015
60116017201",6740
1280 DATA "7DFEFB015B01920073018
800E1005F810D220848308CDCEC866E8
B8020816022301700A5E65CE14E25066
F5C6C5B8D38318D0165C6083D31ABEC4
BE35B9BBA1F",6426
1290 DATA "01C608A6A0A85EA784308
8205A26F46C5C39E64EE05C5A2B0B340
4862017FFBB350420F26F5C6C5B3402E
65BE14D25576A5B8D55860FA740A64D4
A2731A64B9B",6532
1300 DATA "BAE64C1F01A64D4A48484
83412E64E542406A6890100A780E7502
70AEC890100ED816A5026F6351230882
04A26DDC608A65E3414E64EA7805A26F
B3514308820",6159
1310 DATA "5A26EF8602A7403582347
68602A74017FBFF35F68DF3860AA740E
C4B9BBA1F01A65EE64D5858583414E64
EA7805A26FB3514308820SA26EF6F5B6
F5C8602A740",7038
1320 DATA "396F5E398601A75D39860
3A75D39811F241CA74C408B20A74E6F5
B6F5C6C5D398117240AA74B408B18A74
D6C5D396F5D16FE1E27F9A14E22F5A74
E6C5D3927EE",6100
1330 DATA "A14D22EAA74D6F5D3986F
FA75E39EC5B5A2A08E64E5A4A2A024C5
FED5B862017FEB96A5C39A65C4A2A03A
64E4AA75C39A65C4CA14E25014FA75C3
96A5B2A05A6",6510
1340 DATA "4D4AA75B39E65CC40750C
B088620340617FE8835065A26F639000
0000000000000808080800080000A0
A0A0000000000A0A1F0A1F0A0A00081
E281C0A3C08",2783
1350 DATA "0032320408102626001O2
828102A241A0008081000000000040
8101010080400080402020204080008Z
A1C3E1C2A08000008083E08080000000
00000080810",898
1360 DATA "000000003E00000000000
0000000808000202040810202000lC2
2262A32221C000818080808081C001C2
2020C10203E001C22020C02221C00040
C143E040404",956
1370 DATA "003E20203C02023C001C2
2203C22221C003E220404080808001C2
2221C22221C001C22221E02221C00000
8000080000000008080008081000040
81020100804",1190
1380 DATA "0000003E003E000000100
8040204081000lC2202040800008001C2
2021A2A2A1C001C22223E222222003C2
2223C22223C001C22202020221C00382
42222222438",1550
1390 DATA "003E20203C20203E003E2
0203C202020001C22202E22221C00222
2223E222222003E08080808083E000F0
2020202120C0022242830282422002O2
0202020203C",1833
1400 DATA "0022362A2A22222200223
2322A262622001C22222222221C003C2
2223C202020001C2222222A241A003C2
2223C282422001C22201C02221C003E0
80808080808",1898
1410 DATA "002222222222221C00222
22222141408002222222A2A362200222
2140814222200222214080808003E0
2040810203E00382020202020380020Z
01008040202",1548
1420 DATA "000E02020202020E00081
C3E08080808000008183E18080000000
0000000000000000C020E120D00202
03C2222223C0000001C2220221C00020
21E2222221E",961
1430 DATA "0000001C223E201C00040
A081C0808080000001C22261A021C202
02C3222222200080018080808lC00001
8080808082810202024283028240018O
8080808lC",1226
1440 DATA "000000764949494900000
02C3222222200000lC2222221C00000
02C32322C202000001A26261A0202000
02C322020200000001C201C021C00080
81C08080A04",1530
1450 DATA "000000222222261A00000
0222222140800000414149493600000
0221408142200000022221E02021C000
03E0408103E000408100810080400080
80800080808",1136
1460 DATA "000804020402040800000
020500A040000FFFFFFFFFFFFFFF4D4
F555345205645525349 4F4E20322E302
0434F505952494748542031393836204
25920535242",4776
1470 DATA "20534F465457415245204
14C4C2052494748545320524555455255
64544205055424C494320444F4D41494
E2042592053544556452042 4A4F524B2
00000000000",4038
2000 PMODE 0,1:PCLEAR 1:GOTO100
```



Listing 2: LINES

```
10 'LINES/BAS
20 'THIS IS A DEMO PROGRAM FOR
30 'THE MOUSE (2.0) PROGRAM FROM
40 'SRB SOFTWARE
50 'WRITEN BY STEVE BJORK
60 'YOU MUST HAVE MOUSE/BIN
70 'TO RUN THIS PROGRAM
100 CLEAR 200,&H33FF:MO=&H3400
110 ' ADD A ' TO THE START OF TH
E NEXT LINE IF ON A CASSETTE SYS
TEM
120 IF PEEK(MOUSE)<>22 THEN LOAD
M "MOUSE",MOUSE
130 'REMOVE THE ' FROM THE NEXT
LINE IF ON A CASSETTE SYSTEM
140 'IF PEEK(MOUSE)<>22 THEN CLS
:PRINT"LOADING MOUSE (2.0) BINAR
Y FILE":CLOAD"MOUSE"
150 DEF USR0=MOUSE
160 PMODE 4,1:PCLS0:SCREEN 1,1
170 FOR Y=0 TO 2:WINDOW$(Y)=CHR$
(4):FOR X=1 TO 4
180 READ A:WINDOW$(Y)=WINDOW$(Y)
+CHR$(A):NEXT X,Y
190 U=USR(0):PRINT CHR$(3);WINDO
W$(1);CHR$(12);" Exit    Clear";
WINDOW$(2);
200 PRINT WINDOW$(2);CHR$(2);CHR
$(12);"  ^        ^"
210 PRINT"  ^        ^";STRING$(
3,95);"Clears screen"
220 PRINT"  ^":PRINT"  ^";STRING
$(12,95);"Exits program";CHR$(13
)
230 PRINTTAB(13);"Lines":PRINTTA
B(9);"by Steve Bjork":PRINT
240 PRINT"Use the cursor to sele
ct the    start and end points o
f a line.":PRINT
250 PRINT"The top line has two o
ptions,   Exit-exit program,
          Clear- Clear the scree
n."
260 PRINT:PRINT"Press the button
 to continue."
270 IF USR(4)=0 THEN 270
280 GOSUB 500
290 U=USR(1)'TURN ON CURSOR
300 GOSUB 410:IF M=1 THEN 300
310 U=USR(0)'TURN OFF CURSOR
320 Y1=Y:X1=X
330 PSET (X*2,Y,1)
340 U=USR(1)'CURSOR ON
350 U=USR(8)
360 GOSUB 410:IF M=1 THEN 290
370 U=USR(0)'cursor off
380 LINE (X1*2,Y1)-(X*2,Y),PSET
390 U=USR(1):U=USR(7)
400 GOTO 290
410 TY=USR(3):TX=USR(2)
420 IF USR(4)=0 THEN 410
430 Y=USR(3):X=USR(2)
440 IF Y<>TY OR X<>TX THEN 410
450 IF Y>=9 THEN M=0:RETURN
460 IF X<24 THEN 510
470 IF X<32 THEN 410
480 IF X>56 THEN 410
490 M=1:U=USR(7):GOSUB500:U=USR(
1):RETURN
500 PRINT WINDOW$(2);CHR$(2);CHR
$(12);WINDOW$(0);:RETURN
510 'remove cursor and system
520 U=USR(0):U=USR(7):U=USR(6)
530 STOP
540 DATA 0,0,32,24
550 DATA 0,0,32,1
560 DATA 0,1,32,23
```

Listing 3: DISKTIME

```
0 ' DISKTIME/BAS
1 'PROGRAMED BY STEVE BJORK
2 'COPYRIGHT 1986 BY
3 'SRB SOFTWARE, U.S.A.
4 'MOUSE/BIN (VERSION 2.0) IS
5 'USED BY THIS PROGRAM.
6 'A 64/32K COLOR COMPUTER WITH
7 'A JOYSTICK OR MOUSE IS NEEDED
8 'FOR TESTING THE SPEED OF A
9 'DISK DRIVE.
10 'BEFORE A SPEED TEST CAN BE
11 'USED ON A DRIVE IT MUST BE
12 'SELECTED FIRST. JUST MOVE
13 'THE ARROW ON THE SCREEN OVER
14 'THE DISK TO BE TESTED AND
15 'PRESS THE BUTTON.
16 'A NEW DRIVE MAY BE SELECTED
17 'AT ANY TIME FROM THE MAIN
18 'MENU.
19 'THE AT THE BOTTOM OF THE
20 'SCREEN ARE 3 COMMANDS,
21 'DRIVE SPEED HISTORY,ADJUST
22 'DRIVE SPEED AND EXIT PROGRAM
23 'HISTORY GIVES A BAR GRAPH OF
24 'MOTOR SPEED.
25 'ADJUST IS FOR TUNING THE
26 'DRIVE SPEED.
27 'EXIT IS TO END THE PROGRAM
28 'PRESS THE BUTTON OR REMOVING
29 'THE DISK FROM A DRIVE IN
30 'TEST WILL EXIT THE TEST.
31 '
32 'THIS PROGRAM HAS BEEN PLACED
33 'PUBLIC DOMAIN BY STEVE BJORK
34 'AND SRB SOFTWARE FOR PRIVATE
35 'USE ONLY! THIS SOFTWARE MAY
36 'NOT BE USE (IN PART OR IN
37 'WHOLE) AS PART OF ANY
38 'MARKETED PRODUCT.
39 '
100 PCLEAR4:PMODE 4,1:PCLS1
110 CLEAR 50,&H737F:MOUSE=&H7380
:DIM D(3),P(60),C(23,17),MENU$(3
),A(7,7)
120 IF PEEK(MOUSE)<>22 THEN LOAD
M"MOUSE",MOUSE
130 DEF USR0=MOUSE:DEF USR1=&H7F
80
150 FOR X=&H7F80 TO &H7FFF:READ
U:POKE X,U:NEXT X
160 DEF FND(U)=INT((300-((U-2237
0)/74.56))*10+.5)/10
170 U=USR0(0)'LINK IN MOUSE
180 D(0)=41:D(1)=42:D(2)=44:D(3)
=104
190 SC=PEEK(&HBA)*256:SD=-1
200 FOR Y=SC TO SC+544 STEP 32
210 FOR X=Y TO Y+2:READ D
220 POKE X,D:NEXT X,Y
230 GET (0,0)-(23,17),C,G
250 FOR X=0 TO 2:READ MENU$(X):N
EXT
260 PCLS1:FOR Y=SC TO SC+224 STE
P 32
270 READ D:POKE Y,D:NEXT
280 GET (0,0)-(7,7),A,G
290 FOR Y=0 TO 1:WINDOW$(Y)=CHR$
(4):FOR X=1 TO 4:READ U:WINDOW$(
Y)=WINDOW$(Y)+CHR$(U):NEXT X,Y
300 PCLS1:SCREEN 1,1:PRINT WINDO
W$(0);CHR$(3);CHR$(12);:PRINT@19
9,"Disk Drive Selector";:FORD=0T
03
310 GOSUB 2000:NEXT:LINE (0,46)-
(255,100),PRESET,B:LINE(0,57)-(2
55,57),PRESET
320 PRINT@36,"Disk Drive Speed T
ester":PRINT@73,"by Steve Bjork"
330 PRINT@97,"Copyright 1985 by
SRB Software";
340 LINE(0,4)-(255,36),PRESET,B
350 U=USR(0):GOSUB2100:PRINT@522
,"Command Menu"
360 FORX=0TO2:PRINT CHR$(3);:GOS
UB2050:NEXTX
370 LINE(0,123)-(255,171),PRESET
,B:LINE(0,138)-(255,138),PRESET
380 IF SD=-1 THEN PRINT@420,"Ple
ase select disk drive"
400 U=USR(4):U=USR(1):U=USR(7)
410 IF USR(4)=0 THEN 410
420 Y=USR(3):X=USR(2)
430 IF Y<61 THEN 410
440 IF Y>78 THEN 600
450 A=X AND 31:IF A<12 OR A>23 T
HEN 410
460 A=INT(X/32):IF A=SD THEN 410
470 U=USR(0)+USR(8)
480 IF SD<>-1 THEN D=SD:SD=A:GOS
UB 2000
490 SD=A:D=SD:GOSUB 2000
500 GOSUB 3000:GOSUB 2100:PRINT@
551,"Testing for Drive ";CHR$(SD
+48);
510 FOR X=1 TO 300:NEXT
520 U=USR1(0):GOSUB3010:IF U>0 T
HEN 350
530 GOSUB 2100:PRINT@518,"Disk D
rive not on line";:PRINT@548,"or
 a disk is not in drive";
540 PRINT@612,"PRESS BUTTON TO C
ONTINUE";:U=USR(4)+USR(7)
550 IF USR(4)=0 THEN 550
560 D=SD:SD=-1:GOSUB 2000
570 GOTO 350
600 IF Y<144 OR Y>167 THEN 410
610 Y=INT((Y-144)/8)
620 A=LEN(MENU$(Y))*2
630 IF X<64-A OR X>64+A THEN410
640 IF SD=-1 AND Y<2 THEN 410
650 U=USR(0):X=Y:PRINT CHR$(2);:
GOSUB2050
660 U=USR(7):IF X=2 THEN 999
670 GOSUB 3000:FORX=1 TO 300:NEX
T
680 IF USR1(0)=0 THEN GOSUB 3010
:GOTO 530
690 IF Y=0 THEN 800
700 GOSUB 1000:A=-1:U=USR(4)
710 U=USR1(0):IF U=0 THEN GOSUB
3010:GOTO 350
720 IF USR(4)<>0 THEN GOSUB 3010
:U=USR(8):GOTO 350 ELSEP=FND(U):
PRINT@520,USING" ###.#  ##.#!";P
;INT(ABS(300-P)*3.33)/10;"%";
730 IF P<272 THEN P=270 ELSE IF
P>327 THEN P=327
740 P=INT((P-270)*4.2666):IF P=A
 THEN 710
750 IF A>-1 THEN LINE(A-2,152)-(
A+5,159),PSET,BF
760 A=P:PUT(A-2,152)-(A+5,159),A
,PSET:GOTO 710
800 GOSUB 1000:FOR X=0 TO 60:P(X
)=0:NEXT:U=USR(4)
810 U=USR1(0):IF U=0 THEN GOSUB
3010:GOTO 350
820 IF USR(4)<>0 THEN GOSUB 3010
:U=USR(8):GOTO 350
830 P=INT(FND(U)-270):IF P<0 THE
N P=0 ELSE IF P>60 THEN P=60
840 IF P(P)>=40 THEN 810
850 P(P)=P(P)+1
860 LINE(P*4+8,160-P(P))-(P*4+9,
160-P(P)),PRESET
870 GOTO 810
999 U=USR0(6):END
1000 GOSUB 2100:LINE(0,160)-(252
,162),PRESET,BF:FOR X=8 TO 248 S
TEP 40:LINE(X,162)-(X,168),PRESE
T:NEXT:FOR X=0 TO 252 STEP 4:LIN
E(X,162)-(X,165),PRESET:NEXT
1010 PRINT@672,CHR$(3);"270  280
  290  300  310  320 330";CHR$(1
);CHR$(4);CHR$(13);"Press Button
 to Exit Test";:RETURN
2000 IF SD=D THEN PRINT CHR$(2);
:PUT(24+D*64,61)-(47+D*64,78),C,
PRESET ELSE PRINT CHR$(3);:PUT(2
4+D*64,61)-(47+D*64,78),C,PSET
2010 PRINT@322+D*8,"Drive";CHR$(
1);CHR$(D*8+4);CHR$(11);CHR$(48+
D);:RETURN
2050 PRINT CHR$(1);CHR$(16-LEN(M
ENU$(X))/2);CHR$(18+X);MENU$(X);
:RETURN
2100 PRINTWINDOW$(1);CHR$(3);CHR
$(12);WINDOW$(0);:RETURN
3000 POKE &HFF40,D(D):RETURN
3010 POKE &HFF40,0:RETURN
9000 DATA 77,39,17,57,255,255,25
5,255,0,166,140,252,39,5,174,140
,245,175,106,59,52,113,26,80,254
,1,10,239,140,230,51,140
9010 DATA 232,255,1,10,48,140,55
,175,140,220,111,140,219,142,0,0
,182,255,72,16,142,127,255,206,2
55,72,198,224,99,140,201,247
9020 DATA 255,72,30,136,30,136,1
98,2,229,196,38,9,49,63,38,248,1
11,140,181,32,18,48,1,38,252,111
,140,172,32,9,182,255
9030 DATA 72,132,124,31,16,32,14
,134,208,183,255,72,30,136,30,13
6,182,255,72,79,95,174,140,140,1
91,1,10,53,113,126,180,244
10000 DATA 0,0,0,63,255,252,63,2
55,240,63,255,243,63,255,240,63,
255,252,63,195,252,63,129,252,63
,129,252,63,195,252,63,255,252,6
3,255,252,63,231,252,63,231,252,
63,231,252,63,231,252,63,255,252
,0,0,0
10010 DATA "Drive Speed History"
,"Adjust Drive Speed","Exit Prog
ram"
10020 DATA 239,239,239,239,1,131
,199,239
10100 DATA 0,0,32,24, 0,14,32,10
```

A powerful utility for more readable listings

I Can See Clearly Now

By Lynn Sundberg

SRLIST is a utility program intended to give a more powerful version of LLIST to the BASIC programmer for debugging or documentation of a program. It gives two types of printout: a 32-column screen image like THE RAINBOW's format, or a by-statement column.

To illustrate how the by-statement printout can help in tracing program logic, I have taken the liberty of running Harris Allen's one-liner contest winner (RAINBOW, December 1984, Page 212) through *SRLIST* to produce the following printout:

```
0 IFA<>0THENFORA=1TO4:
    FORB=0TO1:
    X=X+A(A):
    Y=Y+A(A+1):
    IFPPOINT(X,Y)=5THENPRINTC;"TURNS
    ":
    ELSEPSET(X,Y):
    IFINKEY$=""THENB=0:
    NEXT:
    ELSEC=C+1:
    NEXT:
    NEXT:
    GOTO0:
    ELSEIFINKEY$=""THENPRINT"CRAM/PR
    ESS KEY":
    GOTO0
  ELSEIFMODE4,1:
    PCLS:
    SCREEN1,1:
    A(2)=1:
    A(4)=-1:
    A=1:
    GOTO0
```

There have been simpler programs that give the two formats, but much of *SRLIST*'s value comes from niceties such as a two columns per page printout, page numbering, run date and program title. Unlike most pretty print programs that are a machine language routine residing in memory along with the program being printed, *SRLIST* is a stand-alone program that uses a program saved in ASCII format (SAVE "NAME",A) as an input file.

To make *SRLIST* compatible with as many CoCo configurations as possible, hardware requirements and hardware control coding have been kept to a minimum. The program works on a 16K CoCo, yet Line 5 uses all available standard BASIC memory in a 64K machine. It does require ECB and an 80-column printer.

As written, the program is for a disk system. For cassette systems, make the following modifications: change Line 110 to: PRINT@416,"SET CASSETTE TO START OF ";X$:INPUT Z$:PRINT@416,B$:PRINT; change the two #1s in Line 115 to #-1; change the (1) in Line 125 to (-1); change the #1 in Line 130 to #-1.

Printer codes are used in Line 10 to set printer tabs to 1 and 40, while the code in Line 350 positions the printer at the 40th space. Line 360 contains a code for "top of form." You may have to change these codes for your printer. Or remove all printer codes with the following changes:

1) Delete Line 10
2) Replace Line 355 with

```
PRINT#-2,LEFT$(L$(Y)+SS$,39)
;:L$(Y)=B$:PRINT#-2,L$(Y+Z
+1):L$(Y+Z+1)=B$:NEXT
```

3) The FOR statement in Line 360 should read

```
FOR X-=0 TO ?
```

where the question mark is replaced with whatever number gets you to the start position of the next page. A little experimentation will be required to find this number, but it will be a small number.

Removing the printer codes causes the program to run slower because it fills memory with old print lines and stops to clean itself every page or so.

To use *SRLIST*, save the program to be printed in ASCII format and run *SRLIST*. A screen displays reminding you the program being printed out must be in ASCII format. The screen then clears and requests input for program parameters.

It asks for the date and the program name. If the program uses the BAS default extension, *SRLIST* automatically adds it to the program name.

Next it requests type of run, either by-statement columns or 32-character columns with a default to the by-statement run. The next parameter is paper type. Continuous paper is the default value and once started, the program runs to completion. If the single-sheet option is chosen, the computer stops and prompts you to enter a new sheet of paper for each page.

Following the program parameters, you are asked for three sub-parameters that can be changed after each run made on the same program. The first sub-parameter is the number of the first line to be printed. The second is the the end line. If the end line is less than the first line, the program defaults to 9999. Line numbers higher than 9999 can be processed, but in the printout only the four right-hand digits are printed. The third is the starting page number.

These sub-parameters allow printing of a portion of a program, or reprinting a program section while keeping page continuity.

continued on Page 37

ADD AN EXTRA 22K OF MEMORY TO DISK BASIC 1.1

DISCOVER THE 'HIDDEN' FIVE TRACKS

by Jim Peake

If you own the new white Radio Shack disk drive (catalog number 26-3029) with Disk BASIC 1.1, you may have a 40-track drive masquerading as a 35-track drive. The new drives without the little tab to eject the disk are different from the old gray drives.

After more than three years of cassette loading, I broke open the piggy bank and bought a new drive in December 1983. Sure enough, it looks different. Rumor has it that it's a Model 4 drive from TPI. After six months of loyal service on my 1980-vintage D-board CoCo, I attached it to my new 64K CoCo 2. After experimenting to see how many tracks it really has, I discovered it works fine as a 40-tracker.

I've written a small program of POKEs to modify the disk ROM for 40 tracks. To use it, you must have a 64K CoCo. You must also have a program to load the BASIC ROMs into the upper 32K of RAM, and switch the CoCo to the all-RAM mode. And you must have the new Disk BASIC 1.1 ROM. This change does not work with the 1.0 ROM because all the POKE addresses are different and you would need 40-track drives rather than the 35- or 36-track gray drives.

So how do we make Disk BASIC 1.1 use 40 tracks? There are several things that must be done. First, move the ROMs to RAM and switch to the all-RAM mode so we can make changes with the POKE command. Second, BASIC must be modified to initialize 40 tracks with the DSKINI command. Third, the file allocation table, which keeps track of what granules have been used, must be enlarged to keep track of 78 granules instead of 68. Changes have to be made so it is all saved to disk and not cut off at 68. Next, FREE has to be modified to look for 78 granules.

I did not make the directory allow for more than 68 files because, as a practical matter, I cannot believe anyone would want that many files on one disk.

The file allocation table is easy enough to expand, but doing so limits the system to a maximum of three drives. This is because the system initialization routine reserves space for up to four file allocation tables. Since each table is 10 bytes bigger to hold 78 instead of 68 granule pointers, there is not enough room for more than three file allocation tables. Again, it's much too complicated to move all subsequent work addresses and not worth the trouble.

The advantages of this modification for using 40 tracks are many. You can save money by using fewer disks. This modification is fully compatible with the diskettes already formatted in the official 35-track format. They operate normally while in the 40-track mode and are protected from unsuccessful attempts to use the last five tracks. The file allocation table indicates to the modified 40-track system that those tracks are not available. BACKUP is the only command that invites disaster if you try to use 35-track diskettes while in the 40-track mode.

There are some disadvantages to this 40-track modification. If you run the system without loading the modification, the unmodified system wreaks havoc with 40-track diskettes. It shortens the file allocation table to 68 granules, thus losing forever the last five tracks. Any files or programs that used any or all of that area are unusable, and the diskette will be a 35-track disk until reinitialized to 40 tracks, which wipes out everything on the diskette. It is also a nuisance to have to load the necessary programs for converting to 40 tracks each time you turn on the computer.

Of course, if you have an older drive that does not support 40 tracks, attempts to use the last five tracks result in an I/O error. And if Radio Shack or its supplier adjusts the drives to prevent using the last five tracks, then this modification won't work. I think the system is set up for upgrading to 40-track drives and Radio Shack probably will. If they do, they will have to put in a few more checks than I did to ensure full compatibility with diskettes previously formatted for 35 tracks.

One last caution: This modification has not been rigorously tested in a variety of environments. Therefore, experiment carefully and test it to be sure it works dependably on your system before entrusting valuable data or hard work to it.

One does not start disassembling the ROMs without a good, comprehensive

map of what's what. In my case, the map and guiding light was the complete CoCo memory map published in the July, August, September and December 1983 issues of THE RAINBOW.

In conclusion, the new 1.1 system can easily be modified to use 40 tracks for storing up to 178,000 bytes of data on each diskette.

The listing: TRACK40

```
1ØØØ REM ROUTINE TO SET DISK
     BASIC 1.1 TO 4Ø TRACKS IN
     A 64K COCO
1ØØ1 ' WRITTEN JUNE 1984
     BY JIM PEAKE
1ØØ2 ' NOTE:: LIMITS TO A MAX OF
     3 DRIVES
1ØØ5 POKE&HD762,&HØ2 ' SETS
     DSKCON TO RETRY 2 TIMES
     INSTEAD OF 5. OPTIONAL MAY
     BE OMITTED.
1Ø1Ø POKE&HD65F,&H28:POKE&HD682,
&H28  ' SETS DSKINI TO 4Ø TRACKS
1Ø15 POKE&HD534 &H27 ' SETS DSKI
1Ø2Ø POKE&HD29D,&H28 ' SETS DSKO
1Ø3Ø POKE&HC735,&H4E ' SETS MOVE
     OF FILE ALLOCATION TABLE TO
     1Ø MORE GRANULES
1Ø35 POKE&HC75A,&H54 ' INCREASES
     SPACE FOR EACH DRIVE'S FAT
     TO ALLOW 1Ø MORE GRANULES
1Ø4Ø POKE&HC7BB,&H4E ' SETS MOVE
     OF FAT TO 1Ø MORE GRANULES
1Ø45 POKE&HC7DØ,&H4E ' USE 78
     GRANULES NOT 68
1Ø5Ø POKE&HC7EF,&H4E ' USE 78
1Ø55 POKE&HCD26,&H4E ' SEARCH 78
     GRANULES
1Ø6Ø POKE&HCEB5,&H4E ' LET FREE
     COMMAND CHECK 78 GRANULES
1Ø65 POKE&HD44D,$H4E ' SET COPY
TO 78 GRANULES
1Ø8Ø POKE&HD7CØ,Ø:POKE&HD816,&H1
4 ' SET TO 6MS STEPPING RATE
```

continued from Page 35

The listing: SRLIST

```
5 CLEAR5Ø:CLEAR MEM-1ØØØ
1Ø PRINT#-2,CHR$(27);"D";CHR$(1)
;CHR$(4Ø);CHR$(Ø)
15 WC$="WORKING":WS$="working":S
P=1
2Ø B$="":S$=" ":S5$="      ":SS$=
STRING$(41," "):E$="ELSE":C$=":"
:I$="IF":R$="'":RR$="REM":Q$=CHR
$(34)
25 L=-1:DIM L$(1Ø3)
3Ø 'START
35 CLS:PRINT@33,"SRLIST BY L.SUN
DBERG":PRINT@97,"S-LIST GIVES A
BY STATEMENT":PRINT@131,"PRINTOU
T"
4Ø PRINT@193,"R-LIST GIVES A 32
CHARACTER":PRINT@227,"PRINTOUT"
45 PRINT@321,"SRLIST IS A STAND
ALONE":PRINT@353,"PROGRAM BUT DO
ES REQUIRE THAT":PRINT@385,"THE
PROGRAM BEING LISTED":PRINT@417,
"BE IN askii FORMAT"
5Ø FORX=ØTO15ØØ:NEXT
55 CLS:PRINT@1Ø5,"DATE ";:INPUT
DA$:PRINT@11Ø,DA$:PRINT@129,"PRO
GRAM NAME ";:INPUT PN$:PRINT@142
,PN$
6Ø PRINT@165,"TYPE RUN  S-STATEM
ENT":PRINT@2Ø7,"R-32CHARACTER":I.
NPUTX$
65 PRINT@2Ø7,B$:PRINT:IFX$="R" T
HENT=1:PRINT@175,"R-32CHARACTER"
7Ø PRINT@195,"TYPE PAPER  C=CONT
INUOUS":PRINT@239,"S-SINGLE SHEE
T":INPUTX$
75 IFX$="S" THEN PRINT@2Ø7,"S-SI
NGLE SHEET":TP=1
8Ø PRINT@224,B$:PRINT:PRINT
85 'RUN LOOP LIMITS
9Ø PRINT@291,"START LINE ";:INPU
TSL:PRINT@3Ø2,SL
95 PRINT@325,"END LINE ";:INPUTX
:IFX>SL THEN EL=X ELSE EL=9999
1ØØ PRINT@334,EL:PRINT@355,"STAR
T PAGE ";:INPUTX:IFX>ØTHEN SP=X
1Ø5 PRINT@366,SP:X$=PN$
11Ø X=INSTR(X$,"."):Y=INSTR(X$,"
/"):IFX=ØAND Y=ØTHEN X$=X$+".BAS
"
115 OPEN"I",#1,X$:INPUT#1,X$
12Ø ' READ LOOP
125 GOSUB32Ø:IFEOF(1) THEN38Ø
13Ø LINEINPUT#1,X$:X=INSTR(X$,S$
):A=VAL(LEFT$(X$,X))
135 IFA<SL THEN125ELSE IFA>EL TH
EN38Ø
14Ø IFT=ØTHEN175
145 ' PROCESS 32 CHAR LINE
15Ø L=L+1:IFL>1Ø3THEN GOSUB335
155 IFLEN(X$)>32THEN165
16Ø L$(L)=X$:GOTO125
165 L$(L)=LEFT$(X$,32):X$=MID$(X
$,33):GOSUB32Ø:GOTO15Ø
17Ø ' PROCESS BY STATEMENT
175 P$=RIGHT$(S5$+LEFT$(X$,X),5)
:X$=MID$(X$,X+1):XX=1:A=Ø
18Ø I=INSTR(XX,X$,I$):IFI>ØAND I
<3ØR A=1 THEN 245
185 C=INSTR(XX,X$,C$):IFC>ØTHEN
195
19Ø P$=P$+X$:GOSUB4Ø5:GOTO125
195 Q=INSTR(XX,X$,Q$):R=INSTR(XX
,X$,R$):RR=INSTR(XX,X$,RR$):IFRR
>ØAND R>RR THENR=RR
2ØØ IFQ>ØTHEN23Ø
2Ø5 IFR>ØTHEN225
21Ø P$=P$+LEFT$(X$,C):GOSUB4Ø5:P
$=S5$:IFA=1THEN P$=P$+S$+S$
215 XX=1:IFC=LEN(X$) THEN125
22Ø X$=MID$(X$,C+1):GOTO18Ø
225 IFR<C THEN19ØELSE21Ø
23Ø IFQ>R AND R>ØTHEN225
235 IFQ>C THEN21Ø
24Ø XX=INSTR(Q+1,X$,Q$)+1:IFXX=1
OR XX=LEN(X$)+1 THEN19ØELSE185
245 A=1:E=INSTR(XX+1,X$,E$):IFE=
ØTHEN185
25Ø C=INSTR(XX,X$,C$):Q=INSTR(XX
,X$,Q$):R=INSTR(XX,X$,R$):RR=INS
TR(XX,X$,RR$):IFRR>ØAND R>RR THE
N R=RR
255 IFC>Ø AND C<E THEN 2ØØ
26Ø IFQ>ØTHEN3ØØ
265 IFR>ØAND R<E THEN 225
27Ø C=INSTR(XX,X$,C$)
275 IFC=ØOR C>E THEN29Ø
28Ø P$=P$+LEFT$(X$,C):GOSUB4Ø5
285 P$=S5$+S$+S$:X$=MID$(X$,C+1)
:XX=1:GOTO245
29Ø P$=P$+LEFT$(X$,E-1):GOSUB4Ø5
295 P$=S5$:X$=MID$(X$,E):XX=1:GO
TO245
3ØØ IFR>ØAND R<Q THEN265
3Ø5 IFQ>E THEN27Ø
31Ø XX=INSTR(Q+1,X$,Q$)+1:E=INST
R(XX,X$,E$):IFE=ØTHEN185ELSE25Ø
315 ' WORKING LOOP
32Ø IFW=ØTHEN W=1:PRINT@46Ø,WC$
ELSE W=Ø:PRINT@46Ø,WS$
325 RETURN
33Ø ' PRINT PAGE LOOP
335 PRINT@448,B$;:IFTP=1THEN INP
UT"INSERT PAPER - <ENTER>";Y$ EL
SE PRINT
34Ø Y$=LEFT$(PN$+SS$,4Ø)+RIGHT$(
SS$+DA$+" PRINTOUT",4Ø):PRINT#-2
,Y$:PRINT#-2:PRINT#-2
345 IFL>1Ø3THEN Z=51ELSE Z=L/2
35Ø FORY=ØTO Z
355 PRINT#-2,L$(Y);:L$(Y)=B$:PRI
NT#-2,CHR$(9);L$(Y+Z+1):L$(Y+Z+1
)=B$:NEXT
36Ø FORZ=Y TO53:PRINT#-2:NEXT:PR
INT#-2,RIGHT$(SS$+SS$+"PAGE"+STR
$(SP),8Ø):SP=SP+1:L=Ø
365 PRINT@448,B$:PRINT#-2,CHR$(1
2)
37Ø RETURN
375 ' END ROUTINE
38Ø CLOSE:GOSUB335
385 PRINT@448,"ANOTHER RUN";:INP
UT" Y/N";X$
39Ø IFX$="Y" THEN FORX=256 TO 41
6 STEP32:PRINT@X,B$:NEXT:L=-1:GO
TO9Ø
395 CLEAR2ØØ:CLS:END
4ØØ ' STORE STATEMENT LINE
4Ø5 L=L+1:IFL>1Ø3THEN GOSUB335
41Ø GOSUB32Ø:P=LEN(P$):IFP>39THE
N42Ø
415 L$(L)=P$:RETURN
42Ø L$(L)=LEFT$(P$,39):P$=S5$+MI
D$(P$,4Ø):IFA=1THEN P$=S$+S$+P$
425 GOTO4Ø5
```

TRANSPLANT SURGERY FOR THE DISK CONTROLLER

by Kerry M. Armstrong

THE CoCo disk controller manufactured by Tandy / Radio Shack Corporation contains a 24-pin ROM chip in which Disk BASIC, a very simple disk operating system, resides. Better disk operating systems for the CoCo have been designed by individuals and third party vendors. These DOSs can usually be customized to utilize such things as 40-track and double-sided drives, faster step rates, additional commands and/or ulitities, ect,.

While most of these tasks can be accomplished with the use of short machine language programs or POKEs, they generallymust be loaded and executed every time the computer is turned on or Reset, and are often not worth the trouble to use. However, if these same patches could be placed permantely into ROM with the present at every startup of the computer. Using an Erasable Programmable Read Only Memory, EPROM, they can be permanently installed.

The pin-for-pin EPROM available for the 24-pin DOS socket on the disk controller board is the 68764 or 68766. (Because EPROMs are erasable, they can be erased, programmed and used many times.) However, there does exist a series of EPROMs (2764 and 27128) that can be utilized. The only drawback is that these are 28-pin chips and thus are not pin-for-pin compatible with the 24-pin socket on the disk controller board, which means you can't just pluck out the old DOS-ROM and stick one of these in its place.

Instructions were provided for the modification of a 28-pin EPROM and a 28-pin chip socket that, when wired together, could be plugged directly into the 24-pin RS-DOS socket. (See "Cooking with CoCo", by Colin J. Stearman, Australian Rainbow June 1985 Page 48. This method is an effective way to adapt a 2764 EPROM to be used in the disk controller, and does have the advantage that you may always pull the EPROM/ADAPTER out of the socket and put the old disk back in.

On the other hand, this method also has a couple of distinct disadvantages. First, the adapter method requires that you do some soldering directly to some of the EPROM pins, which could, if you're not careful, result in damage to the EPROM. Secondly, the adapter method does not allow you to take advantage of nearly 8K additional memory space that is available in the DOS area and could be utilized for additional DOS commands and/or in ROM utilities.

The 2764 is an 8K EPROM and would occupy memory locations &HC000 to &HDFFF, as does Disk BASIC. Memory locations &HE000 to &HFEFF are wasted space on the CoCo (about 8K). This space contains a mirror image of Disk BASIC.

The 27128 is a 16K EPROM and the additional memory can be utilized by it. However, it will not work properly in the adapter because there is no access line to the upper 8K of memory in the 27128.

In light of this diagnosis, we come to the purpose of this treatise, that of doing permanent surgical transplant modification on the RS-Disk Controller so that 2764 and 27128, 28-pin EPROMs, may be used.

(NOTE: Tandy/Radio Shack has marketed three different CoCo disk controllers to date. With the manufacture of the second and third controller board, they have left off several of the "lands". The third and newest type of controller board does not have the necessary land to access the upper 8K on a 27128 EPROM. The required land is number 37 as indicated in Figure 3.)

Risk Disclosures

A good surgeon always warns of the possible ricks and side effects.

1) Same .old .thing you've heard before: opening anything made and sealed by Radio Shack voids the warranty.

2) Do not attempt this surgery unless you have some skill at PC board soldering.

3) Always use proper CMOS handling procedures by properly grounding yourself and the PC board. (Which means don't get all static-ey on this project).

The Transplant Surgery

Let's get down to the business of doing a socket transplant on the disk controller. We need some wire-wrap wire (RS# 278-501) and the all important 28-pin socket. There are three types of 28-pin sockets, only one of which will do for our project. We don't want the wire-wrap type. We want the type that has its pins running parallel to the sides of the socket. In other words, exactly like the pins on the EPROM.

Once you have obtained these items, you need to get your tools together and set up the operating room.

Pre-Op

Prepare the 28-pin socket for transplanting into the controllerby doing the following steps, preferably in order:

1) Prepare a six-inch piece of wire-wrap wire by stripping off approximately 1.5 inches of insulation from one end.

2) Look at the computer-mating end of the disk controller to determine if land 37 is missing (type 3 controller board). If it missing, then the procedure is a little different and is so noted.

3) Carefully solder the bare part of the wire to near the tops of pins 27, 28 and 1 (and 26 if type 3 board). Start at 27 (26 on type 3 board) with the wire and end at pin 1, so that the remaining 4.5 inches of insulated wire run off pin 1. It helps to wrap the wire around the pins before soldering. CAUTION: Do not apply too much heat for too long as the socket will melt and be unusable. (See Figure 1)

4) If you have a controller board with a land 37, solder another piece of 6inch wire near the top of pin 26.

5) Solder one last piece of 6inch wire near the top of pin 2.

6) Finally, if you have the type 3 board, clip off the excess wire coming from pin 1 and clip off pins 27, 28, 1 and 2 just below the soldered connections. Do not clip off pin 26.

6a) If you have one of the other two boards, do not clip off the wire coming from pin 1, but clip pins 26, 27, 28, 1 and 2 just below the soldered connections.

The Operation

The 28-pin socket prepared for transplanting, it is time to bring ourpatient, the disk controller, into the operating room. Prep the patient by carefully peeling back the outer skin label, locate the joining screw and remove it. Then, with consummate surgical skill, spring the plastic looking tabs at one end of the controller, thus exposing the internal organs of your patient. Next carefully pry out the 24-pin DOS-ROM chip and set it aside on a piece of conductive foam (RS# 276-2400) for safekeeping.

The next step is the most difficult, and the biggest test of surgical skills. You must desolder and remove the 24-pin chip socket. Take a deep breath and with soldering iron in one hand and solder sucker in the other, begin. Once you have done this, the operation is halfway over.

Once the removal process is completed, only a few additional preparatory steps remain before the actual transplant phase of the surgery begins. You should next execute the following surgical procedures. Next, on the controller board, locate and cut the trace that went to pin 18 of the 24-pin socket. Make sure you completely cut the trace so that there is no remaining contact. Locate and cut the trace that went to pin 21 of the 24-pin socket. Place a small piece of elect- -rical tape on the controller board across the end of where the old socket sat, to provide an extra safe guard of insulation from accidental contact with the PC board. If you do not have a type 3 board, the tape should also cover the hole for pin 24 of the old 24-pin socket (See Figure 2).

The Transplant

We now turn to the actual transplant of the 28-pin chip socket into the disk controller. It is here you demonstrate your surgical skills by re-connecting the arteries of the PC board that carry the date lines to the brains of the disk operating system. Perform the following steps.

1) Insert the 28-pin socket into the holes left by the 24-pin socket with the rear of the new socket in the same position as the old was (pin 14 to 12, 15 to 13, etc.). (Ssee Figure 3)9.

2) Solder the socket in place being careful not to create any solder bridges between the pins.

3) If you have the type 3 board, go to step five, otherwise cut the wire coming off socket pin 1 to the necessary length and solder it near the end of land 9, (a+5v source). (See Figure 4).

4) Likewise solder the wire from pin 26 to land 37.

5) Solder the wire from pin 2 to land 31.

6) On the PC board, solder a jumper wire from the solder pad for the trace that went to former pin 18 and connect it to pin 23 of the 28-pin socket.

7) Ssolder a jumper from pin 14 of the 28-pin socket to pin 20 of the 28-pin socket.

At this point, it would be very helpful to have a nurse around to wipe your brow. However, as this is a cheap operation, you'll have to do that yourself.

Post-Op

As with every good surgical procedure, it is necessary to inspect your work before colsing up. Checkc your patient's internals for loast tools, bandages, swabs, etc. Be particularly careful to inspect all solder connections and make sure you have no bare wires touching where they shouldn't.

Finally, insert your programmed 28-pin EPROM, making sure the notches are all oriented the same way, and close up the case, installing the screw.

Physical Therapy

Now to see if it works. Hook up all the cables, plugs, drives, computer, etc. Place your finger on the computer'7s on/off button. Take a deep breath and push. If all went well, your DOS sign-on message should be displayed. If not, shut it down, check the patient for insurance and go back to square one.

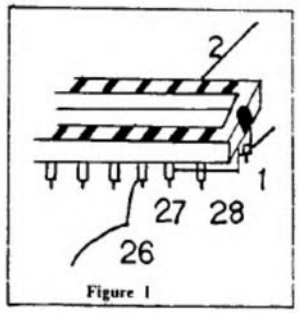


Figure 1

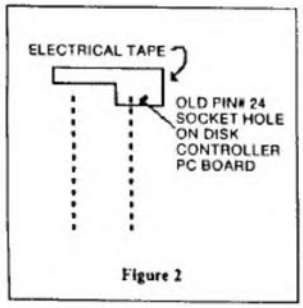


Figure 2

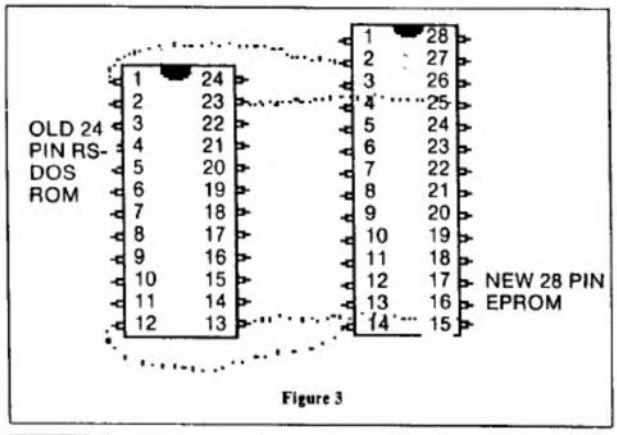


Figure 3

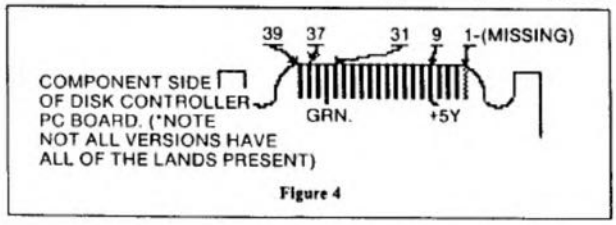


Figure 4

INVESTIGATING THE PIA

by Tony DiStefano

THIS month I'm looking into a PIA. The letters PIA stand for Peripheral Inte-rface Adapter. The Color Computer uses two of the PIAs. The older, regular CoCo uses two MC6821 PIAs. The newer CoCos and the CoCo 2s use one MC6821 and one MC6822. The differences between the two are minor. The 6822 is called an IIA. This stands for Industrial Interface Adapter. Both have the same pinout and function in the same way. I will describe the differences between them later in this article.

It's interesting to know what PIA stands for, but what does it do? A PIA provides the means of interfacing external hardware or devices to a computer. In our case, the MC6809 CPU. Most devices do not conform to the specifications of a CPU. Take for instance, a switch. It turns on the lights, stove, radio and so forth. It works well, but is not computer compatible. This is where a PIA comes in. It's a go-between from the CPU to the switch. With a PIA and a little circuit, the CPU can tell if the switch is on or off. Or in CPU terms, a zero or a one. This known as an input. If the computer had to control a light or a motor, a PIA would be used to switch a transistor on or off and, in turn, the transistor would control a relay and the relay would turn the moter on and off. This is known as an output.

This particular PIA has two bidirectional eight-bit peripheral data buses for interface to external devices and four individually controlled interrupt input lines, two of which can be used as outputs. It also has programmed controlled interrupt and interrupt disable capability plus two control registers and two data direction registers.

The PIA, like many other devices, looks like memory to the CPU. Therefore, the PIA must have address lines, data lines and control lines such as chip enable and read/write. Figure 1 shows the pinout of an MC6821 PIA chip. You should, by now, recognize many of the pins and there names. The following is a pin-by-pin description of this chip.

Vss - Signal ground. A reference to which all other signals are measured.

PA0 to PA7 - The first eight peripheral data lines, which can be programmed as outputs or inputs.

PB0 to PB7 - The second eight peripheral data lines, which can be programmed as outputs or inputs.

CB1 - Is an input only line that sets the interrupt flags of the B control register.

CB2 - Is either an interrupt input line or a periphal control output line.

Vcc - This is the five volt input that powers the chip.

R/W - This input controls the PIA as a read or a write to the registers.

CS0 — Chip Select 0 is an active high input. When this pin is low the chip is disabled.

$\overline{\text{CS2}}$ — Chip Select 2 is an active low input. When this pin is high, the chip is disabled.

CS1 — Chip Select 1 is an active high input. When this pin is low, the chip is disabled.

E — This is the Enable clock or the 'E' clock. Used to enable input or output.

D7 to D0 — These are the eight data lines that the CPU uses to read and write data to the PIA.

$\overline{\text{RESET}}$ — This active low input initializes the PIA to power up conditions.

RS1 — This input is the second address line used to access one of four locations on the PIA.

RS0 — This input is the first address line used to access one of four locations on the PIA.

$\overline{\text{IRQB}}$ — This active low output is used to generate an interrupt to the CPU from port B. The method of interrupt depends on how the control register is set up.

$\overline{\text{IRQA}}$ — this active low output is used to generate an interrupt to the CPU from port A. The method of interrupt depends on how the control register is set up.

CA2 — Is an input only line that sets the interrupt flags of the B control register.

CA1 — Is either an interrupt input line or a peripheral control output line.

First let me talk about the structure of the PIA. Basically, there are two ports. Each port has two control lines. Each PIA has two address lines and takes up four memory locations in the CPU's memory map.

Table 1 shows the memory map of a PIA. Address locations 0 and 2 are ports A and B respectively. Address locations 1 and 3 are control registers A and B respectively. I hope by now you can recognize addresses by binary bits. It may be a little confusing as to what CRA2 and CRB2 have to do with the memory map. There are actually six

registers to a PIA. But, if you remember your binary math, six is not an even power.

The designers could have added another address line and wasted the other two address locations. But instead, they put a software switch in the control register. Bit 2 to be exact. When the switch (bit 2) is low (zero) then address 0 or 2 becomes a data direction register. If you write a one in any bit position in that register, that bit becomes an input. On the other hand, if you write a zero, that bit becomes an output.

After all bits have been selected as ins or outs, then turn the switch at CRA2 or CRB2 back to a one. Now the 0 and 2 addresses become input and output peripheral ports as programmed.

The next part of the PIA is a little more complex. This includes control bits and interrupts. Along with the two eight-bit ports, this PIA also has four other pins. There are two pins used for inputs or outputs and there are two pins that are inputs only. These four pins work in conjunction with the bits in the control register of the PIA. Table 2 explains the bit names of control register A (CRA) and control register B (CRB).

Let's look at CA1 and CB1 first. They are inputs only. On given conditions, these inputs generate an interrupt. Bits 0 and 1 in the respective control registers have the following influence on the interrupts. If bits 0 and 1 are both low (either register), the interrupts are disabled and no interrupts go through. Only the interrupt flags are set on the falling edge of the input. If bit 1 is low and bit 0 is high, the falling edge of the CA1 or CB1 input causes an interrupt and sets the flag. Bit 1 high and bit 0 low sets the flag on the rising edge of the input but does not cause an interrupt. Bit 1 high and bit 0 high causes an interrupt and sets the flag on the rising edge of the input. The CA1 and CB1 interrupt flags are on bit 7 of the respective control byte. In other words, bit 1 enables or disables the interrupts and bit 0 controls on which edge the input causes an interrupt.

Bits 3, 4 and 5 of the control byte control the CA2 and CB2 pins. These pins are a little more flexible than the CA1 and CB1 pins. They can be outputs or inputs controlled by bit 5. If bit 5 (on either control byte) is high, then the pin is an output. If it is low, then it is an input. When bit 5 is low, bits 4 and 3 make these pins behave exactly like bits 1 and 0 make pins CA1 and CB1 behave. When CA2 or CB2 are initialized as outputs, they behave a little differently.

Let's look at CA2 first. There are four possible combinations of operation. The first is when bit 4 is low and bit 3 is low. This goes low after the first negative transition of the E clock after the CPU reads Port A. It returns high when the interrupt flag is set (CRA-7) by the active transition of the CA1 signal. If bit 4 is low and bit 3 is high, it is the same, but goes high after the first 'E' clock cycle. This mode is used mainly as an acknowledgment of a read (handshaking) to another peripheral. If bit 4 is high and bit 3 is low, CA2 is low. If bit 4 is high and bit 3 is high, CA2 is high. This mode is used when CA2 is to be used as a latched bit to control an external device.

Next there is CB2. There are also four possible combinations of operation for this output pin. When bit 4 is low and bit 3 is low, this pin goes low on the positive transition of the first 'E' clock as a result of a write to the B port; then goes high again when the interrupt flag bit (CRB-7) is set by an active transition of the CB1 input. When bit 4 is low and bit 3 is high it is the same, but goes high again on the positive edge of the first 'E' clock following that write. This mode is used when there is a need to autostrobe or select an exterior device. If bit 4 is high and bit 3 is low, it causes CB2 to go low and stay low. If bit 4 is high and bit 3 is high, it causes CB2 to go high and stay high. This is another latched bit to control an external device.

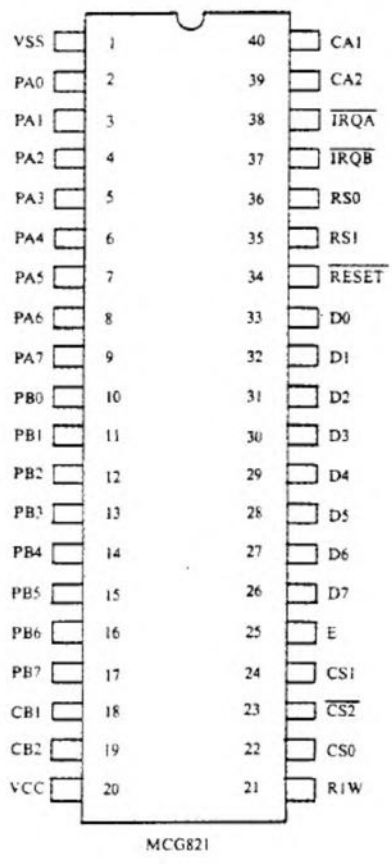


Figure 1

RS1	RS0	Register Bit CRA-2	Register Bit CRB-2	Location Selected
0	0	1	x	Peripheral Reg. A
0	0	0	x	Data Direction Reg. A
0	1	x	x	Control Reg. A
1	0	x	1	Peripheral Reg. B
1	0	x	0	Data Direction Reg. B
1	1	x	x	Control Reg. B

Table 1: Internal Addressing

BITS	7	6	5	4	3	2	1	0
CRA	IRQ A1	IRQ A2	CA2 Control			DDR A	CA1 Control	
CRB	IRQ B1	IRQ B2	CB2 Control			DDR B	CA2 Control	

Table 2: Control Registers

Transfer CoCo Text Files to MS-DOS Disks

By Marty Goodman

In the last installment of this series I discussed transferring text files from a single-sided MS-DOS disk to the CoCo. In this month's article, I will discuss formatting a disk (using the CoCo) as a single-sided MS-DOS disk and transferring text files from the CoCo to that MS-DOS disk.

This converter is designed to work with Disk Extended BASIC versions 1.0 and 1.1 only. The program is not intended to work with OS-9 or other CoCo operating systems. It should work properly with most Disk Extended BASIC Enhancements. It has been tested using ADOS from Spectrosystems and functions properly under that set of enhancements.

You need a CoCo with at least two single-sided disk drives. I have required the use of two 40-track capable disk drives because with one drive the conversion would be unacceptably slow. There is no support provided for single-drive systems or double-sided drives. Old gray CoCo drives from Tandy probably won't work with this program because most of them can access only 35 tracks. Almost all disk drives sold in the last two years by Tandy and by third party sources are 40-track capable and should work fine.

Note that this program package is more useful than the one I presented before, in the sense that any MS-DOS machine can read files from a single-sided MS-DOS disk. So, when transferring files from CoCo Disk BASIC format to MS-DOS machines, the fact that a CoCo (typically) has only single-sided drives available is not as much of a disadvantage as it was in the case of file transfer in the other direction.

This package consists of the following programs:

MS19GEN.BAS — This BASIC program creates the machine language file *MS19SET.BIN*.

MSFORMAT.BAS — BASIC driver program for MS19SET.BIN.

ADDLF.BAS — BASIC utilty for adding line feeds to CoCo text files. This will be needed in most cases to condition CoCo files prior to transferring them.

COCO2MS.BAS — main CoCo to MS-DOS converter

All of these programs are presented in this article, are available on RAINBOW ON TAPE, and are downloadable from the Delphi CoCo SIG from the RAINBOW ON TAPE area. Source code for *MS19SET.BIN* is posted in the free download area of Delphi's CoCo SIG under the filename MS19SET.SRC.

Formatting

To transfer CoCo files to a disk that will be read by an IBM PC or other MS-DOS machine, we must have a disk formatted in the way MS-DOS expects: 40 tracks, nine 512-byte-long sectors per track and nine sectors per track. One can make up such a disk on an MS-DOS machine by using the command `FORMAT B: /1`.

What if you want to move files to an MS-DOS disk and all you have is the CoCo? It happens that the CoCo is capable of formatting a disk in MS-DOS fashion (single-sided only, of course, assuming single-sided drives). My little utility *MS19SET.BIN*, combined with the BASIC driver program *MSFORMAT.BAS* allows you to do this. First type in and save to disk *MS19GEN.BAS*. Then run the program. It creates and saves to disk the needed machine language program *MS19SET.BIN*.

After creating the program *MS19SET*, be sure to have it on the same disk with *MSFORMAT.BAS*. Now run *MSFORMAT.BAS*. You are instructed to put a blank, unformatted disk into Drive 1 and press ENTER. The disk will be formatted in MS-DOS fashion. Some minimal amounts of specific data entered into key sectors in the FAT (File Allocation Table) and Directory sectors will make the disk a proper blank MS-DOS type disk. Never simply `LOADM` and `EXEC MS19SET.BIN`. If you do, it will attempt to format the disk in Drive 0.

Technical Tip

There is an idiosyncrasy in the NEC disk controller chip used on virtually all IBM PCs and clones. Unlike the Western Digital disk controller chip used in the CoCo and Model IV, the NEC chip performs an automatic hardware reset when it sees the index sector hole. So when programming a Western Digitial chip to format a disk to be read by an NEC controller chip, you should be careful to make the initial gap between the index sector hole and the first sector header somewhat bigger than you would if writing a format track for the Western Digital chip itself. If you don't, the machine trying to read the disk with the NEC chip may sometimes fail to see the first sector on a given track because the NEC chip is still recovering from its automatic reset.

Full source code (in Macro 80C format) for *MS19SET.BIN* is available in the free download section of the Delphi CoCo SIG, in the assembly language topic area. Feel free to examine it and compare how I did the format routine there to how Microsoft did the format code for the Color Computer's `DSKINI` command. A complete disassembly of the Disk BASIC `DSKINI` command can be found in *Spectral Associates Disk Basic Decoded*, available from both Spectral Associates and from Spectrum Projects.

Note also that if you are trying to read an ordinary CoCo format disk on an IBM PC or other MS-DOS machine with an NEC controller chip and are having problems getting the IBM PC to read the first sector on the disk, try putting a piece of tape over the index sector hole on the disk. The index sector hole is not used in ordinary sector reads, so covering it will not hurt the ability

of the controller to read the disk.

The Line Feed Problem

Although my converter program faithfully transfers every byte in a text (or other) CoCo file to an MS-DOS format disk in proper MS-DOS file format, this is not sufficient for practical file transfer of ASCII text. In the last installment we saw how MS-DOS type files come with Line Feed control characters (LFs) after each Carriage Return control character (CR). There we had to use a special program after conversion to strip out the line feeds so as not to confuse the CoCo word processors that not only don't use the line feeds, but also choke on them when encountered in a file.

For conversion in the reverse direction, we must (in nearly all cases) add an LF character to the CoCo Text File after each Carriage Return in the text. If we fail to do this, the converted file is unreadable to most MS-DOS based word processors. So, before moving a CoCo text file to an MS-DOS disk, convert it to a file with added LFs using *ADDLF.BAS*. This program is a clever hybrid of BASIC Disk I/O and ML file conditioning written by Art Flexser and enhanced by Don Hutchison. Both of these folks can be found on the Delphi SIG (user names ARTFLEXSER and DONHUTCHISON). It is included in this article by the kind permission of its author.

Simply run *ADDLF.BAS*. It prompts for the filename of the file to be conditioned and then for the filename of the conditioned file that will be written. Then it adds LFs after each CR it finds, and rewrites the file to the new filename specified.

File Conversion

You are finally ready to use *COCO2MS.BAS*, the main file converter program. This program is written in BASIC and is commented. The program is essentially self-prompting and menu driven. Simply LOAD and run the program. Put a CoCo Disk BASIC type disk in Drive 0 and a single-sided MS-DOS format disk in Drive 1. Press ENTER and the program confirms that a proper type MS-DOS disk is indeed in Drive 1. It reads the directory of the CoCo disk in Drive 0, and puts the directory entries on the screen as a menu for you to select the filename you want to convert. Note that a completely full CoCo disk can have up to 68 separate files on it, so if there are more files, the program reminds you there are more to be seen if you press the appropriate arrow keys.

A note here on user interface design: This arrangement of file selection is inferior to a true point and pick selection routine, where all filenames are scrolled past a pointer and you press the firebutton or ENTER key to select a file. But it is far superior to blindly typing in a filename and being expected to get the spelling and syntax precisely right.

Select a file by pressing the letter associated with the filename. The screen clears and you are asked to confirm the choice. Press ENTER again and file conversion begins. The time needed depends on the size of the file, but goes fairly rapidly (roughly 2400 Baud). During conversion, the screen displays the number of bytes still to be converted. After the conversion is completed, you are asked for a filename to be assigned to the file as it exists on the MS-DOS disk. You can then exit the program or restart it.

Note that all files converted to the MS-DOS disk will be written to the root directory. This program is incapable of creating or writing to subdirectories on the MS-DOS disk. The program also adds some trash to the end of a file after converting it. Usually this trash is from the file itself. So don't be alarmed if at first glance you see what appears to be a file with a few of its last lines cut off. Most likely, if you look back you will see the true end of the file. This minor flaw is due to my lazy decision to write MS-DOS files in even 512-byte-sector size increments.

Hacker Tips

Hackers will note that I have successfully used the *MS2 COCO* program (presented in the previous installment of this series) to transfer CoCo machine language programs and CoCo tokenized BASIC programs I downloaded from Delphi over to my CoCo Disk BASIC disks using my IBM PC clone. I deleted statement 5050 from *MS2COCO* to insure a pure byte for byte transfer, then transferred the files to the CoCo Disk. I loaded the .TXT files created into *MIKEYTERM 4.0* and saved them as binary ML program or Compressed (tokenized) BASIC, whichever was appropriate to the given file. With compressed BASIC programs, I'd often get a DS Error when loading a file, but the file *had* loaded fully and properly. Saving the result back to disk cleaned up the file so that I had a working tokenized BASIC file.

If you do not have *Mikeyterm*, condition the file by using a disk zapper program to edit the two flag bytes in the directory entry that tell Disk BASIC what kind of file it is going to deal with. *Mikeyterm* is available in the free download section of Delphi's CoCo SIG Data Communications database. It is the terminal program I use when I am telecommunicating with my CoCo.

Listing 1: MS19GEN

```
5 GOTO 5ØØØØ
1Ø NAM$="MS19SET.BIN" 'FILENAME
2Ø S=&H3ØØØ 'START ADR
3Ø T=&H3242 'ENDADR
4Ø B=&H3ØØØ 'EXEC ADR
7Ø CLS:PRINT@256-32,"NOW CREATIN
G THE FILE"
75 PRINT"I'LL COUNT BACKWARDS."
8Ø PRINT"WHEN I HIT ZERO I'LL BE
 DONE."
9Ø X=Ø
1ØØ CKSM=Ø
1Ø5 FOR C=Ø TO 31
11Ø READ H$
12Ø IF H$="END" THEN GOTO 9ØØ
13Ø D=VAL("&H"+H$)
14Ø POKE S+X+C,D:CKSM=CKSM+D
145 BB=T-(X+S+C)
146 IF BB<Ø THEN GOTO 15Ø
147 PRINT@334,BB
15Ø NEXT C
16Ø READ DSUM
17Ø IF DSUM<>CKSM THEN GOTO 5ØØ
18Ø X=X+32:GOTO 1ØØ
5ØØ K=INT(X/32)
51Ø CLS:SOUND 1ØØ,1Ø
52Ø PRINT@256,"ERROR IN LINE ";1
ØØØ+K
53Ø END
9ØØ CLS:PRINT"PUT A DISK IN DRIV
E Ø"
91Ø PRINT"TAP ENTER TO SAVE"
92Ø PRINT"     ";NAM$
93Ø PRINT"TO YOUR DISK."
94Ø A$=INKEY$
95Ø IF A$<>CHR$(13) THEN GOTO 94
Ø
96Ø SAVEM NAM$,S,T,B
98Ø PRINT:PRINT:PRINT"      ALL D
ONE!"
99Ø END
1ØØØ DATA BE,CØ,6,6F,84,AD,9F,CØ
,4,6D,6,1Ø,26,1,9Ø,17,1,9A,BE,CØ
,6,7F,9,85,6F,2,86,CØ,B7,FF,48,1
7, 328Ø
1ØØ1 DATA 1,5E,1Ø,26,1,79,BE,CØ,
6,2Ø,1C,81,16,25,8,B6,9,86,8A,1Ø
,B7,FF,4Ø,86,53,B7,FF,48,1E,88,1
E,88, 2955
1ØØ2 DATA BD,31,8Ø,1Ø,26,1,58,A6
,2,17,1,1B,BD,31,78,1Ø,8E,FF,4B,
1A,5Ø,8E,3Ø,79,BF,9,83,8E,4Ø,Ø,B
6,FF, 2965
1ØØ3 DATA 48,86,FF,B7,9,82,C6,F4
,F7,FF,48,B6,9,86,8A,8Ø,B7,FF,4Ø
,E6,8Ø,E7,A4,2Ø,FA,BE,CØ,6,B6,FF
,48,1C, 4852
1ØØ4 DATA AF,84,44,1Ø,26,1,18,6C
,2,A6,2,B1,31,A7,1Ø,25,FF,99,8E,
4Ø,Ø,CC,FC,FF,ED,81,CC,FF,Ø,ED,8
1,CC, 3898
1ØØ5 DATA Ø,Ø,ED,81,8C,42,Ø,25,F
9,BE,CØ,6,86,3,A7,84,CC,4Ø,Ø,ED,
4,CC,Ø,2,ED,2,AD,9F,CØ,4,6D,6, 3
279
1ØØ6 DATA 1Ø,26,Ø,DB,86,4,A7,3,A
D,9F,CØ,4,6D,6,1Ø,26,Ø,CD,CC,Ø,Ø
,FD,4Ø,Ø,FD,4Ø,2,FD,4Ø,4,86,3, 2
781
1ØØ7 DATA A7,3,AD,9F,CØ,4,6D,6,1
Ø,26,Ø,B3,86,5,A7,3,AD,9F,CØ,4,6
D,6,1Ø,26,Ø,A5,8E,4Ø,Ø,CC,F6,F6,
 3119
1ØØ8 DATA ED,81,8C,42,Ø,25,F9,8E
,4Ø,Ø,86,E5,A7,84,3Ø,88,2Ø,8C,42
,Ø,25,F6,BE,CØ,6,86,3,A7,84,86,6
,A7, 3562
1ØØ9 DATA 3,AD,9F,CØ,4,E6,6,1Ø,2
6,Ø,74,6C,3,A6,3,81,A,25,EC,BE,C
```

```
Ø,6,CC,4Ø,Ø,ED,4,86,2,A7,84,86,
3Ø95
1Ø1Ø DATA Ø,A7,2,C6,1,E7,3,AD,9F
,CØ,4,6D,6,1Ø,26,Ø,4E,6C,3,E6,3,
C1,A,25,EC,6C,2,A6,2,81,28,25, 2
681
1Ø11 DATA EØ,7F,FF,4Ø,F,FØ,39,8E
,41,E,C6,A,5A,27,8,A7,84,3Ø,89,2
,8Ø,2Ø,F5,39,8E,22,2E,3Ø,1F,26,F
C,39, 3144
1Ø12 DATA 1Ø,8E,Ø,Ø,31,3F,27,8,B
6,FF,48,85,1,26,F5,39,86,DØ,B7,F
F,48,1E,88,1E,88,B6,FF,48,86,8Ø,
39,7F, 3439
1Ø13 DATA FF,4Ø,86,55,97,FØ,39,2
8,7F,FF,4Ø,39,8E,4Ø,Ø,CE,32,B,1Ø
,8E,Ø,5,8D,39,31,3F,1Ø,8C,Ø,Ø,26
,F6, 2915
1Ø14 DATA 4F,B7,32,42,86,1,B7,32
,41,CE,32,15,1Ø,8E,Ø,12,8D,1F,31
,3F,1Ø,8C,Ø,Ø,26,F6,B6,32,41,4C,
B7,32, 2594
1Ø15 DATA 41,81,A,25,E4,CE,32,39
,8D,7,8D,5,8D,3,8D,1,39,EC,C1,C1
,BB,27,A,C1,CC,27,B,E7,8Ø,4A,26,
FB, 3446
1Ø16 DATA 39,F6,32,42,2Ø,F5,F6,3
2,41,2Ø,FØ,96,4E,14,Ø,3,F6,1,FC,
5Ø,4E,C,Ø,3,F5,1,FE,1,AA,1,BB,1,
 3112
1Ø17 DATA CC,1,2,1,F7,16,4E,C,Ø,
3,F5,1,FB,8Ø,FF,8Ø,FF,8Ø,FF,8Ø,F
F,1,F7,44,4E,FF,4E,FF,4E,FF,4E,F
F, 4247
1Ø18 DATA 4E,Ø,Ø,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF, 7473
1Ø19 DATA END
5ØØØØ PCLEAR 1:GOTO 1Ø
```

Listing 2: MSFORMAT

```
5 LOADM "MS19SET"
1Ø CLS:PRINT"PUT BLANK DISK IN D
RIVE 1"
2Ø PRINT"HIT ENTER WHEN READY"
3Ø IF INKEY$="" THEN GOTO3Ø
5Ø POKE &HEB,1:EXEC
6Ø IF PEEK(&HFØ)<>Ø THEN GOTO 1Ø
Ø
7Ø CLS:PRINT"YOUR DISK IS FORMAT
TED "
8Ø PRINT"AS AN MS DOS SINGLE SID
ED DISK"
9Ø GOTO11Ø
1ØØ CLS:PRINT"ERROR FORMATTING T
HE DIS"
11Ø PRINT"TAP ENTER TO RERUN PRO
GRAM"
12Ø A$=INKEY$
13Ø IF A$<>CHR$(13) THEN GOTO 12
Ø
14Ø GOTO 1Ø
```

Listing 3: ADDLF

```
1Ø 'LINEFEED ADDER
2Ø 'ART FLEXSER, MARCH 1986
3Ø '
4Ø 'MODIFIES ASCII FILES BY ADDI
NG A LINEFEED AFTER EACH
5Ø 'CARRIAGE RETURN IF ONE IS NO
T ALREADY PRESENT.
6Ø '
7Ø CLEAR2ØØ,&H7EØØ
8Ø FORI=&H7EØØ TO &H7E37:READ P$
:POKEI,VAL("&H"+P$):NEXT
9Ø DATA 8D,18,8D,2B,8D,1B,81,ØD,
26,F6,8D,ØE,81,ØA,27,F4,34,Ø2
1ØØ DATA 86,ØA,8D,ØB,35,Ø2,2Ø,E8
,C6,Ø1,D7,6F,7E,C5
11Ø DATA 97,C6,Ø2,D7,6F,AD,9F,AØ
,Ø2,ØF,6F,6E,9F,AØ,Ø2,ØD,7Ø,27
12Ø DATA Ø4,ØF,6F,32,62,39
13Ø IF PEEK(&HCØØ4)<>&HD6 THEN P
OKE&H7E2Ø,&HC4
14Ø CLS:PRINT:PRINT:PRINTTAB(8)"
LINEFEED ADDER":PRINT
15Ø LINEINPUT"NAME OF INPUT FILE
: ";I$
16Ø LINEINPUT"NAME OF OUTPUT FIL
E: ";O$
17Ø OPEN "I",#1,I$
18Ø OPEN "O",#2,O$
19Ø EXEC&H7EØØ
2ØØ CLOSE
21Ø END
```

Listing 4: COCO2MS

```
5 CLEAR 4ØØØ, &H5DFF
1Ø DIM NAM$(64):DIM SG(64)
2Ø H=PEEK(&HCØØ4):L=PEEK(&HCØØ5)
:DKON=H*256+L
3Ø CQ=Ø:MQ=1 'SET DRIVE # FOR CO
CO AND FOR MS DOS DISK
1ØØ CLS:PRINT@32,"   COLOR COMPU
TER TO MSDOS"
1Ø5 PRINT"   FILE CONVERSION UTI
LITY"
11Ø PRINT:PRINT" (C) JAN 1986 MA
RTY GOODMAN"
115 PRINT:PRINT:PRINT"  PUT COCO
 DISK IN DRIVE Ø"
12Ø PRINT:PRINT"  PUT MS DOS DIS
K IN DRIVE 1"
125 PRINT:PRINT"(MS DOS DISK MUS
T BE FORMATTED"
13Ø PRINT"AS ONE SIDED 9 SECTOR
PER TRACK)"
14Ø PRINT:PRINT"  *** TAP ENTER
TO PROCEED ***"
15Ø IF INKEY$<>CHR$(13) THEN GOT
O 15Ø
2ØØ REM READ IN COCO GAT AND MS
 DOS FAT, AND READ COCO DIRECTOR
Y
21Ø CLS:PRINT:PRINT"NOW SETTING
UP FOR FILE TRANSFER"
215 PRINT:PRINT"THIS WILL TAKE A
BOUT 1Ø SECONDS"
22Ø PRINT:PRINT:PRINT:PRINT"
      PLEASE WAIT"
23Ø POKE &HEA,2:POKE &HEB,CQ:POK
E &HEC,17:POKE &HED,2:POKE &HEE,
&H5F:POKE &HEF,Ø
235 EXEC DKON
24Ø IF PEEK(&HFØ)<>Ø GOTO 91ØØ
25Ø FOR N=2 TO 3
255 POKE &HEB,MQ:POKE &HEC,Ø:POK
E &HED,N:POKE &HEE,&H72+(N-2)*2
257 EXEC DKON
26Ø IF PEEK(&HFØ)<>Ø THEN GOTO 9
ØØØ
265 NEXT N
27Ø GOSUB 3ØØØØ
3ØØ REM DISPLAY COCO DIRECTORY
AND SELECT A FILE TO CONVERT
31Ø PAGE=1
315 IF LE=Ø GOTO 92ØØ
32Ø REM LOOP
325 CLS
33Ø B=(PAGE-1)*22+1
335 F=B+21
34Ø IF LE<F THEN F=LE
35Ø J=Ø
355 PRINT@INT(J/2)*32,CHR$(J+65)
36Ø PRINT@INT(J/2)*32+2,NAM$(J+B
)
365 J=J+1:IF B+J>F GOTO 395
37Ø PRINT@INT(J/2)*32+16,CHR$(65
+J)
375 PRINT@INT(J/2)*32+18,NAM$(J+
B)
38Ø J=J+1:IF B+J>F GOTO 395
39Ø GOTO 355
395 GOSUB 1ØØØØ
4ØØ A$=INKEY$
4Ø2 IF A$=CHR$(32) THEN GOTO 1ØØ
4Ø5 IF A$="" GOTO 4ØØ
41Ø IF PAGE=2 THEN GOTO 475
415 IF PAGE=1 GOTO 45Ø
42Ø IF A$=CHR$(1Ø) THEN PAGE= 2:
GOTO 32Ø
43Ø GOTO 5ØØ
45Ø IF LE<22 GOTO 5ØØ
455 IF A$=CHR$(94) THEN PAGE=2:G
OTO 32Ø
46Ø GOTO 5ØØ
475 IF LE<44 THEN GOTO 48Ø
477 IF A$=CHR$(94) THEN PAGE =3:
GOTO 32Ø
48Ø IF A$=CHR$(1Ø) THEN PAGE=1:G
OTO 32Ø
5ØØ A=ASC(A$)
51Ø A=A-65+B
515 IF A<Ø THEN GOTO 4ØØ
52Ø IF A>F GOTO 4ØØ
525 CLS
53Ø PRINT@32,"YOU HAVE CHOSEN: "
535 PRINT:PRINT"     ";:PRINTNAM
$(A):PRINT
54Ø PRINT:PRINT"IF THIS IS CORRE
CT, TAP ENTER"
545 PRINT"IF NOT, TAP ANY OTHER
KEY"
55Ø A$=INKEY$
555 IF A$="" GOTO 55Ø
56Ø IF A$<>CHR$(13) THEN GOTO 32
Ø
57Ø GOTO 1ØØØ
9ØØ END
1ØØØ REM MAIN FILE CONVERSION LO
OP
1ØØ5 CLS:PRINT@256," BYTES TRANS
FERRED: "
1Ø1Ø FZ=Ø
1Ø15 YY=Ø
1Ø2Ø G=SG(A)
1Ø25 ZQ=33
1Ø3Ø REM LOOP
1Ø35 EF=Ø
1Ø4Ø GOSUB 4ØØØØ
1Ø45 U=9
1Ø5Ø IF G<7Ø GOTO 11ØØ
1Ø6Ø U=G-1ØØ
11ØØ FOR J=1 TO U
111Ø L=&H6ØØØ+(J-1)*512
112Ø FZ=FZ+512
1121 PRINT@256+19,FZ
1122 IF G<68 GOTO 113Ø
1125 IF J=U THEN EF=255
113Ø GOSUB 25ØØØ
1135 IF YY=Ø THEN WW=OC
1137 IF YY=Ø THEN YY=255
114Ø IF CXN=&HAAA THEN GOTO 93ØØ
115Ø NEXT J
116Ø IF EF=255 THEN GOTO 2ØØØ
117Ø GOTO 1Ø3Ø
2ØØØ CLS
2Ø1Ø PRINT@32,"TYPE IN A FILE NA
ME FOR THE"
2Ø2Ø PRINT"FILE ON THE MS DOS DI
SK"
2Ø3Ø PRINT"USE UP TO 8 LETTERS"
2Ø4Ø PRINT:INPUT A$
2Ø5Ø IF LEN(A$)=Ø THEN GOTO 2Ø1Ø
2Ø6Ø B=LEN(A$)
2Ø65 IF B=Ø THEN GOTO 2ØØØ
2Ø7Ø IF B>8 THEN GOTO 2ØØØ
2Ø75 IF B=8 GOTO 2Ø95
2Ø8Ø A$=A$+STRING$(8-B," ")
2Ø95 A$=A$+"COL"
21ØØ REM WRITE DIR ENTRY  TO MS
DOS DISK
21Ø5 GOSUB 2ØØØØ
211Ø IF A$="" THEN GOTO 93ØØ
212Ø POKE &HEA,3:POKE &HEB,MQ
213Ø REM WRITE TWO COPIES OF UPD
ATED FAT TO MS DOS DISK
214Ø FOR N=2 TO 5
2145 LO=&H72+(N-2*INT(N/2))*2
```

```
215Ø POKE &HED,N:POKE &HEE,LO
216Ø EXEC DKON
217Ø NEXT N
218Ø GOTO 3ØØ
9ØØØ CLS:PRINT"BAD MS DOS DISK":
END
91ØØ CLS:PRINT"BAD COCO DISK":EN
D
92ØØ CLS:PRINT" COCO DISK IS BLA
NK"
921Ø PRINT" PUT ANOTHER ONE IN D
RIVE Ø"
922Ø PRINT" AND TAP ENTER"
923Ø IF INKEY$<>CHR$(13) THEN GO
TO 923Ø
924Ø GOTO 1ØØ
93ØØ CLS:PRINT"MS DOS DISK IS FU
LL"
931Ø PRINT"PUT ANOTHER MS DOS DI
SK IN"
932Ø PRINT"DRIVE 1 AND TAP ENTER
"
933Ø IF INKEY="" THEN GOTO 933Ø
934Ø GOTO 1ØØ
1ØØØØ PRINT@12*32,"SELECT FILE B
Y TAPING A LETTER"
1ØØ1Ø IF LE<22 THEN GOTO 1ØØ5Ø

1ØØ2Ø IF PAGE=2 THEN GOTO 1Ø1ØØ
1ØØ3Ø IF PAGE=3 THEN GOTO 1Ø2ØØ
1ØØ4Ø PRINT"TAP UP ARROW TO SEE
MORE ENTRIES";
1ØØ5Ø PRINT"TAP SPACE BAR TO RES
TART PGM";
1ØØ6Ø RETURN
1Ø1ØØ IF LE>44 GOTO 1Ø15Ø
1Ø11Ø PRINT"TAP DN ARROW TO SEE
PREV ENTRIES";
1Ø12Ø GOTO 1ØØ5Ø
1Ø15Ø PRINT"TAP UP ARROW TO SEE
MORE ENTRIES";
1Ø16Ø PRINT"TAP DN ARROW TO SEE
PREV ENTRIES";
1Ø17Ø GOTO 1ØØ5Ø
1Ø2ØØ PRINT"TAP DN ARROW TO SEE
PREV ENTRIES";
1Ø21Ø GOTO 1ØØ5Ø
15ØØØ REM READ FAT
15ØØ1 REM GN =CLUSTER ENTRY#
15ØØ2 REM GN=Ø TO 44Ø
15ØØ3 REM CV = CONTENTS OF THE
CLUSTER NUMBER REQUESTED
15ØØ5 FB=&H72ØØ
15Ø1Ø GIN=INT(GN/2)
15Ø2Ø GCN=3*GIN
15Ø3Ø GF=GN-2*GIN
15Ø4Ø B1=PEEK(FB+GCN)
15Ø5Ø B2=PEEK(FB+GCN+1)
15Ø55 B3=PEEK(FB+GCN+2)
15Ø6Ø N1=(B1 AND &HFØ)/16
15Ø7Ø N3=(B2 AND &HFØ)/16
15Ø8Ø N5=(B3 AND &HFØ)/16
15Ø9Ø N2=B1 AND &HF
151ØØ N4=B2 AND &HF
1511Ø N6=B3 AND &HF
1512Ø IF GF=Ø GOTO 152ØØ
1515Ø CV=N3+N6*16+N5*256:RETURN
152ØØ CV=N2+N1*16+N4*256:RETURN
17ØØØ REM WRITE A CLUSTER VALUE
 INTO A GIVEN CLUSTER NUMBER
17ØØ1 REM GN=CLUSTER NUMBER, CV
IS THE 12 BIT VALUE TO BE PUT TH
ERE
17ØØ3 REM THIS IS ALL DONE TO FA
T BUFFER AREA.
17Ø1Ø N1=INT(CV/256):T=CV-256*N1
17Ø2Ø N2=INT (T/16)
17Ø3Ø N3=T-16*N2
17Ø5Ø GIN=INT(GN/2)
17Ø6Ø GCN=3*GIN
17Ø7Ø GF=GN-2*GIN
17Ø75 GOSUB 173ØØ
17Ø8Ø IF GF=Ø GOTO 172ØØ
171ØØ REM WRITE TO ODD CLUSTER E
NTRY NUMBER
1711Ø B2=B2 AND &HF:B2=B2+16*N3
1712Ø B3=16*N1+N2
1713Ø POKE &H72ØØ+GCN+1,B2
1714Ø POKE &H72ØØ+GCN+2,B3
1715Ø RETURN
172ØØ REM WRITE TO EVEN CLUSTER
ENTRY NUMBER
1721Ø B1=16*N2+N3
1722Ø B2=B2 AND &HFØ:B2=B2+N1
1723Ø POKE &H72ØØ+GCN,B1
1724Ø POKE &H72ØØ+GCN+1,B2
1725Ø RETURN
173ØØ B1=PEEK(&H72ØØ+GCN)
1731Ø B2=PEEK(&H72ØØ+GCN+1)
1732Ø B3=PEEK(&H72ØØ+GCN+2)
1734Ø RETURN
2ØØØØ REM FIND FREE MSDOS DIR EN
TRY
2ØØØ1 REM WRITE A$ INTO THAT ENT
RY, WITH ATTRIBUTE BYTE SET TO 2
2ØØØ2 REM LEAVE FILE SIZE BLANK
2ØØØ3 REM THIS IS EQUIV OF AN
'OPEN' STATEMENT
2ØØØ4 REM IF A$="" ON RETURN THE
 DIR SPACE IS FULL
2ØØØ5 REM A$ IS A STRING 11 BYTE
S LONG
2ØØØ6 REM WITH FILE NAME AND EXT
ENTION
2ØØØ7 REM WW=# OF FIRST CLUSTER
ENTRY
2ØØ1Ø A$=A$+CHR$(&H2Ø)+STRING$(1
4,CHR$(Ø))
2ØØ2Ø X=WW:GOSUB 45ØØØ
2ØØ3Ø G1$=CHR$(L):G2$=CHR$(H)
2ØØ35 GOSUB 46ØØØ
2ØØ4Ø A$=A$+G1$+G2$+CHR$(D1)+CHR
$(D2)+CHR$(D3)+CHR$(Ø)
2ØØ45 GOSUB 2Ø1ØØ
2ØØ5Ø IF A$="" THEN RETURN
2ØØ6Ø FOR Q=Ø TO 31
2ØØ62 POKE MDB+K*32+Q,ASC(MID$(A
$,Q+1,1))
2ØØ64 NEXT Q
2ØØ7Ø POKE &HEA,3:POKE &HEB,MQ
2ØØ75 EXEC DKON
2ØØ8Ø RETURN
2Ø1ØØ FOR N=Ø TO 3
2Ø11Ø MDB=&H76ØØ:MSB=&H76
2Ø12Ø POKE &HEA,2:POKE &HEB,MQ:P
OKE &HEC,Ø:POKE &HED,6+N:POKE &H
EE,MSB:POKE &HEF,Ø
2Ø13Ø EXEC DKON
2Ø135 IF PEEK(&HFØ)<>Ø THEN GOTO
 91ØØ
2Ø14Ø FOR K=Ø TO 15
2Ø15Ø FB=PEEK(MDB+32*K)
2Ø16Ø IF FB=&HE5 THEN RETURN
2Ø17Ø IF FB=Ø THEN RETURN
2Ø18Ø NEXT K
2Ø19Ø NEXT N
2Ø195 A$="":RETURN
25ØØØ REM FIND FREE CLUSTER
25ØØ1 REM WRITE SECTOR AT LOC L
          TO THAT CLUSTER
25ØØ2 REM FIND NEXT FREE SECTOR
25ØØ3 REM UPDATE FAT
25ØØ4 REM RETURN WITH NEXT FREE
CLUSTER
25ØØ5 REM CXN=ENTRY # OF NEXT FR
EE CLUSTER
25ØØ6 REM IF NO MORE FREE CLUSTE
RS CXN SET = &HAAA
25ØØ7 REM IF EF=255 THEN DON'T B
OTHER TO FIND NEXT CLUSTER
25ØØ8 REM AND WRITE END CLUSTER
FLAD INTO FAT
25Ø1Ø Q=INT(L/256):QL=L-256*Q
25Ø4Ø IF ZQ=33 THEN CXN=2:ZQ=Ø
25Ø5Ø FOR GN=CXN TO 352
25Ø6Ø GOSUB 15ØØØ
25Ø7Ø IF CV=Ø GOTO 251ØØ
25Ø75 IF CV=&HFF7 GOTO 251ØØ
25Ø8Ø NEXT GN
25Ø9Ø CXN=&HAAA:RETURN
251ØØ REM FIND NEXT FREE GRAN
2511Ø OC=GN
2511Ø IF EF=255 THEN GOTO 252ØØ
2512Ø FOR GN=OC+1 TO 352
2513Ø GOSUB 15ØØØ
2514Ø IF CV=Ø THEN GOTO 252ØØ
2515Ø IF CV=&HFF7 THEN GOTO 252Ø
Ø
2516Ø NEXT GN
2517Ø CXN=&HAAA:RETURN
252ØØ CXN=GN
2521Ø GN=OC:CV=CXN
25215 IF EF=255 THEN CV=&HFFF
2522Ø GOSUB 17ØØØ 'WRITE GAT ENT
RY TO BUFFER
2523Ø REM CALC TRACK AND SECTOR
FROM OC
2524Ø T=INT((OC-2)/9)+1:S=OC+7-9
*T+1
2525Ø POKE &HEA,3:POKE &HEB,MQ:P
OKE &HEC,T:POKE &HED,S:POKE &HEE
,Q:POKE &HEF,QL
2526Ø EXEC DKON
25265 POKE &HFF4Ø,8
2527Ø RETURN
3ØØØØ REM READ COCO DIRECTORY
3ØØØ1 REM NAM$(N)=ENTRY
3ØØØ3 REM SG(N) = FIRST GRANULE
3ØØØ4 REM ALL FILE TYPES WILL BE
 CONVERTED
3ØØØ5 REM LE=NUMBER OF LAST ENTR
Y
3ØØ1Ø K=-1:N=1:LE=Ø
3ØØ2Ø REM LOOP
3ØØ3Ø K=K+1
3ØØ4Ø S=INT(K/8)+3
3ØØ45 KS=K-8*(INT(K/8))
3ØØ5Ø DSKI$ Ø,17,S,A$,B$
3ØØ6Ø C$=A$
3ØØ7Ø IF KS>3 THEN C$=B$
3ØØ8Ø KS=KS-(4*INT(KS/4))
3ØØ9Ø T$=MID$(C$,32*KS+1,32*(KS+
1))
3Ø1ØØ FB=ASC(LEFT$(T$,1))
3Ø11Ø IF FB=Ø THEN GOTO 3ØØ3Ø 'K
ILLED ENTRY
3Ø12Ø IF FB=255 THEN RETURN 'DON
E... EXIT ROUTINE
3Ø13Ø REM PROCESS VALID ENTRY
3Ø14Ø NAM$(N)=LEFT$(T$,8)
3Ø15Ø NAM$(N)=NAM$(N)+"."
3Ø16Ø NAM$(N)=NAM$(N)+MID$(T$,9,
3)
3Ø17Ø SG(N)=ASC(MID$(T$,14,1))
3Ø18Ø LE=N:N=N+1:GOTO 3ØØ3Ø
4ØØØØ REM INPUT 2 GRANS INTO THE
 DATA BUFFER
4ØØØ1 REM G=FIRST GRAN ON ENTRY
4ØØØ2 REM G= NEXT GRAN ON EXIT
4ØØØ3 REM IF NO MORE, G=1ØØ + NU
MBER OF VALID 512 BYTE SECTORS
4ØØØ4 REM IN THE BUFFER AREA.
4ØØ1Ø DF=Ø 'FIRST GRAN
4ØØ2Ø REM LOOP
4ØØ25 IF DF>1 THEN RETURN
4ØØ3Ø DB=&H6ØØØ:IF DF<>Ø THEN DB
=&H69ØØ
4ØØ4Ø GT=G:IFGT>33 THEN GT=GT+2
4ØØ5Ø T=INT(GT/2):Q=GT-2*T
4ØØ55 S=1:IF Q<>Ø THEN S=1Ø
4ØØ6Ø GOSUB 41ØØØ
4Ø1ØØ G=PEEK(&H5FØØ+G) 'GET NEXT
 GRAN #
4Ø11Ø DF=DF+1
4Ø12Ø IF G<69 GOTO 4ØØ2Ø
4Ø13Ø G=G-&HCØ
4Ø14Ø IF G>9 GOTO 91ØØ
4Ø15Ø IF G<1 THEN GOTO 91ØØ
4Ø16Ø G=G+1
4Ø17Ø IF DF=2 THEN G=G+9
4Ø18Ø G=INT(G/2):G=G+1ØØ
4Ø19Ø RETURN
41ØØØ REM BRING GRAN INTO BUFFER
41Ø1Ø FOR N=Ø TO 8
41Ø2Ø POKE &HEA,2:POKE &HEB,CQ:P
OKE &HEC,T:POKE &HED,S+N:POKE &H
EE,DB/256+N:POKE &HEF,Ø
41Ø3Ø EXEC DKON
41Ø35 IF PEEK(&HFØ)<>Ø THEN GOTO
 91ØØ
41Ø4Ø NEXT N
41Ø5Ø RETURN
45ØØØ H=INT(X/256):L=X-256*H:RET
URN
46ØØØ D3=INT (FZ/65536)
46Ø1Ø FZ=FZ-65536*D3
46Ø2Ø D2=INT(FZ/256)
46Ø3Ø D1=FZ-D2*256
46Ø4Ø RETURN
```

Build a lap keyboard and you'll have a . . .

Remote Control CoCo

By Marty Goodman

One of the single most crippling aspects of the present day Color Computer is the fact that its keyboard is physically fixed in the main "mother unit" box. I prefer to sit back in a reclining desk chair while typing. I cannot stand being hunched over a desk.

Over the last few years my partner and I have designed and built nearly a dozen remote Color Computer keyboard units and cables. We considered marketing such an item, but after perfecting a design for one we realized the necessary retail cost of such an item would be too high for it to succeed in the market. Only Tandy, with its in-house injection molding capabilities and mighty marketing muscle, is able to bring such an add-on item to the market. But, it has shown no signs of interest.

Making a lap keyboard for your CoCo is not hard for a hacker of even modest experience and ability. Nor is it terribly expensive. Here are some tips for those who would embark on such a project. Note that these are just tips. This is not intended as a "how to" article, but as a collection of helpful hints for someone who already has a good idea how to do the job.

You will need to run at least 15 lines between the CoCo and the keyboard. I also suggest running the Reset and Ground lines so you can put a remote Reset button on the lap keyboard. I also send the +5 volt line to the keyboard to support a power-on pilot light, and I send the "unused" extra keyboard PIA line as well. This comes to a total of 19 wires for my protocol.

Now comes the tough choice of a cable. The simplest way to go is to use ordinary rainbow ribbon cable, 20 conductor. This has the enormous advantage of being widely available and easy to work with. It does tend to tear, though it should work well if treated with care. I have used 20 conductor shielded ribbon cable for some of my remote keyboards. Such shielding does not really limit RFI, but will make your cable much more sturdy. Shielded ribbon cable is, however, much harder to work with, and if you are not familiar with how to use it I'd suggest you stay away from it.

The PIA chips that talk to the keyboard were not designed to send signals over long wires. But experience has shown that there is no problem when the keyboard is separated by less than eight feet of cable. I suggest you keep the keyboard cable under six feet, since there is a foot or two of wire inside the keyboard case and the CoCo.

I arranged for my keyboards to plug into my CoCo. The connectors I used were DB25 style. These are widely available and come in versions that can crimp on (IDC type) to ribbon cable.

The next problem is the choice of keyboard to use. One of the best choices is the CoCo II keyboard itself. Or an HJL or Mark Data keyboard. If you can find one, a discarded Model III keyboard is also an excellent choice. The Model III keyboard is wired very like the CoCo keyboard. Obtain schematics to see how few changes are needed to make it work properly. I also very much like the feel of the Model III keyboard.

Do not try to rewire other types of keyboards. Some keyboards available on the surplus market use Hall effect or capacitive switches. Neither will work on a CoCo. Even keyboards that do use ordinary spst NO type switches are not good choices.

If you are using an existing CoCo 2 type keyboard, you need a way to connect to the plastic ribbon connector. This is easy. Just order one or two of part number AJ 7504 from Radio Shack National Parts in Fort Worth, Texas. When they ask you what product it is for, say it's for the CoCo 2 Cat. No. 26-3134. This item is an inexpensive AMP connector that fits the plastic ribbon cable. Solder that connector to a bit of printed circuit board and solder one end of the 20 conductor cable to it.

Making contact with the PIA lines in the CoCo where the keyboard normally plugs in is a little more tricky. You may wish to desolder the existing connector from the CoCo board and replace it with a piece of ribbon cable directly soldered in. Then just crimp a DB 25 connector on the other end of the ribbon cable and mount it on a face plate put where the old keboard used to go. If you are using an older CoCo, you will find the connector on the CoCo mother board is a little easier to use.

Finally, you must make a cabinet for your lap keyboard. I've used a lot of different approaches. In one case I used an LMB brand keyboarder chassis box. In another I used a Model I shell as the keyboard case. By far the nicest looking case was one that my partner Leonard Haines made from plywood. Another alternative is to literally saw off the front part of your CoCo (talk about hacking) and use that. I have seen this done successfully. I suggest putting the remote Reset button in the rear of the keyboard case, and a little recessed. It's easy to reach when you want it, and hard to hit accidentally.

FORUM

by John Poxon

A fairly well known phrase due, I believe, to Robbie Burns is "The best laid plans of mice and men oft gang awry". Mine certainly did while preparing the article for this months FORTH Forum. The timetable for this month declares our topic to be single or double length numbers, or something similar. I had begun an article on that topic when John Redmond's disk based A*FORTHD was released.

At the outset I must say that the new FORTH is a pleasure to use, but my unfamiliarity with it kept leading me into a certain difficulty which obstructed completion of the article. The difficulties persist, I'm convinced, due to some bad technique on my part. To put it in John Redmond's words, I'm "bashing the FORTH code" somehow. It waits on the CoCoConf and conversation with John to resolve my mistakes. I'll tell you more about them in due course.

Subsequent to my (temporary) abandonment of the new A*FORTHD I received a call from a user group attendee complaining that he was having trouble getting FORTH code typed in, etc. etc.. He requested that I spend some time explaining, blow by blow, the input of a FORTH program, adding in passing that these things were being treated too lightly in the column.

I resolved therefore to do just that in this months article. To make things more entertaining I cast about for a task to write a simple FORTH program for, and elected to create a simple prime number generator. Here it is, listed exactly as on my screen:-

```
DECIMAL
VARIABLE N
VARIABLE C
: LABEL 80 1164 C! 82 1165 C! 73
 1166 C! 77 1167 C! 69 1168 C! 8
3 1169 C! ;
: PAUSE 200 0 DO 200 0 DO LOOP
LOOP ;
: SCREEN 1024 512 63 FILL ;
: IN-N N ! ;
: 3=>? N @ 2- 0> IF 3 . CR THEN
;
: 5=>? N @ 4 - 0> IF 5 . CR THEN
;
: WRITESTOP C @ 1+ DUP C ! 14 MO
D 0= IF KEY THEN ;
: TEST N @ 1+ 4 DO I 2 MOD 0> IF
 I 3 MOD 0> IF I 5 MOD 0> IF I .
CR WRITESTOP THEN THEN THEN LOOP
;
: PRIMES? 0 C ! 0 N ! IN-N SCREE
N LABEL PAUSE PAGE 3=>? 5=>? TES
T ;
```

Type it in and use it, i.e. 100 PRIMES? will find all the prime numbers up to the number 100. PRIMES? will halt at each pageful to let you take notes. Press <ENTER> to see the next pageful.

Here is how it works. First the variables. Variable N holds the value of the upper limit of the range of numbers to be tested for prime numbers. Variable C represents the number of prime numbers which have been printed on the screen, permitting printing to be halted each pageful.

Lets work in a top down manner through the code.

PRIMES? commences by initialising the variables N and C to 0.

IN-N stores the value placed on the stack in N.

SCREEN FILLs the screen with questionmarks.

LABEL prints the name PRIMES on the screen by storing appropriate values at certain text screen memory locations. This is analogous to a series of POKEs in BASIC. Screen and label are of course merely window dressing.

PAUSE holds the title screen display long enough for the observer to read it, just like an empty BASIC FOR . . . TO . . . NEXT loop.

PAGE clears the screen by setting the cursor to the top left hand corner of the screen.

3=>? and 5=>? deal with a small weakness of this method of calculating the prime numbers. The program TESTs for divisibility by 2, 3 and 5. Any number divisible by any value other than 1 or itself cannot be a prime number. Clearly the numbers 3 and 5 are prime numbers yet are not detected by this technique, so I fudged the program a bit. I hope that all the mathematicians out there will pray for me tonight!

TEST of course does the actual TESTing for divisibility by 2, 3 or 5. Any number which is not divisible by these values without a remainder will fall through the IF . . . THENs to reach the print, CR, etc..

The limit value which may be processed by PRIMES? is about 32000. Actually it's 32767. Why? Think about it for a moment. The variable N is a signed number, so it is limited to the value 2 raised to the power 15, minus 1. You can of course alter the program to check larger numbers by using unsigned or double length numbers.

How did I get it up and running? Follow my example.

a) CLOADM A*FORTHC and EXEC it if you have A*FORTH configured for tape, otherwise do likewise (with a LOADM) for the old A*FORTHD, when loading it from disk. Note that I am not using the new disk based A*FORTHD in this article i.e. the one that LOADs screens from disk and does other wonderful things.

b) When A*FORTH has signed on, type EDIT <ENTER>.

c) Type I (for insert). The cursor will then be located at the top left hand of the screen. Type in the FORTH code, remembering to insert all spaces as shown above. Extra spaces are O.K.. Terminate each line with <ENTER>. Escape from insert mode using <BREAK>.

d) When you have finished typing the code in, you may compile it or save it to tape immediately by using the following series of keystrokes: T 1 B 2 8 8 PRIMES? <ENTER> Y 9. Code can be reloaded using 7 7 PRIMES? <ENTER>.

If you would like a full printed copy type D 2 T 1 B 2 4.

Note that in each case the keystrokes T 1 B 2 define the block to be saved or printed to be all of the text. The block markers can be positioned anywhere to mark off a block for an incomplete save or print simply by positioning the cursor, (using up/down and right/left arrows, associated with the shift keys if necessary).

e) compile the code by typing O. If an error message appears pressing <ENTER> will position the cursor at the error. Check your code. Often the error is merely a misspelling or use of a word which has not yet been defined prior to the error position. Corrections may be made by positioning the cursor at a location and either using a C, M or L to eliminate any offending Character, Word or Line, then entering Insert mode and writing any new text.

Remember that help is available using the H key.

f) Press <BREAK> to escape the editor. To run the program type (value) PRIMES?, e.g. 100 PRIMES? Don't forget that the program waits on an <ENTER> to display each pageful.

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

BARDEN'S BUFFER

Interfacing Tricks for BASIC and Assembly Language

By William Barden Jr.

One of the best ways to learn assembly language is to write short assembly language programs and use them in conjunction with BASIC programs. This is called interfacing assembly language to BASIC. One reason for doing this is that the short assembly language programs can speed up BASIC programs dramatically.

Need a fast sort of records in a disk file? Read in the records using BASIC and then go to an assembly language sort to put them in alphabetical order. Need a fast way to scroll data on the screen? Use an assembly language screen scroll program called by the main BASIC program. Combining assembly language with BASIC allows you to have the speed of assembly language with the text and formatting capability of BASIC, so you get the best of both worlds.

In this column we'll show you how to go about interfacing simple assembly language programs to BASIC so you can try your hand at combining them. We'll keep it at a beginner's level, for the most part. Among the examples are an assembly language program that counts the number of words on the screen and an assembly language program that "explodes" screen text.

Assembly vs. Machine Language

The terms machine language and assembly language are often confusing so before starting, we should straighten out exactly what is meant by the two terms.

There's only one language the 6809 microprocessor in the Color Computer understands and that's machine language. The 6809 has a built-in set of instructions to do simple things like adding two numbers, transferring a byte from memory to a register in the 6809 or the other way around and branching to a memory location (like a GOTO in BASIC). These instructions are decoded by the microprocessor from the ones and zeroes it reads in memory as it executes a machine language program. Each instruction is held in one, two, three or four bytes of memory, and are generally arranged in a long sequence of instructions that the 6809 accesses one after another. A branch can alter the order of the instruction sequence just like a GOTO or IF statement can alter the sequence in BASIC. A typical sequence of machine language numeric values is shown in Figure 1, along with the instructions represented.

The built-in instruction set is generally not used by writing down long lists of binary (or hexadecimal) values. Instead, a programmer writes down a mnemonic (memory jogging) abbreviation for the instruction, such as ADD for the machine language code of 187 (add two numbers) instruction and SUB for the machine language code of 176 (subtract two numbers) instruction. These mnemonics are used even if the programmer is hand assembling the code, simply because it's easier to write instruction names rather than numbers. In the simplest form, the mnemonic names and the operands associated with the instructions are a form of assembly language.

An assembler program is a computer program that translates mnemonic names and operands into the numeric machine language values the 6809 understands. When an assembler program is used, the assembly language may contain more bells and whistles than the hand written mnemonics written by a programmer, but it's basically the same idea. Listing 1 shows the assembly language version of the machine language code of Figure 1.

Listing 1: Assembly Language Coding

```
BNE     TSTEND      GO IF NOT LINE START
INC     WC,PCR      NEW LINE IN WORD CASE
LDY     #0          RESET "IN WORD"
```

Assemblers

The assembler we've been using in this column is the Color Disk *EDTASM* assembler (or the *EDTASM+* version for cassette). It's easy to use, inexpensive and contains enough features to make even experienced programmers happy. We can't provide a tutorial on using *EDTASM* here, but if you refer to the manual, it's not too difficult to get to the point where you can assemble a short program. In the approach we're using this month, you don't have to do much more than just get a screen listing of the program — you won't have to worry about saving the program and reloading it.

A Simple Program

The simplest program I can think of that does anything useful is a program to write the letter 'A' on the text screen. The text screen uses memory locations from 1024 decimal through 1535 to store its 16 lines of 32 characters each, a total of 512 characters. Assembly language programmers use hexadecimal notation a lot, because so many things in a computer are done in powers of 16. The "hex" equivalent of 1024 is $400, where the '$' stands for hexadec-

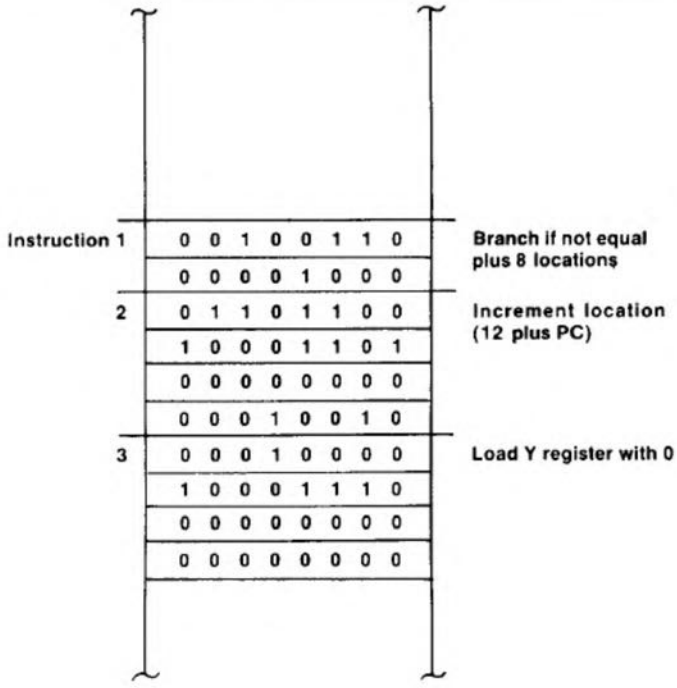


Figure 1: Assembly Language Coding

imal to follow. We'll use both decimal and hexadecimal values in the following examples to make life easier for those who don't understand hex notation.

The problem is to write the ASCII code of 65 decimal ($41), representing an 'A' to the screen. Let's shoot for the screen center at line 8 (counting lines from 0) and character position 15 (again counting from 0). The memory location is at location (1024 + 7*32 + 15) or 1263 ($4EF).

The program to do this is shown in Listing 2. It first loads the A register in the 6809 with a code of 65, then stores the contents of A to location 1263. The A register in the 6809 is a special high-speed memory location within the 6809 and not in user memory. It's used to store temporary results and as a working 6809 memory location for processing data. The 6809 has another register called B which is used for the same purpose.

Why two registers? Ideally, a microprocessor would have dozens of registers such as these so all kinds of values could be held temporarily, but the two accumulators of the 6809 represent a compromise of price and integrated circuit technology. The LDA and STA are two mnemonics for the instructions involved. The 65 and 1263 are the operands for each of the instructions.

Listing 2 is a complete program. It stores an 'A' in the screen center. But, how can we use it? One way would be to assemble the program with a CoCo assembler such as *EDTASM*, store the resulting machine code in a machine code file on disk or cassette (called an object file) and then load it in and execute it by a system LOADM and EXEC command. The LOADM loads in the machine language bytes to memory. The EXEC transfers control to the machine language instructions and the 6809 then executes them one by one.

Typically, the program would be located in an area of memory protected from BASIC, such as $3E00 (decimal 15872, just under the 16K byte area). The two instructions would be at $3E00/1 and $3E02/3/4 in this case.

If this program were executed, you would indeed see an 'A' appear in the screen center, superimposed on whatever other characters were on the screen before the program executed (the screen is not cleared by the program or by the action of executing the program). But what happens next?

After the STA 1263 is executed at locations $3E02/3/4, the 6809 attempts to execute the next machine language instruction, starting at location $3E05. However, this location was never filled with an instruction, and just about anything could be in the memory location at that point. In programming terms, this is garbage. As a result, the 6809 tries to execute whatever it finds, which, at best, would be a meaningless set of random instructions. At this point you lose control and the system locks up, looping with meaningless instructions. (At worst, it might jump to the middle of a ROM program, which could clobber disk files or do some other system shenanigans, but that's not too likely.)

Simple BASIC and Assembly Language Interfacing

What we'd really like to do is to store that 'A' from a controlled environment, like BASIC, and then come back to BASIC after the action. Is there a way to do it? I'm glad you asked . . .

BASIC has "hooks" in it to transfer control to an assembly language program (really the machine language instructions of that assembly language program, but we'll call it assembly language here) and to get back from the assembly language. One hook is the DEFUSR statement. The other is the USR statement.

The DEFUSR statement tells BASIC where the assembly language is. Suppose that we had the two-instruction program at locations $3E00 through $3E04. We could tell BASIC where the assembly language program was by using `100 DEFUSR0 = &H3E00` or `100 DEFUSR0 = 15872`.

Whenever we wanted to store the 'A' in the center of the screen, we could tell BASIC to transfer control to those instructions by the USR statement, which in essence says "transfer control to the assembly language program at the location defined by the DEFUSR statement." In this case we'd have `1000 A = USR0(0)`.

When Line 1000 is encountered by the BASIC interpreter, a transfer to location $3E00 is made and the program executes, storing the 'A' on the screen.

Returning from Assembly Language

How do we get back to BASIC? After all, normally we'd want to jump out to execute a short piece of assembly language code and then come back to BASIC to execute more BASIC instructions. The way this is done is with a 6809 instruction called RTS, Return From Subroutine. RTS is identical in function to BASIC's RETURN. It causes a return to the instruction following the one that caused the transfer.

RTS is a normal 6809 instruction used as the last instruction in all 6809 subroutines. It just happens to be a convenient instruction to use as a return to BASIC. When the 6809 executes the RTS, it goes to a special area in memory called the stack. It gets the first two data bytes from the stack and uses them as a return address. The return address in

this case represents an address in the BASIC interpreter code just after the code that caused the transfer to take place. As a matter of fact, the instruction executed to cause the transfer is a JSR-type instruction (Jump to Subroutine) which branches to the assembly language code and stores the return address in the stack.

It appears then, that we have to modify the program slightly to include an RTS instruction to get back to BASIC. Listing 3 shows the new code.

How Does the Code Get There?

Before we can execute the instructions, however, we must first guarantee the code is in memory. One way is to load the code by LOADM (or CLOADM), using the object file generated by an assembler such as *EDTASM*. Another way — the one we'll be using here — is to actually store it in memory from values in BASIC data statements. Incorporating the code in BASIC data statements means that machine language code can be included in BASIC programs, making the entire process a simple one-step process.

Let's assume we're using the $3E00 area as before. This BASIC code:

```
13Ø FOR I = &H3EØØ to &H3EØ5
14Ø READ A: POKE I,A
15Ø NEXT I
16Ø DATA &H86,&H41,&HB7,&HØ4,&HEF,&H39
```

moves the machine language values from the DATA statements into the $3E00 area just as if the data was loaded from a LOADM (CLOADM) file. Of course, the disadvantage of this method is that you wouldn't want to try it for a thousand-byte assembly language program. But for short programs, it works fine. The &H indicates hexadecimal values to BASIC. The hex values are taken directly from the program listing.

Once the code is stored in the $3E00 area, we're ready to execute . . . almost.

Protecting Memory

Actually, there's one important thing we should have done before storing the data — protect the memory area where the code is to be stored. BASIC uses a number of areas in high and low memory, as shown in Figure 2. Variables and arrays are stored as required, working up towards higher memory. Strings and the stack area (the same stack we've been talking about) are stored in high memory, with the stack working down. If the area in which we're going to store the data is not protected, storage of stack or string data results in valid instruction data being clobbered. What's to be done?

The CLEAR statement in BASIC is specifically used for this protection

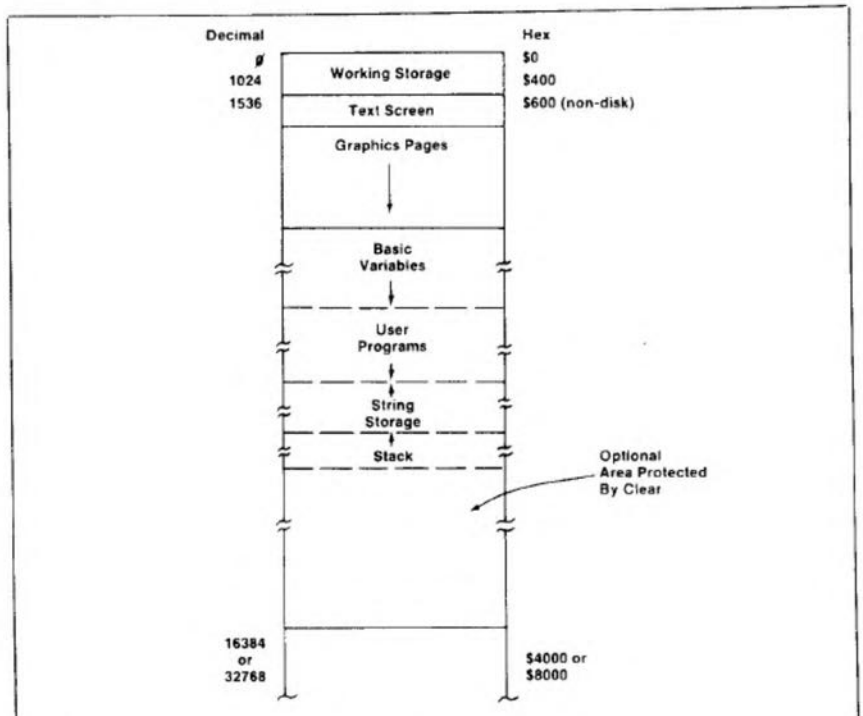


Figure 2: BASIC Memory

Listing 2: Write an 'A' Program

```
                    ØØ1ØØ * "A" STORE PROGRAM
ØØØØ 86   41        ØØ11Ø        LDA     #65      A CHARACTER
ØØØ2 B7   Ø4EF      ØØ12Ø        STA     1263     STORE IN SCREEN CENTER
          ØØØØ      ØØ13Ø        END

ØØØØØ TOTAL ERRORS
```

Listing 3: Write an 'A' Program Corrected

```
                    ØØ1ØØ * "A" STORE PROGRAM
ØØØØ 86   41        ØØ11Ø        LDA     #65      A CHARACTER
ØØØ2 B7   Ø4EF      ØØ12Ø        STA     1263     STORE IN SCREEN CENTER
ØØØ5 39             ØØ13Ø        RTS              RETURN
          ØØØØ      ØØ14Ø        END

ØØØØØ TOTAL ERRORS
```

Listing 4: WORDCNT1

```
                    ØØ1ØØ * WORD COUNT PROGRAM
ØØØØ 8E   Ø4ØØ      ØØ11Ø        LDX     #$4ØØ    PNTR TO SCREEN
ØØØ3 1Ø8E ØØØØ      ØØ12Ø        LDY     #Ø       "IN WORD" FLAG
ØØØ7 6F   8D ØØ36   ØØ13Ø        CLR     WC,PCR   ZERO WORD COUNT
                    ØØ14Ø * MAIN LOOP HERE
ØØØB A6   8Ø        ØØ15Ø LOOP   LDA     ,X+        GET NEXT CHARACTER
ØØØD 81   6Ø        ØØ16Ø        CMPA    #96        IS IT BLANK?
ØØØF 26   1Ø        ØØ17Ø        BNE     NOTBLK     GO IF NOT BLANK
                    ØØ18Ø * BLANK HERE
ØØ11 1Ø8C ØØØ1      ØØ19Ø        CMPY    #1         ARE WE IN WORD?
ØØ15 26   1C        ØØ2ØØ        BNE     TSTEND     GO IF NO
ØØ17 6C   8D ØØ26   ØØ21Ø        INC     WC,PCR     YES - BUMP WORD COUNT
ØØ1B 1Ø8E ØØØØ      ØØ22Ø        LDY     #Ø         RESET "IN WORD"
ØØ1F 2Ø   12        ØØ23Ø        BRA     TSTEND     TEST FOR END
                    ØØ24Ø * NOT BLANK HERE
ØØ21 1Ø8E ØØØ1      ØØ25Ø NOTBLK LDY     #1         SET "IN WRD"
ØØ25 1F   2Ø        ØØ26Ø        TFR     Y,D        PNTR TO D
ØØ27 C4   1F        ØØ27Ø        ANDB    #$1F       TEST FOR LINE START
ØØ29 26   Ø8        ØØ28Ø        BNE     TSTEND     GO IF NOT LINE START
ØØ2B 6C   8D ØØ12   ØØ29Ø        INC     WC,PCR     NEW LINE IN WORD CASE
ØØ2F 1Ø8E ØØØØ      ØØ3ØØ        LDY     #Ø         RESET "IN WORD"
                    ØØ31Ø * TEST FOR SCREEN END
ØØ33 8C   Ø6ØØ      ØØ32Ø TSTEND CMPX    #$6ØØ      AT END?
ØØ36 26   D3        ØØ33Ø        BNE     LOOP       LOOP IF NO
                    ØØ34Ø * SCREEN END HERE
ØØ38 E6   8D ØØØ5   ØØ35Ø        LDB     WC,PCR   WORD COUNT TO B
ØØ3C 4F             ØØ36Ø        CLRA             NOW IN A,B
ØØ3D BD   B4F4      ØØ37Ø        JSR     $B4F4    CONVERT TO INTEGER
ØØ4Ø 39             ØØ38Ø        RTS              RETURN
```

feature, along with setting the size of the string storage area. To protect the $3E00 area and provide 1,000 bytes of string storage (usually enough, depending upon the program), you'd do 110 CLEAR 1000, &H3DFF.

Note that one less than the starting address of the program was used, as the protection address specifies the last location BASIC can use. If this statement is executed once at the beginning of the program, the area from $3E00 up will never be used by BASIC.

At Long Last, An 'A' Appears

We're now ready to execute. The complete program looks like this:

```
100 ' AL DEMO PROGRAM
110 CLEAR 1000, &H3DFF      'protect $3E00 area and up
120 CLS
130 FOR I = &H3E00 to &H3E05
140 READ A: POKE I,A
150 NEXT I
160 DATA &H86,&H41,&HB7,&H04,&HEF,&H39
170 DEFUSR0 = &H3E00
180 A = USR0(0)
190 GOTO 190
```

If you execute this BASIC program, you'll see the screen clear and an 'A' appear in the screen center. Ah, the magic of assembly language. But don't scoff, this is the basis for many more impressive and useful programs.

Color BASIC Gotchas

This sequence of steps is geared to Extended Color BASIC. The sequence for Color BASIC is essentially the same, but there are some differences. Color BASIC does not have a DEFUSR, and the address of the assembly language location must be POKEd into locations 275 and 276 decimal. For $3E00, for example, this would be 140 POKE 275,62: POKE 276,0 :REM 62*256+0 = 15872.

Color BASIC also uses just a USR statement and not a USR0 (or USRn) statement to make the transfer: 150 A = USR(0).

More about the DEFUSR and USRn Statements

In Extended Color BASIC, the format of DEFUSR is DEFUSRn, where the 'n' may be a digit from zero through nine. DEFUSR0, DEFUSR1, . . . DEFUSR9 are tied in with the corresponding USRn statements which take the form USR0, USR1, . . . USR9. This means that 10 different entry points in an assembly language program may be defined, each one addressable by a different combination of DEFUSRns and USRns. Many times there is only one assembly language program involved, however. The DEFUSR and USR digit does not have to start from zero — you could have only one program with DEFUSR5 and USR5, for example. Since you can redefine the address specified by DEFUSR at any time, you can really have any number

```
0041    00      0039  WC    FCB   0       WORD COUNT 0 TO 255
        0000    0400        END

00000 TOTAL ERRORS
```

Listing 5: WORDCNT2

```
100 'WORD COUNT PROGRAM
110 CLEAR 1000,&H3DFF
120 CLS
130 FOR I=&H3E00 TO &H3E41
140 READ A: POKE I,A
150 NEXT I
160 PRINT "THEY CAN HAVE MY"
170 PRINT "COCO WHEN THEY PRY"
180 PRINT "MY COLD, DEAD"
190 PRINT "FINGERS FROM IT."
200 PRINT
210 PRINT "IN THIS SECTION"
220 PRINT "YOU'RE GOING TO"
230 PRINT "LEARN HOW TO"
240 PRINT "PROGRAM. DON'T"
250 PRINT "WORRY, IT'LL BE"
260 PRINT "PAINLESS!"
270 PRINT "(FROM A RADIO SHACK"
280 PRINT "MANUAL)"
290 PRINT
300 PRINT "    THE COCO LIVES!"
310 DATA &H8E,&H04,&H00,&H10
320 DATA &H8E,&H00,&H00,&H6F
330 DATA &H8D,&H00,&H36,&HA6
340 DATA &H80,&H81,&H60,&H26
350 DATA &H10,&H10,&H8C,&H00
360 DATA &H01,&H26,&H1C,&H6C
370 DATA &H8D,&H00,&H26,&H10
380 DATA &H8E,&H00,&H00,&H20
390 DATA &H12,&H10,&H8E,&H00
400 DATA &H01,&H1F
410 DATA &H20,&HC4,&H1F,&H26
420 DATA &H08,&H6C,&H8D,&H00
430 DATA &H12,&H10,&H8E,&H00
440 DATA &H00,&H8C,&H06,&H00
450 DATA &H26,&HD3,&HE6,&H8D
460 DATA &H00,&H05,&H4F,&HBD
470 DATA &HB4,&HF4,&H39,&H00
480 DEFUSR0=&H3E00
490 A=USR0(0)
500 PRINT @ 480,"WORD COUNT=";A;
510 GOTO 510
```

Listing 6: EXPLODE1

```
3E00                0100        ORG     $3E00
                    0110 * EXPLODING SCREEN
3E00 BD   B3ED      0120 ENTER  JSR     $B3ED   CONVERT DELAY TO INTEGER
3E03 FD   3E74      0130        STD     DELAY   STORE
3E06 108E 3E51      0140        LDY     #START  SET TO TABLE START
                    0150 * OUTER LOOP SCANS TABLE
3E0A 7F   3E76      0160 LOOP1  CLR     DONE      RESET ACTIVITY FLAG
                    0170 * INNER LOOP HANDLES EACH CHAR
3E0D AE   A4        0175 LOOP2  LDX     ,Y          GET CURRENT CHARACTER
3E0F 8C   0400      0180        CMPX    #$400       TEST FOR OFF SCREEN
3E12 2D   22        0190        BLT     OUT         GO IF OFF (UP)
3E14 8C   05FF      0200        CMPX    #$5FF       TEST FOR OFF SCREEN
3E17 22   1D        0210        BHI     OUT         GO IF OFF (DOWN)
3E19 86   60        0230        LDA     #96         BLANK CODE
3E1B A7   84        0240        STA     ,X          STORE BLANK
3E1D EC   A4        0250        LDD     ,Y          GET LOCATION
3E1F E3   22        0260        ADDD    +2,Y        ADD DISPLACEMENT
3E21 ED   A4        0270        STD     ,Y          STORE NEW LOC'N
3E23 1F   01        0280        TFR     D,X         IN X FOR INDEXING
3E25 8C   0400      0290        CMPX    #$400       TEST FOR OFF SCREEN
3E28 2D   0C        0300        BLT     OUT         GO IF OFF (UP)
3E2A 8C   05FF      0310        CMPX    #$5FF       TEST FOR OFF SCREEN
3E2D 22   07        0320        BHI     OUT         GO IF OFF (DOWN)
3E2F A6   24        0330        LDA     +4,Y        GET CHARACTER
3E31 A7   84        0340        STA     ,X          STORE NEW CHARACTER
3E33 7C   3E76      0350        INC     DONE        SET ACTIVITY
3E36 31   25        0360 OUT    LEAY    +5,Y        BUMP TABLE PNTR
3E38 108C 3E74      0370        CMPY    #TEND       TEST FOR END OF TABLE
3E3C 26   CF        0380        BNE     LOOP2       GO IF NOT
                    0390 * END OF ONE PASS HERE
3E3E 7D   3E76      0400        TST     DONE      TEST FOR ACTIVITY
3E41 27   0D        0410        BEQ     ENDRET    GO IF NO ACTIVITY
3E43 108E 3E51      0420        LDY     #START    RESET TO TABLE START
3E47 BE   3E74      0430        LDX     DELAY     USER DELAY
3E4A 30   1F        0440 LOOP3  LEAX    -1,X      DELAY
3E4C 26   FC        0450        BNE     LOOP3     FOR USER COUNT
3E4E 20   BA        0460        BRA     LOOP1     CONTINUE
3E50 39             0470 ENDRET RTS             RETURN HERE
                    0480 * EXPLODE AT LINE 8, CP 12
3E51      050C      0490 START  FDB     $50C    E LOCATION
3E53      FFDF      0500        FDB     -33     E DISPLACMENT
3E55      45        0510        FCC     'E'
3E56      050D      0520        FDB     $50D    X LOCATION
3E58      001F      0530        FDB     31      X DISPLACEMENT
3E5A      58        0540        FCC     'X'
3E5B      050E      0550        FDB     $50E    P LOCATION
3E5D      FFE0      0560        FDB     -32     P DISPLACEMENT
3E5F      50        0570        FCC     'P'
3E60      050F      0580        FDB     $50F    L LOCATION
3E62      0020      0590        FDB     32      L DISPLACEMENT
3E64      4C        0600        FCC     'L'
3E65      0510      0610        FDB     $510    O LOCATION
3E67      FFE0      0620        FDB     -32     O DISPLACEMENT
3E69      4F        0630        FCC     'O'
```

of entry points for assembly language programs, as long as you precede each entry by a new DEFUSR.

```
1ØØ DEFUSR5 = &H3EØØ      'first entry point
.
2ØØØ A = USR5(Ø)           'go to AL code at $3EØØ
.
.
3ØØØ DEFUSR5 = &H3E1Ø      'new entry point
.
.
4ØØØ A = USR5(Ø)           'go to $3E1Ø
```

The same thing applies to Color BASIC — POKEing new values at 275/276 sets up a new entry point as many times as you'd like.

Dummy Variables

We've been using a value of zero within parentheses for the USRn. This value is a dummy value that satisfies the syntax (format) of the USRn statement. The variable 'A' on the left of the equals sign in A = USRn(0) is also a dummy. Use any variable you'd like for this. A little later on we'll see how these variables are used in passing data to and from the assembly language subroutine.

How Big Can the AL Subroutine Be?

In the preceding, we've used a three-instruction program to put an 'A' on the screen. In fact, we could make the assembly language subroutine as large as we want, anywhere from one instruction (just an RTS) to thousands of instructions. A final RTS returns to BASIC, however. (We say final because there may be any number of subroutines within the assembly language code, just as there are nested GOSUBs/RETURNs in BASIC.) Typically though, useful assembly language code is on the order of dozens of instructions. Some of the things that may be coded in dozens of lines of assembly language are screen text processing such as word wrap, sorts of data into alphabetical order, sound subroutines to make noises and music, high-speed animation and fast disk file processing.

Again, the idea is to find the parts of a BASIC program that really bog down in speed and break out these parts in assembly language code. Also, there are functions that are just not possible in anything other than assembly language because it's fast enough to handle real-world events such as sound generation and data communications (serial port) applications.

Passing an Argument Back from an Assembly Language Subroutine

How about another example of an assembly language program? This time we'll make it a little more useful. Let's try a program that counts the words on the screen. The hard part is coming up with the program in the first place. We'll use this following logic. A word is any text (including numbers and special characters) bracketed by spaces, line start, or a line end. In the line DISK EXTENDED COLOR BASIC 007 there are five words. More than one space may be present between words, as in UNDER LICENSE TO KILL. In this line there are four words.

The program for counting words is shown in Listing 4. It reads characters from text screen memory, locations 1024 through 1535, looking for space characters (96). Each time two space characters are found, a word count is incremented (one is added to the word count).

The BASIC language program with embedded assembly language code is shown in Listing 5. It looks pretty much like the first example, but there are 23 instructions in 66 bytes instead of three instructions in six bytes, as shown in the listing. This means that 66 values must be stored in the protected memory area. Also, the 'A' variable is used to receive the word count. Try this program from within any BASIC program — we've included a little driver program in BASIC to generate sample text on the screen for the count.

The operation of the program is described in the listing and we won't go into the details here. However, there is one "gotcha" that should be explained. In order to pass back the word count, a ROM subroutine must be used. The subroutine is located at $B4F4 in the Color BASIC interpreter. This subroutine takes a value in the D register and converts it into a BASIC variable. The variable is then set equal to the dummy variable used in the USR call, in this case 'A'. The call to $B4F4 is one way of passing back an argument computed in an assembly language subroutine to BASIC. We'll look at some other ways in the next column.

Passing an Argument to an Assembly Language Subroutine

As you might suspect, if you can pass an argument back from an assembly language subroutine, you can also pass an argument to an assembly language subroutine. As an example, look at Listing 6. It shows a short program to cause the letters EXPLODE to explode on the screen, flying apart as shown in Figure 3. The speed at which this is done is controlled by passing a value to the assembly language subroutine from BASIC — the smaller the value, the faster the explosion takes place.

The BASIC version of this program is shown in Listing 7. Like the other two programs, it has the machine language bytes in DATA values which are moved to the $3E00 area. The explosion speed is input from BASIC and then used in the argument of the USR0 call.

continued on Page 53

```
3E6A    Ø511    ØØ64Ø          FDB    $511    D LOCATION
3E6C    ØØ21    ØØ65Ø          FDB    33      D DISPLACEMNT
3E6E    44      ØØ66Ø          FCC    'D'
3E6F    Ø512    ØØ67Ø          FDB    $512    E LOCATION
3E71    FFE1    ØØ68Ø          FDB    -31     E DISPLACEMENT
3E73    45      ØØ69Ø          FCC    'E'
        3E74    ØØ7ØØ TEND     EQU    *
3E74    ØØØØ    ØØ71Ø DELAY    FDB    Ø       USER DELAY COUNT
3E76    ØØ      ØØ72Ø DONE     FCB    Ø       ACTIVITY FLAG
        ØØØØ    ØØ73Ø          END
ØØØØØ TOTAL ERRORS
```

Listing 7: EXPLODE2

```
1ØØ ' EXPLODING TEXT PROGRAM
11Ø CLEAR 1ØØØ,&H3DFF
12Ø CLS
13Ø PRINT@268,"EXPLODE";
14Ø FOR I=&H3EØØ TO &H3E76
15Ø READ A: POKE I,A
16Ø NEXT I
17Ø DATA &HBD,&HB3,&HED,&HFD
18Ø DATA &H3E,&H74,&H1Ø,&H8E
19Ø DATA &H3E,&H51,&H7F,&H3E
2ØØ DATA &H76,&HAE,&HA4
21Ø DATA &H8C,&HØ4,&HØØ,&H2D
22Ø DATA &H22,&H8C,&HØ5,&HFF
23Ø DATA &H22,&H1D,&H86,&H6Ø
24Ø DATA &HA7
25Ø DATA &H84,&HEC,&HA4,&HE3
26Ø DATA &H22,&HED,&HA4,&H1F
27Ø DATA &HØ1,&H8C
28Ø DATA &HØ4,&HØØ,&H2D,&HØC
29Ø DATA &H8C,&HØ5,&HFF,&H22
3ØØ DATA &HØ7,&HA6,&H24,&HA7
31Ø DATA &H84,&H7C
32Ø DATA &H3E,&H76,&H31,&H25
33Ø DATA &H1Ø,&H8C,&H3E,&H74
34Ø DATA &H26,&HCF,&H7D,&H3E
35Ø DATA &H76,&H27,&HØD,&H1Ø
36Ø DATA &H8E,&H3E,&H51,&HBE
37Ø DATA &H3E,&H74,&H3Ø,&H1F
38Ø DATA &H26,&HFC,&H2Ø,&HBA
39Ø DATA &H39,&HØ5
4ØØ DATA &HØC,&HFF,&HDF,&H45
41Ø DATA &HØ5,&HØD,&HØØ,&H1F
42Ø DATA &H58,&HØ5,&HØE,&HFF
43Ø DATA &HEØ,&H5Ø,&HØ5,&HØF
44Ø DATA &HØØ,&H2Ø,&H4C,&HØ5
45Ø DATA &H1Ø,&HFF,&HEØ,&H4F
46Ø DATA &HØ5,&H11,&HØØ,&H21
47Ø DATA &H44,&HØ5,&H12,&HFF
48Ø DATA &HE1,&H45
49Ø DATA &HØØ,&HØØ,&HØØ
5ØØ DEFUSR=&H3EØØ
51Ø A=USRØ(15ØØØ)
52Ø GOTO 52Ø
```

FRICKERS FOLLIES

by Jack Fricker

IF you have been reading the magazine you will have noticed that I have taken over the job of modifing the CoCoLink Bulletin board after Kevin left.

However this is an ongoing process and will probably be going on for some time to come until we get it to work just the way we want it to.

While we are on the subject of CoCoLink some of the processes have been changed. For instance there will be public domain software available for down-loading, but the facility for up-loading will have been removed. If you do wish to donate some PUBLIC DOMAIN software please send it to myself or to Graham.

While we are on the subject of public domain software, I was reading that there is a PUBLIC DOMAIN version of the WORDPAC 2 driver that works with OS-9 version 2.0, that was available on either the SOURCE or one of the other BBS's in the U.S. Iif anyone has this I would be grateful for a copy.

Other changes have been made to make it easir to use the system and changes have been made to allow more access to the public. There will be other changes to the Bulletin Board and I will keep you informed when they are made.

Why am I talking about the Bulletin Board in an OS-9 column you may ask? Well, the Bulletin Board is OS-9. When you log on to the Bulletin Board you are entering as the t2 or third user which means that you may be sharing the system with two other users.

To give you an example when I took over the running of the Bulletin Board the first thing that I did was to put it on my 68000 system which runs a 68000 version of OS-9. Because of the increased memory capacity of the 68000 in my case 500K (516,000 bytes) I am able to load all the basic modules of the Bulletin Board into memory at one time by giving BASIC09 the option of using 100K or more of memory (try that MSDOS users!).

This still left me with a couple of hundred K of memory for other uses. In fact I did and still do have 2 or more versions of the Bulletin Board in memory at one time using different terminals. Remember that OS-9 is a multi-user and multi-tasking Operating System. It is just these facilities that enable me to have 2 or more terminals hooked up to the system and 2 or more versions of the program in memory at the same time, without any problems.

In fact I have 2 out of a possible 4 terminals and 2 printers hooked up to my system. One of these is a dedicated data terminal and the other is one of my Colour Computers running a terminal program. This allows me to run a program using one terminal and use the other terminal to check what the first program is doing.

One of the advantages of OS-9 is that I can run a BASIC09, C or PASCAL program on either my 68000 OS-9 system or my CoCo Level 1 OS-9 system or even on a 6809 level 2 OS-9 system when I get one. Even the disks are interchangable if you have the right software. The software to read the CoCo OS-9 disks on the OSK system comes with it. The ability to read and write standard OS-9 format disks on the CoCo comes with SDIDK replacement module.

OSK is the nickname for OS-9 68000 because it is a lot easier to say and itis still OS-9. You may have read that microware and OSK is the operating system that was chosen by PHILLIPS and SONY for the new CD-ROM (compact disk) systems. These read only disks hold vast amounts of data available at lightning speed. You will not be able to actually see the operating system but it will handle all the input and output to the laser disks.

One of the things that makes all these things possible is that no version of OS-9 cares where its' input comes from or where its' output goes to. To add a device such as an additional terminal or printer all that is required is to add the device driver and a device descriptor and the hardware to communicate with.

With this ability theoretically you could have 12 terminals and multiple printers all working on your CoCo at the same time. This really wouldn't be practical but it is possible. This is how things like Hard Disks and 80 column displays and RAM disks are added.

continued from Page 52

Within the assembly language code, the first action that must be taken is to call another ROM subroutine in Extended BASIC at $B3ED. This ROM subroutine converts the BASIC variable to an integer value from zero to 65,535 in the D register, which is then used to control the speed of the explosion.

Again, this subroutine is one way to pass an argument to an assembly language program from BASIC, but not the only way. We'll look at alternative approaches next month. In the meantime, try your hand at embedding short assembly language code in BASIC. A few samples will help in your understanding of the process. □

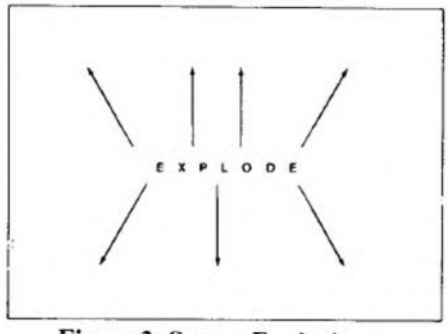


Figure 3: Screen Explosion

GOLDSOFT

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

ORDER FORM

AUSTRALIAN RAINBOW	12 mnths	$ 39.95 .....
	6 mnths	$ 24.75 .....
	1 mnth	$ 4.50 .....
AUSTRALIAN CoCo	12 mnths	$ 35.00 .....
	6 mnths	$ 21.35 .....
	1 mnth	$ 3.75 .....
CoCoOZ on tape	12 mnths	$ 75.00 .....
	6 mnths	$ 42.00 .....
	1 mnth	$ 9.50 .....
CoCoOZ on Disk	12 mnths	$102.50 .....
	6 mnths	$ 58.00 .....
	1 mnth	$ 10.95 .....
MiCoOZ on Tape	12 mnths	$ 75.00 .....
	6 mnths	$ 42.00 .....
	1 mnth	$ 9.50 .....
RAINBOW ON TAPE (AUST/US)	12 mnths	$144.00 .....
	6 mnths	$ 81.00 .....
	1 mnth	$ 15.00 .....
RAINBOW ON DISK (AUST/US)	12 mnths	$172.00 .....
	6 mnths	$ 96.75 .....
	1 mnth	$ 15.00 .....

Or charge my credit card monthly TAPE/DISK ONLY

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

Additional Requirements:

.....

.....

.....

Sub No: ☐☐☐☐☐☐ or ☐ New Subscription

Name: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Address: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

☐☐☐☐☐☐☐☐☐☐☐☐☐P.C.☐☐☐☐

Phone No.: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Please find enclosed CHQ / MONEY ORDER / NO CASH
Please charge my MASTERCARD / BANKCARD / VISA

Credit Card No.: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Authorised amount $ .....

Signed : .....

GOLDSOFT
Hardware & Software for your TANDY computer.

HARDWARE		
The CoCoConnection: Connect your CoCo to the real world and control robots, models, experiments, burglar alarms, water reticulation systems — most electrical things. Features two MC 6821 PIAs; provides four programmable ports; each port provides eight lines, which can be programmed as an input or output; comes complete with tutorial documentation and software; supplied with LED demonstration unit. Switchable memory addressing allows use with disk controller or other modules via a multipack interface; plugs into Cartridge Slot or Multipack, uses gold plate connectors; a MUST for the hardware designer and debugger!		$206.00
Video-Amp: Connects simply to your CoCo to drive a Colour or Mono monitor.	With Instructions With Instructions and sound	$25.00 $35.00
The Probe: A temperature measuring device which attaches to the joystick port of your CoCo or T1000, or to the joystick port of your CoCo Max. Comes with programs to start you thinking, and is supported monthly in Australian CoCo magazine.		$39.95
SOFTWARE		
Magazines: Australian Rainbow Magazine — THE magazine for advanced CoCo users! Australian CoCo Magazine — THE magazine for the new user of a Tandy computer. Also suits owners of CoCos, MC 10s, Tandy 1000s, 100s, 200s & 2000s.		
Back Issues: Australian Rainbow Magazine. (Dec '81 to now.) Please Note: Some months out of stock. Australian CoCo Magazine. (Aug '84 to now.) Please Note: Some months out of stock. CoCoBug Magazine. For CoCo — usually 8 programs in each magazine. (Sep '84 to Oct '85) Australian MICo Magazine. For Tandy MC 10 computers. (Dec '83 to Jul '84) Australian GoCo Magazine. For Tandy Model 100 users. (Jul '83 to Jul '84)	Australian Rainbow 1986 1982–1985 Australian CoCo 1986 Sept 1984–1985 each each each	$4.50 $2.50 $3.75 $3.00 $1.00 $2.00 $1.50
CoCoOz, on Tape or Disk: The programs you see listed in Australian CoCo Magazine are available on CoCoOz! No laborious typing — just (C)LOAD and Go! Back issues of CoCoOz are always available	Each Tape Subscription, 6 months 12 months Each DISK Subscription on disk, 12 months	$9.50 $42.00 $75.00 $10.95 $102.50
Rainbow on Tape, or Disk: Australian. The programs you see listed in Australian Rainbow Magazine are available on tape. A boon if you don't understand the language! American. We also supply the programs found in American Rainbow on tape. Please specify either Australian or American.	Each Tape Subscription, 12 months NEW for 1986 ONLY Each DISK Subscription on disk, 12 months	$15.00 $144.00 $15.00 $172.00
MICoOz: The programs in the MiCo section of Australian CoCo Magazine. (For MC 10 computers only) Back issues of MICoOz are always available.	Each Tape Subscription, 12 months	$9.50 $75.00
GOLDDISK 1000 — programs from 'softgold' for your Tandy 1000 on disk		$10.95
CoCoLink: CoCoLink is our Bulletin Board which you can access with any computer if you have a 300 baud modem and a suitable terminal program. There is a free visitor's facility, alternatively membership entitles you to greater access of the many files available. We can also be contacted through Viatel (Telecom).	Subscription to CoCoLink, 12 months	$29.00
Books: HELP: A quick reference guide for CoCo users. BYTE: Guide for new CoCo users. MiCo HELP: A quick reference for owners of MC 10 computers.		$9.95 $4.00 $9.95
Othello: by Darryl Berry The board game for your CoCo.	Tape 16K ECB	$15.95
Say the Wordz: by Oz Wiz & Pixel Software Two curriculum based speller programs for your Tandy Speech/Sound Pack.	Tape 32K ECB	$29.95
Bric a Brac: Blank tapes .....12 for $18.00 or $1.70 each Cassette Cases ..... 12 for $3.50 Disks .. (They work!) ..... $3.50 each or $29.50 per box of 10.		

HOW TO ORDER

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

The Best of CoCoOz:

Tape	$10.00
Disk	$16.00

What's on:

Best of CoCoOz #1. **EDUCATION**

ROADQUIZ	ROB WEBB	MARKET	ALEPH DELTA
HANGMAN	ALEPH DELTA	TOWNQUIZ	ROB WEBB
AUSTGEOG	P. THOMAS	ALFABETA	RON WEBB
SPELL	IAN LOBLEY	TANK ADDITION	DEAN HODGSON
FRACTUT	ROBBIE DALZELL	TABLES	BARRIE GERRAND
ICOSA	BOB WALTERS	KIDSTUFF	JOHANNA VAGG
TAXMAN	TONY PARFITT	FLAGQUIZ	ROB WEBB

Best of CoCoOz #2 part 1. **16K GAMES.**

LE-PAS	Wrongsoft	PYTHON	?
COCOMIND	STEVE COLEMAN	POKERMCH	GRAHAM & MATTHEWS
OILSLICK	JEREMY GANS	SPEEDMATH	DEAN HODGSON
CCMETEOR	BOB THOMSON	LNDATTCK	ALDO DEBERNARDIS
BATTACK	JEREMY GANS	INVADERS	DEAN HODGSON
PROBDICE	BOB DELBOURGO	RALLY	TONY PARFITT
CHECKERS	J & J GANS	FOURDRAW	JOHANNA VAGG

Best of CoCoOz #2 part 2. **32K GAMES.**

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

Best of CoCoOz #3. **UTILITIES.**

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

Best of CoCoOz #4. **BUSINESS**

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

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

Tape	**$10.00**
Disk	**$16.00**

User Group Contacts

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

ACT:

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

NSW:

SYDNEY:

BANKSTOWN	CARL STERN	02 646 3619
BNKSTWN WEST	ARTH PITTARD	02 72 2881
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSKO MCKAY	02 624 3353
CHATSWOOD	BILL O'DONNELL	02 419 6081
CLOYTON	HERMAN FREDRICKSON	02 6236379
GLADESVILLE	MARK ROTHWELL	02 817 4627
HILLS DIST	ARTHUR SLADE	02 622 8940
HORNSBY	ATHALIE SMART	02 848 8830
KENTHURST	TOM STUART	02 654 1610
LEICHHARDT	STEVEN CHICOS	02 560 6207
or	GORGE ECHEGARAY	02 560 9664
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACQUARIE FIELDS	BARRY DARNTON	02 618 1909
ROSEVILLE	KEN UZZELL	02 467 1619
SUTHERLAND	IAN ANNABEL	02 528 3391
SYDNEY EAST	JACKY COCKINOS	02 344 9111

ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	DOUG BARBER	067 72 7647
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BROKEN HILL	TERRY NOONAN	080 88 2382
CAMDEN	KEVIN WINTERS	046.66.8068
COFFS HARBOUR	BOB KENNY	066 51 2205
COOMA	ROSS PRATT	0648 23 065
COORANBONG	GEORGE SAVAGE	049 77 1054
COOTAMUNDRA	CHERYL WILLIS	069 42 2264
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 2095
or	MIKE MUNRO	068-82-5011
FORBES	JOHANNA VAGG	068 52 2943
FORSTER	GARY BAILEY	065 54 5029
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	PETER LINDSAY	066 42 2503
GUYRA	MICHAEL J. HARTMANN	067 79 7547
JUNEE	PAUL MALONEY	069 24 1860
KEMPSEY	RICK FULLER	065-62-7222
LEETON	BRETT WALLACE	069-53-2081
LISMORE	ROB HILLARD	066 24 3089
LITHGOW	DAVID BERGER	063 52 2282
MAITLAND	BILL SNOW	049 66 2557
MOREE	ALF BATE	067 52 2465
MUDGEE	BRIAN STONE	063-72-1958
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 2095
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
ORANGE	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PORT MACQUARIE	RON LALOR	065 83 8223
SPRINGWOOD	DAVID SEAMONS	047 51 2107
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
URALLA	FRANK MUDFORD	067 78 4391
WAGGA WAGGA	CES JENKINSON	069 25 2263
WYONG	JOHN WALLACE	043 90 0312

NT:

DARWIN	BRENTON PRIOR	089.81.7766

QLD:

BRISBANE:

BIRKDALE	COLIN NORTH	07 824 2128
BRASSALL	BOB UNSWORTH	07 201 8659
EAST	ROB THOMPSON	07 848 5512
IPSWICH	MILTON ROWE	07 281 4059
NORTH	JACK FRICKER	07 262 8869
PINE RIVERS	BARRY CLARKE	07 204 2806
SOUTH WEST	BOB DEVRIES	07 375 3161
SANDGATE	MARK MIGHELL	07 269 5090
SCARBOROUGH	PETER MAY	07 203 6723

BIGGENDEN	ALAN MENHAM	071 27 1272
BLACKWATER	ANNIE MEIJER	079.82.6931
BOWEN	TERRY COTTON C/O	077 86 2220
BUNDABERG	RON SIMPKIN C/O	TANDY
CAIRNS	GLEN HODGES	070 54 6583
DALBY	ANDREW B. SIMPSON	074.62.3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 51 0015
HERVEY BAY	LESLEY HORWOOD	071 22 4989
MACKAY	LEN MALONEY	079511333x782
MARYBOROUGH	NORM WINN	071 21 6638
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280
MURGON	PETER ANGEL	071 68 1628
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
TARA	STEVEN YOUNGBERRY	
TOOWOOMBA	GRAHAM BURGESS	076 30 4254
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
WHITEROCK	GLEN HODGES	070 54 6583

SA:

ADELAIDE	JOHN HAINES	08 278 3560
NORTH	STEVEN EISENBERG	08 250 6214
GREENACRES	BETTY LITTLE	08 261 4083
MORPHETTVALE	KEN RICHARDS	08 384 4503
PORT NOARLUNGA	ROB DALZELL	08 386 1647
SEACOMBE HTS	GLENN DAVIS	08 296 7477
PORT LINCOLN	BILL BOARDMAN	086 82 2385
PORT PIRIE	KEVIN GOWAN	086 32 1368
WHYALLA	MALCOLM PATRICK	086 45 7637

TAS:

HOBART	BOB DELBOURGO	002 25 3896
KINGSTON-	WIM DE PUIT	002 29 4950
WYNYARD	ANDREW WYLLIE	004 35 1839

VIC:

MELBOURNE:

MELBOURNE CCC	JOY WALLACE	03 277 5182
DANDENONG	DAVID HOPROCKS	03 793 5157
DONCASTER	JUSTIN LIPTON	03 857 5149
FRANKSTON	BOB HAYTER	03.783.9748
NARRE WARREN	LEIGH EAMES	03 704 6680
NTH EASTERN	KEVIN KAZAZES	03 437 1472
MELTON	MARIO GERADA	03 743 1323
RINGWOOD	IVOR DAVIES	03 758 4496
SUNBURY	JACK SMIT	03.744.1355

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

WA:

PERTH	IAN MACLEOD	09 448 2136
KALGOORLIE	TERRY BURNETT	090.21.5212

CANADA - CoCo:

Ontario	Richard Hobson	416 293 2346

SPECIAL INTEREST GROUPS

BUSINESS:

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

OS9 GROUPS:

NATIONAL OS9 USERS' GROUP	GRAEME NICHOLS	02 451 2954

NSW

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

QLD

BRISBANE	JACK FRICKER	07 262 8869

VIC

LATROBE VLY	GEORGE FRANCIS	051 34 5175

WA

KALGOORLIE	TERRY BURNETT	090.21.5212

MC-10 GROUPS:

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

TANDY 1000 / MS DOS:

QLD:

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

VIC:

MELBOURNE	TONY LLOYD	03 500 0878

NSW:

GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY WEST	ROGER RUTHEN	047.39.3903
WYONG	JOHN WALLACE	043 90 0312

FORTH:

BRISBANE	JOHN POXON	07 208 7820
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
SYDNEY	JOHN REDMOND	02 85 3751

ROBOTICS:

BOWEN	TONY EVANS	077 86 2220
GOLD COAST	GRAHAM MORPHETT	075 51 0015
TAMWORTH	ROBERT WEBB	067 65 7256
WAGGA WAGGA	CES JENKINSON	069 25 2263

CHRISTIAN USERS' GROUP:

COLLIE	RAYMOND L. ISAAC	097 34 1578

300 BAUD BULLETIN BOARDS

SYDNEY:

INFOCENTRE	02 344 9511
TANDY ACCESS	02 625 8071
THE COCO - CONNECTION	02 618 3591
DAIL DUBBO (6pm - 8am)	068 82 5011

QLD:

CoCoLink	075 32 6370

WA:

COCO UG	09 307 1397

1200/75 BAUD TANDY INFORMATION

VIATEL:

GOLDLINK	#642#

GOLDLINK

a Goldsoft Service

on

VIATEL®

*642#

AUSTRALIAN RAINBOW MAGAZINE
Registered by Australia Post –
Publication No. QBG 4009
AUSTRALIAN CoCo / softgold
Publication No. QBG 4007
P.O. BOX 1742
SOUTHPORT. QLD. Australia. 4215.

POSTAGE
PAID
AUSTRALIA