

For your  
TANDY  
Color Computer

Registered by Australian Post — Publication No. QBG 4009

PNG K5.95 NZ \$3.95

\$4.50

AUSTRALIAN

# RAINBOW

May, 1986

No.59

WELCOME TO THE LAND OF  
**ADVENTURE**  
THRILLS-THRILLS-THRILLS

THE  
COCO

★  
ZONE

★ The Maze of Moycullen

★ The Mis-adventures of Martha G!

TANDY ELECTRONICS DEALER. (No 9320)

**TANDY COMPUTERS & ACCESSORIES**

best prices!

FREE DELIVERY THROUGHOUT AUSTRALIA

**90 DAYS WARRANTY**

bankcard welcome here

DISK DRIVE Ø FOR CoCo  
40 TRACK DSDD  
DISK ECB 1.4  
AUTO LINE NUMBERING, SUPPORTS FLEX & OS9.  
GMS Access time  
inc Controller and Manual \$599

**BAYNE & TREMBATH**  
3 Boneo Rd., Rosebud, Victoria 3940  
Ph: (059) 86 8288. A/H: (059) 85 4947

Bankcard & Cheque Orders accepted

**BLAXLAND  
COMPUTER  
SERVICES  
PTY. LTD.**

**(047) 39-3903**

**COCO & 1000  
SPECIALISTS**

CLASS ( 68008 ADD ON BOARD )

Standard unit includes:-

- RS MINI-DOS
- OS9 MINI-DOS totally interactive with COCO-OS9
- 256K RAM DISK full 256K continuous Ramdisk
- 8, 16, 32 bit Processing
- 2K or 8K Variable storage
- Centronics Parallel Port
- Built-in Power Supply
- Plugs directly into Rom Pack Port or Multipack

Price \$650.00

Options:-

- KAMELION- Interface Operating System approx. \$80
- LFASTV68- Ass. Lang. library of utilities POA.
- FHL CSC- Cross Assembler approx. \$90
- The BEST Cross Assembler- CRASMB16.32 POA.
- Spare Parallel Port

Options (FUTURE):-

- 68000 COCO Compatible BASIC- 150K+ free space
- OS9 full 68K (Disk)
- 512K Upgrade Planned
- RS232 Port

SUPER SPECIALS (Limited Stocks)

- 64K COCO ECB with true lower case \$399
- T200 Portable Computer (save \$400) \$999
- DMP-105 Printer (save \$40) \$310
- THOMPSON Amber Monitor with sound \$190
- DESKMATE for COCO (save \$20) \$100
- Tandy DELUXE Joystick (save \$7) \$ 43
- Modified ARCHER Joystick \$ 24
- CICADA 300 baud Modem \$150
- Tandy 300 baud Modem \$170
- CEI 2123 Modem (for business use) \$450
- AVTEK Multi Modem with auto answer \$349
- Illustrated Computer \$ 60
- Color Computer Learning Lab. \$ 60
- COCO NANOS Reference Card \$ 5

SEND FOR OUR FREE CATALOGUE AND SAVE \$\$

ANNOUNCING A CONTEST  
"THE BEST OS9 OR 68000 PROGRAM"  
THE PRIZE

"CLASS" 68008 ADD ON BOARD (VALUE \$650)  
THE WINNER WILL BE NAMED AT COCOCONF.86

**76A MURPHY ST. BLAXLAND 2774**

Reviews ..... P 4  
 Planegeo and Pgcalsrt  
 Symphony 12  
 Challenging Simulation  
 PenPal  
 Electronic Organizer  
 Access09 BBS  
**THE CoCo ZONE**  
 ..... Bruce Bell P 11  
**MAZE OF MOYCULLEN**  
 ..... Thomas Riley P 18  
**FANTASY**  
 ..... Dereck Powell P 22  
**MIS-ADVENTURES OF MARTHA**  
 ..... Andrew White P 23  
**MARQUEE**  
 ..... Chuck Foynter P 30  
**BANDY WORD GAME**  
 ..... H. Allen Curtis P 34  
**NOUVEAU ROCK'N'ROLL**  
 ..... Bill Bernico P 38  
**TITLE MAKER**  
 ..... Fred Scerbo P 39

**HOME MARKET VALUE**  
 ..... Harry Hallstrom P 42  
**What's New: THE 16 BIT!**  
 ..... P 44  
**CoCoCAD EXPANDED**  
 ..... Dennis Page P 45  
**Wrestling Assembly**  
 ..... Ian Clarke P 47  
**Pi to 10,000 DIGITS**  
 ..... William Barden P 48  
**RS-232 Interfacing**  
 ..... Graham Morphett P 54  
**Forth**  
**PATTERNS OF MANY COLOURS**  
 ..... John Redmond P 55  
**OS9**  
**Very Basically OS9**  
 ..... Jack Fricker P 58  
**Firing up BASIC09**  
 ..... Richard White P 60  
 .... Subscription Page .... P 64

## DISKETTES

SSDD (10) Nashua/Memx life wrnty .....	\$25.00
DSDD (10) C.I.S. 10 yr wrnty .....	\$25.00
DSDD (10) Nashua/Memx life wrnty .....	\$32.00

## MODEMS

1200/75, 300, Autoans/disconnect .....	\$260
with MS-DOS Videotex software .....	\$355
1200/75, 300, Hayes compatible, Auto dial, answer, disconnect, Auto baud rate select .....	\$499
1200, 1200/75, 300, Hayes compatible, Auto dial, answer, disconnect, Auto baud rate select .....	\$699

## PRINTERS

Brother M1109 100CPS .....	\$415
Brother M1509 180CPS .....	\$865
Epson GX80 100cps .....	\$495
Star SG10 120cps .....	\$580

## MONITORS

DTX 2000 medium res .....	\$499
DTX 2001 high res .....	\$589
Green or Amber composite .....	\$199

NEIL CARPENTER COMPUTING P/L  
 Phone (02) 818-4220 7 days  
 BANKCARD/MASTERCARD AVAILABLE  
 Phone orders welcome  
 PO Box R401, Royal Exchange, 2000  
 Overnight courier anywhere \$7

## COMPUTER STATIONERY SUPPLIES

**Continuous Computer Paper**  
 WE CATER FOR SMALL & LARGE  
 ORDERS  
 NONE TO SMALL, PLAIN OR PRINTED

**Dust Covers**  
**Continuous Labels**  
**Diskettes - Head Cleaners**  
**Adhesive Tapes**  
**Post-It-Products**  
**Ribbons and Tapes**

**FREE DELIVERY ANYWHERE IN  
SYDNEY**

PHONE RICK OR JEAN  
 R & J DISTRIBUTORS  
 CNR ELIZABETH DRIVE &  
 ROSLYN ST LIVERPOOL  
 (02) 601 1319 A/H 602 2396

# 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 IF W=7985 THEN B0 ELSE PRINT
  "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 CLEAR 200,&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 Morphet  
**Managing Editor** Graham Morphet  
**Accounts** Annette Morphet  
**Assistant Editor** Sonya Young  
**Advertising** Tracey Yapp  
**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 preceding month.

**Printed by:**  
Australian Rainbow Magazine  
P.O. Box 1742  
Southport, Qld. 4215.  
Registered Publication QBG 4009.

This material is COPYRIGHT. Magazine owners may maintain a copy of each program plus two backups, but may NOT provide others with copies of this magazine in ANY form or media.

## PRINT #—2

You either like adventures - or you hate them!

Me, I'd just like to think I could get through one occasionally!

This month our authors present a special treat - three adventures which really are "state of the art".

In fact two of these adventures are prize winners in US Rainbow's annual Adventure Games Contest.

The other is by our Andrew White.

Andrew takes you into our own offices here at Goldsoft for the adventure of your life; and in so doing, shows you some very interesting programming techniques.

So whether you are a mad adventurer, or you are just into programming, you should enjoy the brace of adventure programs this month.

Problems associated with the continuing cost of American dollars, and the resultant credit squeeze here in Australia, have been well documented on national television of late.

In addition to these problems, those who would supply you with programs and hardware for your computer, have had to deal with overseas suppliers who are sick of being ripped off by Australian pirates.

Hence many of the programs seen in the US press for CoCo, will not be available here; one, because some suppliers are not prepared to trust us, and two, because it is becoming uneconomic to be a software distributor - because of the privacy and because of the increased cost of doing business.

Strong support for your computer, no matter what brand, can only be had if YOU support the supporters!

We are very aware that a high percentage of Tandy home computer owners make a special effort to support this magazine. Needless to say, we are very grateful for that support and we need the support to continue.

But we - and you - need strong suppliers too, to give you local help and to supply you with new and exciting soft & hard ware.

It is tough getting the dollars together these days I know, but next time you decide you need a piece of equipment or software for your computer, remember those who support you (-ie. the advertisers in this magazine!).

Speaking of advertisers, we are grateful to a number of our advertisers who have offered prizes for the various competitions. These people include Bayne & Trembath, Blaxland Computer Services, Paris Radio, The Computer Hut and Tandy (Australia) Pty Ltd.

I understand that several other suppliers are offering prizes too, so in deference to them, I'll have to leave details of the actual prizes till next month (didn't I say that last month?!).

As you probably realise, these prizes, which include a printer, a 68000 upgrade board, a CoCoConnection, a Probe, a heap of software and a very NICE surprise for the Games Contest winner, will be awarded at CoCoConf'86.

So once more, we urge you to make that extra effort to be at CoCoConf'86. Apart from being at THE place to see the very latest in everything for your computer, "Conf" is designed to teach you about computing.

The tutorials we have lined up this year are just excellent!

Most of you started with a tape based system.

With any other computer, a tape system spells

disaster, but even businesses can operate a CoCo under tape with confidence.

There can be no denying however, that a disk system improves the performance of the CoCo very significantly, and right now, disk systems can be purchased for very little - especially compared with the prices of a few years ago.

Which disk system should you get?

As usual, there are always as many different pieces of advice as there are advisors. However do consider first the advantages of purchasing your disk drive from Tandy.

Tandy's DOS (Disk Operating System) is, of course, the standard. Those of us with "funny DOS's" are finding more and more that programs are being written which just won't work on anything other than a Tandy DOS.

And Tandy have the ability to service your disk system locally. That is a very big plus with any computer equipment!

Going against Tandy, is their stubborn refusal to sell double sided drives.

In this day and age, it seems silly not to have a double sided 40 track drive available for CoCo.

Others can supply such an animal - people such as Computerware for Micro's, St Mary's Software, Blaxland, and Paris Radio who have just released a very nice unit, based around the J&M controller.

And what about DOS?

There are several available, including one which will allow you to operate 80 track drives and another which is already set up to include a hard disk system.

There are several rules to follow when looking for a DOS -

1. get one which is compatible with the Tandy DOS;

2. check with other users of the DOS on its reliability - if you don't know anyone with it, then ask the salesman for the phone numbers of people with the DOS;

3. make sure it will handle not only the jobs you need to do now, but also tasks you may want to do in the future;

- & 4. make sure the DOS has some sort of warranty.

Many disk drives for sale in the recent past, have been thrown together very roughly, some could be dangerous in some situations, and others just don't work when their owners get them home - check all this before you take the system home.

Sounds almost scary doesn't it!

It can get worse - the disk system is not nearly as robust as CoCo, and needs careful attention.

Move it as little as possible - if you must move the drives, then do it with the head protectors in place. And keep liquids well away from the system.

It doesn't have to be bad news though.

We have drives here which have not been serviced in two years. Look after yours, and you too will get long and reliable service!

Finally, my sincere thanks to the very generous individuals who chose to put something back into their hobby by donating or arranging for computers at very special prices, to go to the unemployed kids in Liverpool, care of the local Tandy Computer Users' Group.

More are still needed, and the offer of six months CoCoOz and a program from The Computer Hut still holds. Please give NOW!

*Johanna*

# REVIEWS

Software Review

## Euclid would be Proud of *PLANECEO* and *PGCALPRT*

By John McCormick

*PLANECEO* and *PGCALPRT* both do practically the same thing: they calculate the various parts of most plane geometric figures (square, circle, polygon, rectangle, parallelogram, trapezoid, right triangle, oblique triangle and ellipse) from the various combinations of data you have available (diagonal, side, area, etc.). These programs are aimed at engineers and others who have a regular need for these calculations.

You might ask just why an engineer would need to have a program to tell him the area of a square is found by squaring one side. Well, if you do, please tell me (off the top of your head) what is the area of a circle having a chord length of 100 feet and a chord height of 10 feet? According to *PLANECEO* the area is 53,092.917 square feet.

Both *PLANECEO* and *PGCALPRT* are written entirely in BASIC "to permit the user to make changes" (a comment in the documentation I found prophetic).

The lack of machine language is no handicap since the actual calculations performed in the program are quite short and BASIC doesn't really slow down the operation at all.

A nice touch is the Conversions program, which converts various data, such as feet, inches and fractions to feet and decimals, or the reverse, to the nearest  $\frac{1}{16}$  inch and also converts angles in various ways.

### *PLANECEO*

This program comes on a double-sided disk (which must be turned over for some calculations) and has a smaller user's manual because almost anything you want to know about these geometrical figures is included in the program itself.

Talk about user friendly! There are over 100 menus in this program (if I didn't lose count) and you can even call up an illustration of a square and be shown graphically what a side looks like. The menus also offer a complete listing of the formulas used in all calculations and word descriptions of the process.

These extensive help menus don't slow down the program (other than the fact that the extensive on-screen documentation requires the two-sided disk) because they only appear if you request to see them. Even this is not much of a problem since all calculations for a given shape are done with no disk changes.

What *PLANECEO* doesn't have is a way to print out the results, which brings us to:

### *PGCALPRT*

This version does exactly the same calculations as the other but, because of the reduced on-screen documentation (the user's manual is over twice as big as *PLANECEO*'s), the entire program fits on one side of a disk, thus eliminating some delays.

*PGCALPRT* prints out results. Unfortunately, in my opinion, it always prints out results. In fact, it won't even run without a printer online. This feature can lead to lots

of wasted paper.

### BUGS

I wish I didn't need to have this section but, alas, these very user-friendly programs do contain some bugs.

In *PGCALPRT*, for instance, if you try to exit the conversions portion of the program you find that the authors accidentally used the conversions subprogram from *PLANECEO* and it therefore calls the wrong program (NE Error in Line 610). Anyone with experience in BASIC can easily repair this bug (change `RUN"PLANGE2"` to read `RUN"PGCALPRT"`). This repair allows the program to operate but still leaves a problem. The version mistakenly used in *PGCALPRT* does not print out results. This means that after running a conversion you must copy down the results before returning to your calculations. TASC acknowledges the problem and I feel certain they will have it corrected.

Another problem cropped up in *PLANECEO* when trying to use "Parts of a Circle." Line 1800 contains a reversed ')' (close parenthesis mark) after `POKE`. The program ran fine when I changed this to '(' (open parenthesis mark).

### Room for Improvement

The first point is a matter of judgment. The program contains no provision to prevent erroneous input data (for instance, you can calculate the area of a one-sided polygon). One of the authors, a chemical/mechanical engineer, told me that since the program is aimed at professionals he felt (subject to user complaints) that this modification is not needed.

Personally, I feel such a user-friendly program cried out for this further enhancement to prevent accidental input but, on the other hand, since the program only performs very limited calculations, there is very little chance that an erroneous input would lead to further trouble.

Now we come to what I consider a serious mistake. I did not like the fact that *PGCALPRT* did not allow me the option of printing or not printing my results; I think that changing this would be a great improvement in the usefulness of the program.

It may be nit-picking, but the fact is that neither program addresses the problem of the trapezium (for those who need a refresher, a trapezium has four sides with no two parallel). The user's manual points out . . . "It is a very common figure. Most real property is of this shape . . ." and recommends that a trapezium be broken down to two triangles and solved that way. This is certainly the way to approach a trapezium, but I feel a program that offers to define the area of a square should have presented more information about this "very common figure."

I know it is easy for a reviewer to make suggestions since he didn't sweat over the original product, but I have a few anyway.

The authors consistently point out that the program is designed for professionals, yet they include even the most elementary of explanations. This is not a fault, it is a virtue. I feel that if *PLANECEO* had a bit more information added it would make a fine educational tool; it is almost ready to give to a high school student in its present form, and certainly with a little added information (perhaps just a new user's manual) it would be a better educational program than many I have seen.

My other suggestion is to make printing optional for

PGCALPRT instead of mandatory. In my opinion, this would make it a better product for professionals.

### Conclusion

Apart from some minor bugs, these are good programs — very user friendly. In my college days we carried slide rules and CRC books at all times and dreamed of programs like these.

### Hardware/Software Review

## Symphony 12 is an Excellent Music Synthesizer

— James Ray

*Symphony 12* is another release from Speech Systems and is one of a long line of excellent products developed by them for the Color Computer. *Symphony 12* is a 12-voice hardware music synthesizer. It comes in a ROM pack along with tape/disk software that contains the necessary programs to operate the synthesizer and some sample music files for listening. This creation is a marvel for a machine that is supposed to have one operational voice. Some software developers have created programs with BASIC and machine language to create up to four voices, but this from Speech Systems emulates 12 separate voices. It features four noise channels and plays in stereo or mono. It can be hooked up to your home stereo or jam box. Special sound effects can be created and *Symphony 12* is compatible with *Musica* files (Version 2.7).

*Symphony 12* alone comes with a program disk or tape and the ROM Stereo Pack. The options available include: Piano Keyboard (2½ octaves) for \$79.95, Piano Keyboard (4 octaves) for \$119.95 and Y-Cable (required for disk) for \$28.95.

I used a Mutli-Pak Interface to connect the hardware cartridge and disk controller to my computer. I put the disk controller in Slot 4 and the sound cartridge in Slot 3. The sound cartridge has two RCA-type jacks that can connect to your stereo system. If you want to connect the stereo cartridge to your computer monitor, you need a Y-adapter cable so you can hear both output channels. *Symphony 12* comes with a stereo patch cord to connect the stereo cartridge to your stereo or computer monitor.

I didn't have one of the piano keyboards for this review, therefore I had to use the CoCo keyboard. *Symphony 12* uses the bottom two rows of keys as the piano keyboard. The 'Z' through '/' keys are the white notes (A, B, C, D, E, F, G, A, B and C). The second row of keys, 'S' through 'L', are the black notes and correspond to B flat, D flat, etc. The CoCo keyboard allows you to play around with the various voices of *Symphony 12*. However, it is very limiting and difficult to play anything halfway serious. Octaves can be raised and lowered, but must be done manually by pressing the SHIFT key and one of the up- or down-arrow keys. You are given a visual piano keyboard on the computer screen and indicators are given as each note or chord is played. For serious users, Speech Systems and I both recommend one of their piano keyboards.

If you are already a *Musica* owner, you may use your *Musica* creations with *Symphony 12*. However, only four voices are available to you. To take advantage of the 12 simultaneous voices, you must create the music with *Symphony 12*.

*Symphony 12* gives you control of many aspects of generating various sounds. You have control of volume, the

envelope (sound shape), noise, rhythm and preset instrument settings. The program comes with nine preset settings, each of which can be changed or adjusted to fit your particular needs. This allows the user to quickly change from one voice setting to another. Once you set up voicing to your liking, you can save the settings to tape or disk for future use.

The music you create with *Symphony 12* can be recorded in real time, or saved to disk or tape for future playback. When you press the 'R' key, everything you play is remembered by the computer and replayed at your command. The manual does not indicate how much the program can remember, but this is an excellent feature. If you make a mistake, you must start over again with the recording and the previous one is erased. A special feature of playing music is when you press a note to sound, you can "bend" the sound up or down. By pressing the CLEAR key you raise the pitch, and the SHIFT-CLEAR to lower the pitch. I found this feature absolutely fascinating!

Envelope and noise control are somewhat complicated, but the manual gives very simple information to control these features. The manual also gives technical specifications for the A4-3-8912 chip that is the heart of the *Symphony 12* program. This would be useful to serious programmers and technical users. Noise can be used to accompany the sound so that it takes on a breathy quality, but noise and rhythm cannot be on at the same time.

When playing musical files, you are asked if they are in the *Musica* format or the *Symphony 12* format. You can play a *Musica* file in the *Symphony 12* format, but remember, you can only use four voices. One nice feature this method allows is to change instrument settings as the music is played. This is particularly fun and enjoyable. You can also play each of the four voices as separate instruments.

*Symphony 12* has a demo program that shows how to create various sound effects. You hear a wolf whistle, race car, Pacman laser, bomb, steam locomotive, and a little Bach, among others. *Symphony 12* files can be accessed from BASIC and the manual gives those instructions and a sample printed program.

This program would be a great addition to any music lover or user's library. *Symphony 12* does not support printer operations, therefore files must be recorded in the *Musica* format and transferred to the *Musica* program for printing. Speech Systems has quite a library of *Musica* files for your enjoyment and they are quite inexpensive. There is also a National *Musica* Users Group that supports this program.

You can gather that I enjoyed *Symphony 12*. My only dream is to have a program that allows you to play a note on a keyboard (piano or computer), display the note on a musical staff, sound the note or chord and send it to a printer. Maybe that will be next from Speech Systems.

## COCOCONF '86

### 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.

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

## Conquering Armies is a Challenging Simulation

— Mark Williams

*Conquering Armies* is a satisfyingly difficult game to play. There are 50 levels of difficulty. Level 50 must be a real killer, because the highest I was able to go was level seven. The game is played on a map of the mythical kingdom of Glasco. You are the heir to the throne, and six of seven of your provinces have been taken over by enemy forces. Your task is to recapture your lands. (No matter at which level you play, this scenario stays the same, i.e., the same six provinces are always under enemy control and you are always in control of Avon.)

Depending on the level of play selected, the number of troops you control will differ. They are divided into the categories of knights, light cavalry, men-at-arms and archers. Typing 'I' at any point gives an inventory of the areas and troops you control and gives the option of creating an army with some or all of those troops. You also have one, two or three allies that can help by giving extra troops if you can get one of your armies to their castles — not always an easy task. The number of allies varies with the playing level and random chance. Each time you recapture one of your lands, you also gain troops. The liberated inhabitants also enlist with you, showing their gratitude by their willingness to be slaughtered in your next battle.

The playing screen is a nicely done map. The small red castles represent castles that rule the country of their location. The large red castles are capital cities. Red lines mark off countries, with yellow areas controlled by you and green areas controlled by the enemy. A blinking green dot indicates an enemy army on the move. Since enemy territories are green, their armies are invisible until they cross your borders. They will occasionally flash, giving you the equivalent of a "rumor" of their approach, a nice touch. Your armies (you can have up to five armies active at any one time depending on the troops you control) also show as a blinking dot, visible at all times. You move your troops with the right joystick.

You win this game by recapturing all your lands, a feat accomplished by successfully laying siege to each area's castle. Once you start a siege, the computer runs the battle. You will win a siege if you outnumber the defenders, but your troop strength is often reduced by ambushes that occur with distressing regularity, another realistic touch. Also, the enemy is likely to attempt to retake a castle they have just lost, so leaving a garrison behind is a must. This, of course, limits the amount of troops you can put in the field.


You can also confront an enemy army in the field—here's where those "rumors" come in handy. Since the enemy moves at the same speed that your armies do, you must head into them or cut them off; there is no catching up with them in this game. Again, once the battle is joined, the computer takes over, and again, you will win if you outnumber the enemy, especially in the key areas of knights and light cavalry. The odds are slightly in your favor if you are defending a castle, slightly in the enemy's favor if you are besieging them, and equal for a battle in the field (assuming equal numbers of forces). This is an attempt to recreate the actual odds of medieval battles, where the defenders of a castle did have an advantage over an attacking army.

Earlier I said that you win the game by recapturing your lands. I find that this is not quite true. Once you have

retaken your lands, the enemy launches anywhere from one to three more attacks. If you defeat these, you win. If the enemy recaptures one of your castles, the game continues with the enemy sending in increased numbers of troops, a touch you may not think of as a plus.

It takes several playings to get a feel for this game, but your skill increases with each round. Luck does play a part in this game, but skill and foresight can carry you through some bad luck. There is just enough randomness (the ambushes, the number of allies, etc.) to make the game a challenge at any level. The game played smoothly at all the levels I tried. Indeed, the only complaint I have is the slowness of movement of the armies. All the people I invited to try this game, adults and children, had the same complaint. Some also complained about the length of time some of the battles took, but this becomes a problem only if you have or are confronting really massive (20,000+) numbers of troops. The documentation (three typed pages) is more than adequate. The game has many little pluses, even including some humor in the documentation.

All in all, *Conquering Armies* is a winner — fun to play, and as challenging in its own way as some text Adventure games.

Software Review 

## PenPal is Useful and Affordable

— Dale Shell

*PenPal* is an integrated software package that includes a word processor, a spreadsheet, database, communications package and a graphics package. Users have options to print out data from the spreadsheet and produce pie graphs, horizontal or vertical lines, dot or bar graphs. Files can be transferred between modules, saved to disk, sent to a printer or transmitted over a modem. All these functions can be accomplished without leaving the integrated program.

Each program module has its own online help screen at the bottom of the screen. The keys '1' through '9', '0', '.' and '-' in conjunction with the CLEAR key and the SHIFT-CLEAR keys act as functions 1 through 12 and alternates 1 through 12 respectively.

*PenPal* requires 64K and at least one disk drive. While a graphics printer and a modem are considered optional, there are many features you will not be able to use without them. A second disk drive is also useful since you must have a program disk in Drive 0. If you only have one drive and must save disk space, you make a program disk that contains only the modules and help files you are using during that particular session. With two drives you have all the modules on the program disk in Drive 0 and you have a files disk in Drive 1. Two drives increase the versatility of *PenPal*.

Besides the five modules in the integrated package, the main menu also allows manipulation of files. Function keys in the main menu allow swapping the default drives, renaming files, display free space on the default drive, kill files, display an expanded directory, change the drive step rates or run a configure program to set all of the above defaults. Along with these seven functions, *PenPal* has four alternate functions that are available at any time from any of the modules and main menu. A simple calculator is available, the option to set up the printer defaults, toggle between white and green screen display modes or a help file can be called up for the specific module currently engaged.

Now, the individual programs do not always have all the



bells and whistles some of the larger, more expensive programs have, but in a few cases *PenPal* has more bells and less whistles. For example, the Write module is slightly restricting in that you definitely do not get what you see, i.e., the way the text looks on the screen is not necessarily how the printout looks. In my opinion, this is the chief weakness of the program. I would like to see a 64-column mode. The screen is Hi-Res consisting of 50 columns. While the Write module does not have all the options some bigger programs have, it does have 18 functions, making the first nine function keys used twice. These functions include Find and Replace, an overstrike or insert mode, delete character and line functions and the ability to merge files to the end of the current file. Also available are multiple block functions.

The Calc module includes most of the more popular functions. The spreadsheet is laid out into 255 columns and 255 rows. Of course, you cannot have a 255 column by 255 row spreadsheet but this layout does allow the user good flexibility. The Database module includes many options of the more popular databases. While no database is going to cover everyone's needs, *PenPal*'s version should cover most requirements.

The Graph-It module is very useful for creating graphs from the data of the Calc portion of *PenPal*. Users are given seven options of the type of graph to plot. As mentioned earlier, possible selections are a pie graph, horizontal or vertical line dot or bar graphs. Users can then add more than five styles of text as labels, print the graphs out using standard or double-width modes, or save the graph in a binary format and later modify it with any of the several commercially available graphics programs. *PenPal* comes

with six printer drivers: Epson, Gemini, CGP-220, the DMP series, LP VII and VIII, and C. Itoh 8510A. This module has many options and a review could be written on just this module alone. It is the most powerful module of *PenPal*.

The Telecom module is the last of the *PenPal* modules and is also well-designed. When at the main menu, users can choose any of the auto-log files they have created for any BBSs wanted or they can create a new auto-log file to get on a new BBS in the area. Once in Telecom, users have all the options of most terminal packages and maybe even some added features.

For instance, you can use the auto-log file you called up or you can go straight to the terminal mode. For transferring data you have three protocols from which to choose. You can use the simple mode that has no built-in checks or you can use the very popular XMODEM protocol. You also have a third choice, *PenPal*'s own protocol that uses a checksum to ensure error-free transmission. This last option can only be used if the other party also uses *PenPal*.

The more I used *PenPal* the more impressed I became. It is nice to be able to switch between the modules quickly and have all the data from the different modules compatible with each other. There are other packages available that do this, but they are not as affordable as *PenPal*. It may be true that if taken separately, any one of the modules is not outstanding, but together, *PenPal* makes a very useful package. The weakest part of *PenPal* is the 50-column screen on the Write module, but I believe the power in the Graph-It module compensates for this weakness.

The documentation is well-written and is very easy to follow. Four Star Software seems to have done it again, and I readily recommend *PenPal*.

# 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: .....

## Complete Electronic Organizer — Options Galore to get You in Order

By Robert E. Foiles

This may not be the usual way to start a review, but the "bug trapping" routine in *Complete Electronic Organizer (CEO)* is so unique that it rates being first.

Built into *CEO* is a system-monitoring subroutine that kicks in if a program error is encountered. This routine takes over the program and prints to the screen a message instructing the user that a program error has been encountered and the disk should be returned to Computerware with an explanation of what the user was doing just before the message appeared. Unfortunately, the routine also locks up the program and only a total system shutdown restores control of the computer to the operator.

In the first run-through of the program the "bug trap" snapped and the original disk with my explanation as requested was shipped back to Computerware. A few days later a reply acknowledged receipt of the disk and noted it had been sent back to the programmer for review. Six days later, the corrected disk was mailed back (very good turnaround time, considering this exchange had to take place just before Christmas and from east to west coast).

*CEO* is full of surprises, as you will learn, and most of the surprises were welcomed. Overall, *CEO* is a graphics delight. The program uses a 51 by 24 Hi-Res display for all its screens and has more options than a politician has excuses.

The system requirements are a 64K Color Computer with at least one disk drive. If more than one drive is online, then a couple of additional options can function. The program works just as well with only one drive, but the user has to swap the system disk for a data disk as called for by the option selected. Most of the program is loaded into memory at startup, but a couple of the other options are loaded from the system disk as needed.

The main menu screen shows the top of a desk with a border of either red or blue (border colors are random choice by *CEO*). Listed below the desk top, in two columns, are the program's options. Also appearing at the top of screen is the date and on the right side is a real-time clock. The user's name also appears after a "Good Day" message.

To get the main menu, the program has a start-up routine that asks the user to enter the number of disk drives in use, Baud rate for the printer, user's name, the time and date. It also provides for the formatting of the data disk needed for either a single- or double-drive system. The disks must already be initialized for use in the drives (e.g., using the `DSKINI` command) and the program establishes the "tables" where the data is stored. When *CEO* asks for the time to be entered, it expects the time to be in a 24-hour format, i.e., 5 p.m. is entered as 1700. The date is entered as `MMDDYYYY`, i.e., 01011986 for New Year's Day. If the user does not change the system configuration, the program uses the previously entered data and only asks for a new time and date on future runs.

The main menu displays the entered date and the time with the first digit, either a 'P' or 'A', followed by the hour and after a blinking colon, the minutes. The clock functions throughout all the options even if it is not visible in some

screens. Since the clock is running in the background it is possible to use the alarm feature to sound off at the user's selected time.

From within the main menu, the user is able to select any of the options by moving the box with the arrow keys so it encloses the option wanted. Tapping the `ENTER` key loads and/or executes the option within the box. Some of the options execute within the main screen and three others have separate screens. The first group of options take care of the housekeeping for the program and are called up individually or by selecting the "Other Things" option. The main menu options are:

1) Set Time allows the user to set the correct time, which might have been slowed because of extensive printer use.

2) Adjust Time provides a means of correcting the speed of the clock for accurate time.

3) Alarm Set/Reset allows the user to preselect one time for the alarm to sound. This clock program "beeps" on the hour, but the alarm sounds a tone for almost a minute or until the space bar is pressed.

4) Set Date allows the user to enter a new date after the start-up routine. While not covered in the manual, this option is necessary to move from one year to another. The program appears to have been designed to handle only one year at a time and will not move into a new year without using this option. A word of caution: If a new year is entered with the Set Date option, *CEO* overwrites the information in the data disk. The program prints to screen the month and year as entered at startup or as entered with Set Date and reads the saved data from the data disk, regardless of the year saved on that disk. The user must be sure to mark each data disk with the year on it to avoid such problems.

5) Other Things option brings up a submenu of additional options: A) Set Baud Rate allows for a change of the Baud rate if the data is not entered at startup or if the user changes his system configuration with a new printer. B) Change # of Drives allows the user to add a second drive and adjust the program to accept the second unit online (or vice versa).

As noted earlier, a couple of additional options come into play with two drives online. C) Change the Name provides for another individual's name to be entered on the "desk top" of the main menu screen. D) Return to Menu does just as the name suggests. E) Format a Data Disk turns an initialized blank disk into a data disk for the system to use. While not covered in the manual, when data or dates are to be entered into a new year, a formatted *CEO* data disk must be ready to receive the information. Thus, if a data disk had not been prepared for the new year, the user has to abort the option in use and return to the main menu to format a data disk. F) All of the Above option actually does a total start-up routine that covers all the steps needed to adapt the program to the user's configuration. G) Done for the Day is the command used to return to BASIC. It is the only safe way to exit the program without losing data. To get to this decision, the user must be in the main menu, then get into the submenu and answer "yes"; this is a somewhat cumbersome way to exit the program. It would be more convenient for the user if the Exit Routine was one of the major main menu options and maybe have the clock speed adjustment option become part of the housekeeping subroutines.

The Calculator option draws the face of a standard pocket calculator as an overlay on the main menu screen. The calculator has a border of either red or blue depending on the color of the border around the desk. The design does brighten the screen. The graphics display of the calculator has a window into which the figures are entered from the computer's keyboard. The unit operates as a four-function

calculator (add, subtract, multiply and divide). The usual keyboard keys are used to control the procedures. There is room for a nine-digit result and the digits are entered from left to right. If the results of a math operation require more than the nine spaces, the readout is given as a natural exponential of the number and no further number crunching is allowed. The CLEAR key must be tapped to start another math problem. The 'S' key may be used to change the sign of a number. To end the option and return to the main menu the '@' key is used.

The remaining main menu options have their own individual screens. For many users, the Calendar option will be the most frequently used option because of the many ways it can be set up and used.

The Calendar option draws a familiar calendar with blocks and dates. The month presented the first time is the month of the year as entered in the start-up routine (or through the Set Date suboption). The day of the month is enclosed in a set of brackets. The days of the month are selected by moving the brackets with the arrow keys. Once a specific day of the month has been selected, tapping the ENTER key brings up a new screen.

Across the top of this screen is a row of icons depicting operations that may be called up by moving the pointer under the specific icon (with the arrow keys) and tapping the ENTER key. Just below the icon line is a space for the name of the first item on the Clipboard to flash. Under that is the line that shows the name of the day of the week and the date (month, day, year) under consideration. Below that is a Special Occasion data field for a 32-character message to be logged. Below that line are 10 hourly entry fields (limited to 15 characters each) followed by a Memo field, which is also a 32-character field. When the program is first executed all these fields are empty until the user loads in appointments and saves the data. The dozen data fields are the backbone of the data entry of the calendar and are recalled from the data disk whenever that date is requested. A special feature of data entry into the Special Occasion line is that this date on the calendar has its number highlighted. Thus, when a month has any Special Occasion dates recorded, they are quickly visible.

The program automatically activates the keyboard icon upon entry into scheduling operation. The icons depict the operation they support and the user can always tell which icon is operational because it is in inverse image. Because the keyboard option is active upon arriving at the selected date, the user can move the highlighted box down the appointment time lines with the arrow key to the selected time. The ENTER key is pressed and a blinking cursor appears at the head of the highlighted line allowing data to be typed in. When finished (15-space limit), the ENTER key is pressed again to hold the data in place and the user can move to another time line. To save the data entered for that day, the user moves the pointer to the disk icon and presses ENTER. A new submenu pops up.

The user now must choose to save the date, save the data to another date or both, return to the day for more work, or return to the calendar and not save the data. An option is selected by moving the pointer in front of the option wanted and pressing ENTER to execute. With two disk drives online, the program is supposed to be able to transfer a Special Occasion date forward to another year to save typing in all those items. If this selection is attempted, a screen message shows up telling the user to insert a blank data disk in Drive 0 to receive the data. However, the program reads that disk, reports "disk not blank" and aborts the transfer. The same message appeared with a blank disk (not a formatted data disk).

There are six additional icons to choose from while on the scheduling screen. The Help icon can be invoked and when the pointer is stopped under any of the other icons, pressing the ENTER key produces a Help screen of specific instructions for that icon. A handy feature when you don't want to go searching for the manual to look up something.

The Scissors icon is used to cut a specific data field from the display screen. Once cut, the data field can be pasted to another field, tacked onto the Clipboard or even put in the "Trash Can." The cut and paste operation helps move data around within the same day without having to retype it, and by posting the data to the Clipboard, the data can be transferred to another day within the month or to another month of the same year. The Clipboard can hold up to nine data fields in storage and the actual number of items stored are listed on the Clipboard outline. The top entry in the Clipboard file is flashed on the Clipboard data line just under the row of icons each time an icon is accessed as a reminder. If several items are posted on the Clipboard it may be necessary to remove the top items (by selecting the Scissors and the Trash Can icons) to get to the correct item to transfer. It takes some work shifting between icons to get the job done, but once familiar with what comes first, it really becomes fun to switch data around with just a couple of keystrokes.

Another icon is the Duplicator. This icon allows for a duplicate copy of a data field to be created. For example, an item placed on Clipboard might be copied with Duplicator and moved to more than one location.

The last icon is the Printer, which produces reports on a printer. Since no graphics symbols are used in the printouts, there are no special restrictions on the types of printers that will work with *CEO*. *CEO* produced preformatted reports on both an Okidata 82 and a DMP-200 printer without problems.

The Printer submenu allows for a printout of just the selected day being worked on, a printout for a week starting with the selected date, prints a monthly calendar (with the Special Occasion dates set off by asterisks) or prints out a listing of days by "key word." A search for a key word such as "birthday" or "doctor," between specific dates, produces a printout of those days. The key word to be used in the search must be exact or it will not be found. Again, there is a little extra effort by the programmer to set up a check in the system to see if a printer is online or not.

Maybe the most useful report to be printed is the single page "week-at-a-glance" printout. Each day is printed with its day of the week, date and year and two columns of "Today's Schedule." To exit the Calendar option and return to the main menu, the 'Q' key is tapped.

The Phone Directory option also has its own screen and commands and handles up to 192 names and phone numbers in a fast and friendly way. The user is permitted to select one of six suboptions by moving the pointer to the proper selection and pressing ENTER. To add a person to the file, the Add Names suboption is selected and the user is presented a place to enter a name (up to 32 characters); by tapping ENTER the cursor moves to the phone line awaiting the number (up to 14 characters). Pressing ENTER again offers another sequence. Pressing ENTER with no entry returns the user to the menu.

The manual suggests that entries be made with the last name first so when the program automatically sorts the names alphabetically they are in acceptable order. It is interesting that the Find routine locates a name in upper- or lowercase or presents name(s) and number(s) of all that group if only one letter is entered (i.e., just a 'v'). Pressing ENTER lists the total file.

One small quirk in this section is when the Change of Name suboption is used, *CEO* writes the old name to the screen and leaves it there even after the new name has been committed to the file. The screen stays cluttered until the user goes back to the main menu.

The Note Keeper rounds out the complete organizing functions of *CEO*. Six of the 15 pages of the manual are devoted to suboptions of this section.

The data handling of this "free-form file drawer" is supposed to be made easier by the use of single letter codes. However, among the suboption codes is the use of the same letter code to accomplish different missions — confusing to say the least. Due to what must be a typing error, the manual also lists a key to be used as a command, but the Color Computer keyboard does not include such a key.

The manual states that up to 479 records can be stored on the data disk, depending on how long each record might be. An individual record may be as long as 5,400 characters and its title must fit into a maximum 48-character line. The record titles are stored in an index file that can be called up for viewing. An individual record can be called up by title directly, and any record can be edited, have lines added or deleted, or be printed (a line at a time or total record).

There are three different "search" routines in the command list. Two of the routines are used to locate a record by finding a key word within the title of the record. The GET and "?" command worked. However, all too frequently the only response with Find (which was to locate a key word in a record) was a screen message "<E> Sorry No Match."

The data entry into a record is not very user friendly in that only 230 characters are all the program accepts at a time. It is frustrating to be typing along from a source and realize the program had stopped accepting data a couple lines back. However, pressing the ENTER key moves the cursor down to an empty line and more data can be entered. The manual does not explain or warn the user of this little quirk. Also, trying to free up space by removing some records proved to be less than complete. While the record was gone from the disk, the title of the record remained in the index.

On the bright side, the programmer created several special handling procedures that are great. The user can toggle the scroll rate from fast (default) to slow. The slow scroll of the data is smooth as it moves up/down the screen. There is also a disk housekeeper routine that can be called. The Organizer command packs the disk to make the best use of the space. The manual notes that the system automatically invokes this command if the user tries to save a record and there is not enough room for it. The program provides a fast means to check disk space available and storage space in memory. The report gives the size of the record in memory (in bytes), the remaining amount of memory, the number of free records left and the amount of free disk space remaining.

The last option is the Memo Pad, which allows the user to create *one* record (up to 5,400 characters) as the only entry in that file. This record can be called to screen or printed out. If a new memo is entered, it replaces the former message.

*CEO* has many bells and whistles that work and are not just for show. However, the database portion of the program would not be my choice as a tool to use often. If someone wants an excellent appointment scheduling program, *CEO* is worth considering. The Calculator option may be useful to have online for those who might want to keep *CEO* up and running for most of the day and need a fast math job done. On balance, the appointment scheduling portion

carries the day for *CEO*.

The manual, 15 half-pages in length, covers most of the functions in detail. There are some omissions and typos that make the manual better than some, but with room for improvement.

## Hardware Review

### ACCESS09 BBS

Supplied by  
Paris Radio

By Kevin

Many of our users will be aware that over the past months we have suffered a series of trials and tribulations in the operation of our Bulletin Board System. This has led to our testing of PARIS RADIO's Bulletin Board Program known as ACCESS09. Happily we can report that we are having infinite more success in the operation of the program and the reliability of the system and have decided transfer all of our files across to the new system. But let's tell you a little more about the package supplied by Jacky.

ACCESS09 comes as a complete package ready for you to install on your OS9 system. An auto answer modem is needed in addition to the normal OS9 hardware. Included is the source code for the entire package and a complete instruction manual, very clearly and precisely written.

Essentially the system is operated in two parts. Firstly the ACCESS09 program itself, completely menu driven and self explanatory throughout. Error trapping and system protection has thus far proved to be very good. The best way to see how the program interfaces with the user is to phone COCOLINK (075) 326 370 and log in as a visitor.

Secondly, and this is the real beauty of the system, is the system operators maintenance program. Complete file handling and system maintenance is supported again using the familiar menu driven techniques used in the BBS program proper. Individual user records can be updated easily. A history file of user access and errors encountered is maintained and easily read or printed.

I believe that finally we have found ourselves a good BBS that will be capable of doing all that we want it to do. Anyone interested in establishing their own system would be advised to check ACCESS09 out.

ACCESS09 is available from:  
PARIS RADIO ELECTRONICS  
161 Bunnerong Rd.,  
Kingsford, N.S.W. 2032  
Ph. 344 9111

#### One-Liner Contest Winner . . .

If you enjoy word games, *Anagram* may help you with the guesswork. It takes any word you enter and randomly scrambles it. This means you don't have to decide how to best scramble the word.

#### The listing:

```
Ø INPUT"WORD";A$:L=LEN(A$):FORJ=
1TOL:R=RND(L):T$=MID$(A$,J,1):MI
D$(A$,J,1)=MID$(A$,R,1):MID$(A$,
R,1)=T$:NEXTJ:PRINTA$
```

Bruce Wulfsberg, M.D.  
Moorestown, NJ

# THE COCO



Program by Bruce K. Bell, O.D.

**AS** the prison guard slams the door in your face and walks away laughing, you wonder how in the world a perfectly organized vacation has turned into a mysterious prison sentence. With the use of the Rubix Vacation Planner, a program designed specifically for your CoCo, you outlined and booked what promised to be a simply marvelous vacation. It was almost as if the entire trip would be controlled by the CoCo. But since being arrested and found guilty of a crime you could not possibly have committed, you have to wonder if your program was somehow sabotaged.

Not long after reaching your destination, the news of the disappearance of the infamous African jewel was made public. The most magnificent and mystifying of all stones was on a touring exhibition, but was stolen before making it to the museum. However, the local police announced they had a suspect in mind and felt they would have the dastardly criminal behind bars before the end of the day.

Sacrificing your curious preoccupation with the African jewel incident to resume the vacation, you packed your backpack for an afternoon of sightseeing and some off-the-beaten-path ex-

ploring. However, before making your exit, a troop of uniformed officers bolted through the door of the beach hut, handcuffed you and provided an escort to the city's Hall of Justice. Although never resisting, you were literally dragged into the courtroom. With no booking, no processing and without making one comment in your defense, you were found guilty of stealing the African jewel. The result of the mock trial: a sentencing of imprisonment for an undetermined amount of time.

You must have been framed. Could the *Rubix Vacation Planner* have something to do with this mess? You know you could prove your innocence in a fair trial, but not hopelessly stuck behind bars. Your imagination ponders escape. Fat chance! But suddenly, a nervous-looking guard approaches delivering a tray of food. The meal looks less than appetizing. As you lift the napkin to wipe the perspiration from your brow, a note falls to the floor. The guard dashes away. You can disregard the note and dream of being rescued, or pay heed to it, which could lead you on an Adventure through another dimension known as . . . *The CoCo Zone*.

#### Loading Instructions

*The CoCo Zone* Adventure requires 64K Extended Color BASIC and consists of three BASIC programs: *CoCo Draw*,

*Boot* and *CoCo Zone*. Some of *CoCo Draw*'s subroutines (lines 1-25) are taken from Fred Scerbo's "Wishing Well" program, *Seven More PMODE4 Colors* (THE RAINBOW, January 1985, Page 32), and were used by express permission. The *CoCo Draw* program creates and saves the 10 graphics screens that are used in the game. The screens are loaded in the upper 32K of a 64K Color Computer. *Boot* loads the machine language routines and the 10 graphics screens created by *CoCo Draw*. *CoCo Zone* is the actual game, which is also loaded and executed by *Boot*.

Carefully type in *Boot* (Listing 1) and save a copy on either cassette or disk. If you are saving on cassette, the disk controller must be unplugged (if you plan to also RUN *CoCo Zone* with the controller unplugged) due to the different address of graphics video memory on a non-disk system. Then, type in *CoCo Draw* (Listing 2) and save a copy. If you are saving on cassette, you may wish to CSAVE "COCODRAW" on a separate tape; it may be stored away since it is not needed for actual game play. You must also return the same cassette — which you saved earlier as *Boot* (Listing 1) — to the cassette recorder. Returning the same cassette enables you to save the 10 graphics screens created by *CoCo Draw*, following the *Boot* program.

Now, RUN *CoCo Draw* and a colored

screen appears. If it is red, press ENTER to continue. If the screen is blue, press Reset and RUN the program again until the screen is red. After the color test, you are asked if you wish to observe the graphics screens while they are being drawn. Though this will spoil some of the Adventure's mystery, it allows you to spot any obvious typing mistakes before playing the game. The choice is up to you, but the process is quite time-consuming, especially for cassette. If you choose not to see the scenes being created, a message informs you of the program's progress.

After *CoCo Draw's* creating and saving process is complete your cassette or disk should contain 11 files: the *Boot* program and the graphics screens *Zone 0* through *Zone 9*. It is particularly important that the 11 programs are saved on the cassette in this order. The final step of game preparation is to type in and save the actual *CoCo Zone* game (Listing 3) following *Zone 9* on your cassette or disk.

You are now ready to play *The CoCo Zone Adventure*. Simply CLOAD or LOAD"BOOT" and RUN. However, on cassette systems, after running *Boot* you must also ENTER RUN at the OK prompt. The color test appears and game play will begin.

### 32K Disk Modification

*The CoCo Zone* uses bank switching for storage and retrieval of many of the game's graphics. However, in a 32K computer, the upper 32K of memory is not available, therefore, the program cannot be loaded in its entirety. In order to reserve enough memory, save (by

running *CoCo Draw*) *Zone 0* through *Zone 9* on your game disk and make the following changes to listings 1 and 3.

On the *Boot* program, Listing 1:

- DELETE lines 6-9, 12-13 and 30-31.
- Replace Line 5 with 5 FOR K=1 TO 2000:NEXT
- Replace EXEC32714 in Line 32 with LOADM"ZONE 9.PIC"

On *CoCo Zone*, Listing 3:

- Replace EXEC32714 in Line 138 with GOSUB187:LOADM"ZONE"+STR\$(X-1)+"PIC":EXEC32211
- Replace POKE491,10:EXEC32714 in lines 179 and 184 with GOSUB187:LOADM"ZONE 9.PIC":EXEC32211

*CoCo Zone's* graphics screens will now be called from the disk drive individually, allowing it to be played on a 32K disk system.

### Hints on Playing the Adventure

Since *CoCo Zone* contains a true directional map, tracing your moves can be extremely helpful. One-letter directional commands can be used (N, S, E and W) and you can always LOOK in any of the four directions (e.g., LOOK NORTH). Two-word commands are used: a verb followed by a noun. The first three letters of each verb and the first four letters of each noun can be used as abbreviation if desired to speed game play.

The program keeps track of the number of command entries made. Each command is referred to as a move,

and by typing SCORE, the number of moves made at that point is revealed. However, entering SCORE does not count as a move.

In case the BREAK key is inadvertently pressed during game play, the Hi-Res screen will not be affected. By entering GOTO 4, the screen is cleared and play may resume by typing LOOK at the WHAT NEXT? prompt. You may also quit playing at any point by entering QUIT. Then, when prompted with ARE YOU SURE?, enter YES, or just 'Y'. The program performs a cold start and will be erased from memory.

You may save and retrieve your game at any stage if desired. To save a game, enter SAVE and you are asked if the save is to disk or cassette. Entering 'D' or 'C' initiates a disk or cassette save, respectively. Entering 'A' aborts the procedure. Before responding, however, prepare your disk or cassette for saving.

To load a saved game, enter LOAD. Again you are prompted for disk or cassette (or abort). Prepare your disk or cassette and press the appropriate letter, 'D', 'C' or 'A'. Press ENTER and after loading, the game resumes at the point at which you saved the Adventure.

The challenge of *CoCo Zone* is not only to prove your innocence, but to do so in as few moves as possible. But you must remember, once you enter the *CoCo Zone*, trying to exit could result in death. Unless, of course, you are able to make the right moves at precisely the right time. However, as Dr. Bell might also remind you, don't bury yourself in the *CoCo Zone* without an escape plan!

9.....	60
18.....	74
25.....	159
END.....	141

Listing 1: BOOT

```

Ø 'COCO ZONE BOOT 1.Ø, (C) 1985
BRUCE BELL
1 BMODE4,1:POKE179,2:PCLS:SCREEN
1,1:IFINKEY$=""THEN1
2 CLEARØØ,3221Ø:CLS:PRINT"COCO
ZONE (BOOT) 1.Ø","(C) 1985 BRUCE
K. BELL":PRINT:INPUT"CASSETTE O
R DISK":CD$:IFCD$<>"C"ANDCD$<>"D
"THEN2
3 CLS:PRINTSTRING$(32,124);
4 PRINT"YOU ARE SEATED IN A DING
Y COURT-ROOM, ENDURING HEAT SO S
TIPLING THAT NOT EVEN THE FLIES
HAVE BOTHERED TO DROP IN. YOU
HAVE BEEN TRIED FOR THEFT OF
THE PRICELESS AFRICAN JEWEL
FOR","WHICH YOU KNOW YOU ARE INN
OCENT.

```

```

5 FORK=ØTO53:READD$:POKE32714+K,
VAL("&H"+D$):NEXT
6 DATA34,37,7F,1,EA,CC,8Ø,1,1Ø,9
E,BC,BE,1,EA,3Ø,1F,27,5,C3,B,FF,
2Ø,F7,1F,1,C3,B
7 DATAFF,FD,7F,FE,A6,AØ,1A,5Ø,7F
,FF,DE,A7,8Ø,7F,FF,DE,1C,AF,BC,7
F,FE,26,ED,35,B7,Ø,Ø
8 FORK=ØTO4:IFCD$="C"THENCLOADM"
ZONE"+STR$(K)ELSELOADM"ZONE"+STR
$(K)+"PIC"
9 POKE491,K+1:EXEC32714:NEXT
1Ø PRINT" A MILDLY PLUMP JURY
FOREMAN SMILES AT A PREOCCUPIED
JUDGE ASHE STATES A VERDICT OF
GUILTY! THE JUDGE WINKS AT AN O
VERCONFI-DENT DISTRICT ATTORNEY
AS HE PRONOUNCES SENTENCE. A
SENTENCE"
11 PRINT"THAT NEITHER HE NOR YOU
FULLY COMPREHEND. ONE WHOSE O
NLY LIMITS ARE THE BARRIERS
OF THE IMAGINATION. FOR YOU HA
VE JUST CROSSED OVER INTO THE..
."
12 FORK=5TO9:IFCD$="C"THENCLOADM
"ZONE"+STR$(K)ELSELOADM"ZONE"+ST
R$(K)+"PIC"
13 POKE491,K+1:EXEC32714:NEXT
14 FORK=ØTO5Ø2:READD$:POKE32211+
K,VAL("&H"+D$):NEXT

```

```

15 DATA3Ø,8D,Ø,19,BF,1,68,3Ø,8D,
Ø,76,BF,1,6B,86,7E,B7,1,67,B7,1,
6A,86,39,A7,8C,ES,39,Ø,34,37,D6,
6F,26,5C
16 DATA1F,2,DC,88,C4,EØ,E7,8C,FØ
,86,C,3D,DB,89,EØ,8C,E8,D3,BC,C3
,Ø,6Ø,9E,88,8C,5,Ø,25,3,C3,C,Ø,1
E,2,81
17 DATAFF,27,16,81,D,27,1A,81,8,
27,16,81,2Ø,27,12,81,2F,2F,A,81,
5B,2C,6,2Ø,A,86,2E,2Ø,6,86,5B,2Ø
,2,86,2F
18 DATA8E,7E,52,8Ø,2D,3Ø,8,4A,26
,FB,C6,8,A6,8Ø,A7,A4,31,A8,2Ø,5A
,26,F6,35,B7,34,37,86,FF,2Ø,9C,F
F,DF,D7,D5,D5
19 DATAD7,DF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,C7,BB,BB,BB,BB,BB,C7,FF,
EF,CF,EF,EF,EF,EF,83,FF,C7,BB,FB
,E7,DF,BF,83,FF
2Ø DATAC7,BB,FB,E7,FB,BB,C7,FF,B
B,BB,BB,81,FB,FB,FB,FF,83,BF,C7,
FB,FB,BB,C7,FF,C7,BF,BF,A7,9B,BB
,C7,FF,83,FB,F7
21 DATAEF,DF,BF,BF,FF,C7,BB,BB,C
7,BB,BB,C7,FF,C7,BB,BB,C3,FB,FB,
C7,FF,FF,BF,BF,FF,FF,BF,BF,FF,FF
,FF,FF,FF,EF,EF
22 DATADF,BF,F7,EF,DF,BF,DF,EF,F
7,FF,FF,FF,FF,83,FF,FF,FF,FF,BF,
DF,EF,F7,EF,DF,BF,FF,C7,BB,FB,E7
,EF,FF,EF,FF,EF

```

```

23 DATA CF,AF,EF,EB,E7,EF,FF,EF,D
7,BB,83,BB,BB,BB,FF,87,BB,BB,87,
BB,BB,87,FF,C7,BB,BF,BF,BF,BB,C7
,FF,87,BB,BB,BB
24 DATABB,BB,87,FF,83,BF,BF,87,B
F,BF,83,FF,83,BF,BF,87,BF,BF,BF,
FF,C3,BF,BF,B3,BB,BB,C3,FF,BB,BB
,BB,83,BB,BB,BB
25 DATA FF,83,EF,EF,EF,EF,EF,83,F
F,FB,FB,FB,FB,FB,FB,C7,FF,BB,B7,
AF,9F,AF,B7,BB,FF,BF,BF,BF,BF,BF
,BF,83,FF,BB,93
26 DATA AB,BB,BB,BB,BB,FF,BB,9B,A
B,B3,BB,BB,BB,FF,C7,BB,BB,BB,BB,
BB,C7,FF,87,BB,BB,87,BF,BF,BF,FF
,C7,BB,BB,BB,AB
27 DATA B7,CB,FF,87,BB,BB,87,AF,B
7,BB,FF,C7,BB,BF,C7,FB,BB,C7,FF,
83,EF,EF,EF,EF,EF,EF,FF,BB,BB,
BB,BB,BB,C7,FF
28 DATABB,BB,BB,BB,BB,D7,EF,FF,B
B,BB,BB,AB,93,BB,FF,BB,BB,D7,
EF,D7,BB,BB,FF,BB,BB,D7,EF,EF,EF
,EF,FF,83,FB,F7
29 DATA EF,DF,BF,83,FF,FF,FF,FF,F
F,FF,FF,FF,FF
30 FORK=0TO13:READD$:POKE32745+K
,VAL("&H"+D$):NEXT
31 DATA 1A,50,7F,FF,DF,A6,80,7F,F
F,DE,1C,AF,A7,A0
32 PMODE4,1:PCLSD:FORK=1TO100:PS
ET(RND(255),RND(96)+95,5):NEXT:P
OKE491,10:EXEC32714:SCREEN1,1
33 POKE492,PEEK(360):POKE493,PEE
K(361):POKE494,PEEK(363):POKE495
,PEEK(364):"store unmodified ram
vector
34 IFCD$="C"THENLOAD"COCOZONE"E
LSELOAD"COCOZONE",R

```

15	.....	170
24	.....	132
33	.....	216
39	.....	3
45	.....	175
50	.....	7
56	.....	140
60	.....	248
64	.....	197
END	.....	12

**Listing 2: COCODRAW**

```

0 'COCO ZONE (DRAW) 1.0, (C) 198
5 BRUCE K. BELL
1 '*****
2 '* SEVEN MORE PMODE4 COLORS *
3 '* BY FRED B. SCERBO *
4 '* 149 BARBOUR ST. N.ADAMS.MA*
5 '* COPYRIGHT (C) 1984 *
6 '*****
7 CLEAR1000:R=3:B=2
8 PMODE4,1:PCLSD:SCREEN1,1:PMODE
3:PCLSD
9 IFINKEY$=CHR$(13)THEN1ELSE9
10 'START COLOR SET
11 CLSD:PMODE4,1:PCLSD:SCREEN1,1
:DIM Y(3),B(3),G(3),S(3),P(3),L(
3),V(3):LINE(32,0)-(48,5),PSET,B
F
12 FORX=31TO47STEP4:PSET(X,0):
PSET(X+2,1,0):PSET(X+1,4,0):PSET
(X+3,5,0):NEXT
13 FORX=32TO47STEP8:PSET(X,8):PS
ET(X+4,9):LINE(X,12)-(X+1,12),PS
ET:LINE(X+4,12)-(X+5,12),PSET:LI
NE(X+2,13)-(X+3,13),PSET:LINE(X+
6,13)-(X+7,13),PSET
14 PSET(X,16):PSET(X+1,17):PSET(
X+4,16):PSET(X+5,17):PSET(X+1,20

```

```

):PSET(X+5,21):NEXTX:PMODE3:COLO
R2,3:LINE(32,24)-(48,24),PSET:LI
NE(32,25)-(48,25),PRESET
15 PMODE4:GET(32,0)-(47,1),Y,G:G
ET(32,4)-(47,5),B,G:GET(32,8)-(4
7,9),G,G:GET(32,12)-(47,13),S,G:
GET(32,16)-(47,17),P,G:GET(32,20
)-(47,21),L,G:GET(32,24)-(47,25
),V,G
16 GOTO26:'PAINTING ROUTINES
17 LC=VAL(MID$(PT$,2,3)):TC=VAL(
MID$(PT$,6,3)):RC=VAL(MID$(PT$,1
0,3)):BC=VAL(MID$(PT$,14,3))
18 XX$=LEFT$(PT$,1):IFXX$="Y"THE
N19ELSEIFXX$="B"THEN20ELSEIFXX$=
"G"THEN21ELSEIFXX$="S"THEN22ELSE
IFXX$="P"THEN23ELSEIFXX$="L"THEN
24ELSEIFXX$="V"THEN25ELSERETURN
19 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),Y,OR:NEXTZZ,YY:RETURN
20 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),B,OR:NEXTZZ,YY:RETURN
21 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),G,OR:NEXTZZ,YY:RETURN
22 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),S,OR:NEXTZZ,YY:RETURN
23 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),P,OR:NEXTZZ,YY:RETURN
24 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),L,OR:NEXTZZ,YY:RETURN
25 FORY=TC TO BC STEP2:FORZZ=LC
TO RC STEP16:PUT(ZZ,YY)-(ZZ+15,
YY+1),V,OR:NEXTZZ,YY:RETURN
26 CLS:PRINT"COCO ZONE DRAW 1.0"
,"(C) 1985 BRUCE K. BELL",,,:I
NPUT"CASSETTE OR DISK OPERATION"
:QS:IFQS<>"C"ANDQS<>"D"THEN26
27 PRINT:INPUT"DO YOU WISH TO VI
EW THE PICTURES THEY ARE DRAWN
(Y/N)":IS:IFI$="Y"THENSREEN1,1
28 FORA=0TO9:IFI$<>"Y"THENPRINT@
352,"DRAWING PICTURE #"+A
29 PCLSD:DRAW"BM0,0C0R255D96L255
U96":ONA+1GOSUB32,37,40,45,50,55
,60,65,68,71:F$="ZONE"+STR$(A):I
FQ$="C"THEN30ELSEVERIFYON:SAVEMF
$+"PIC",3584,6656,380:GOTO31
30 PK=PEEK(188)*256+PEEK(189):CS
AVEM F$,PK,PK+3072,380
31 NEXTA:PRINT@384,"JOB COMPLETE
D":END
32 DRAW"BM255,96C0H10L200U20R100
BR397ND20F6BF2F2U20G2BG2G6BE10L
20NU90G6NU96L154NU96BE10NE6U50R6
ND44R54D44NL54D6L60BR74NE6U50R6N
D44R54D44NL54D6L60":PAINT(96,4),
0,0:PAINT(170,4),0,0
33 PT$="Y092,003-148,046":GOSUB1
7:PT$="Y166,003-220,046":GOSUB17
:DRAW"BL14NU50BR74NU50BR10NU50BL
70BD4BL84L20G10D8L10NU90G22":POK
E178,2:PAINT(80,4),0,0:POKE178,1:
PAINT(128,70),0,0
34 FORK=60TO210STEP30:LINE(K,70)
-(K+20,80),PRESET,BF:CIRCLE(K+10
,75),2,5:NEXT:PAINT(128,94),0,0:
FORX=0TO2:POKE178,K:FORX=1TO8:CI
RCLE(8,42),(3-K)*8-X,,.8:NEXTX,K
:PAINT(2,2),0,0:POKE178,124:PAINT
(254,4),0,0:PAINT(40,4),0,0
35 G$="C0U3LD3LU3LD3LU6ED4RU6ED7
RU16GD2D2F2U8RD14EU22ED20EU30LD10
":FORK=96TO136STEP8:X=K+74:DRAW"
BM=K;48XG$:BM=X;48XG$:"NEXT
36 RETURN
37 LINE(15,0)-(239,75),PSET,BF:P
T$="Y016,000-239,073":GOSUB17:FO
RX=24TO239STEP24:FORY=10TO40STEP
30:POKE178,2:LINE(X,Y)-(X+16,Y+2
2),PSET,BF:CIRCLE(X+4,Y+10),2,0:
NEXTY,X

```

```

38 DRAW"BM239,0C0D75F15BL255E15"
:CIRCLE(128,95),50,0,.2,.5,1:POK
E178,1:PAINT(128,90),0,0:PAINT(50
,90),0,0:PT$="S000,075-255,095":
GOSUB17:DRAW"G15BR255H15":POKE17
8,212:PAINT(2,2),0,0:PAINT(250,2)
,0,0
39 RETURN
40 DRAW"BM255,83C0M-34,-25NU58L2
8NU58GBNU66L66U8NU58L84NU58G4BM1
26,4R52D5L26NU52L26U52BD17BR7U4
R4U4R4D4R4D4L4D4L4U4L4R26U4R4U4
R4D4R4D4L4D4L4U4L4":POKE178,2:PA
INT(139,18),0,0:PAINT(165,18),0,0:
PAINT(190,2),0,0:CIRCLE(146,30),2
,0:CIRCLE(158,30),2,0
41 PAINT(100,4),0,0:PAINT(200,4)
,0,0:PT$="S022,001-110,052":GOSU
B17:PT$="S194,001-216,052":GOSUB
17:FORK=5TO40STEP5:CIRCLE(150,80
),K,0,.2:NEXT:PT$="P111,072-190,
088":GOSUB17
42 CIRCLE(200,66),200,0,.3,.5,1:CI
RCLE(200,76),200,0,.3,1,.5:CIRCLE(
40,71),4,0,1,3,.25,.75:CIRCLE(0,
71),4,1,3,.75,.25:DRAW"BM40,66C
0R6FND10M-14,+29R6M+8,-18D18L2NU
12LU10"
43 LINE(70,10)-(95,40),PSET,BF:C
IRCLE(82,22),9,5,1,.1,.9:CIRCLE(
88,22),5,5,1,.8,.2:PAINT(82,25),
5,5:DRAW"BM75,22C5G4D2E6C0BR11BU
2U4BD8D4C5BD5L14BD2R10BF2L12"
44 POKE178,1:PAINT(2,94),0,0:PAIN
T(39,94),0,0:POKE178,104:PAINT(2,
2),0,0:PAINT(254,2),0,0:RETURN
45 DRAW"BM255,70C0M-84,-30NU400":
PAINT(200,20),0,0:PT$="S171,000-
256,70":GOSUB17:DRAW"LI2R2BF18D6
L2D28L4NU28H2U26L8D10L4NU10H2U8R
18L126R2D26F2NU28R4U28L8U6R40BR3
R85BL20H24D6NF24F2D26F2NU26R4U2
2H8U6R12NU14D6F12R100U18NM+30,+2
4U08H12L88M-11,+6"
46 PAINT(128,40),0,0:PT$="Y038,0
14-134,051":GOSUB17:DRAW"M+11,-6
R88F12ND26L20NH12L68NH12M-11,+6N
H12D20H12BL14L24BR40BU08F6UH6BR3
0BD6R8DL8BR28R8UL8BR28R8DL8":PAI
NT(136,20),0,0
47 POKE178,2:PAINT(128,55),0,0:PO
KE178,1:PAINT(2,90),0,0:DRAW"BM18
0,4C5D12M+18,+4U16L18BD20M+18,+4
D14M-18,-6U12BM+28,+6M+22,+4D20M
-22,-8U16BU8M+22,+4U22L2D18BM+3
2,+6U24R16D28M-14,-4BD8M+14,+4D2
2M-15,-4U22":PAINT(128,55),0,0
48 POKE178,1:PAINT(2,90),0,0:DRAW
"BM180,4C5D12M+18,+4U16BM20,0C0D
25R16BE12BR8U12":PAINT(22,2),0,0
:DRAW"C5BG2L4BD2R2BF2L4BD2R3BF2L
5BG8BL2H4G8NL2E8U6NF6NG6U2LH2UER
3FDG2":POKE178,201:PAINT(2,2),0,0
:PAINT(170,2),0,0
49 RETURN

```







```

;04;1;2;3;4;5;6;7;8;9":NEXT:PMOD
E4,1:GOTO178
20 IFR=14THENR=31:GOTO14ELSEIFR>
1THENIFO(1)<>R ANDO(1)<145THENPR
INT"THE NOTE FROM GEORGE WAS FOU
ND AND YOU HAVE BOTH BEEN CAUGH
T":GOTO178
21 GOTO4
22 PRINT@448,"YOU CANNOT GO THAT
WAY":GOTO4
23 AV$=LEFT$(A$+" ",3):AN$=MID
$(A$+" ",INSTR(1,A$," ") +1,4)
:V=INT((2+INSTR(1,V$,AV$))/3):N=
INT((3+INSTR(1,N$,AN$))/4)
24 ONV GOTO25,25,39,39,46,49,49,
51,61,62,68,75,76,82,84,86,88,90
,92,99,103,104,108,113,118,121,1
21,124,126,128,130,133:PRINT"SOR
RY. I DO NOT UNDERSTAND.":GOTO4
25 IFP(5)=1ANDR=60THEN14:GOTO4EL
SED=INSTR(1,"NORTSOUTEASTWEST",A
N$):IFD>0THENU=(D+3)/4:ONU GOTO1
0,11,12,13ELSEIFAN$="LOOK"THEN14
:GOTO4ELSEIFN=15THEN3ELSEIFO(N)
<150THENPRINT"YOU DO NOT HAVE IT
.":GOTO4
26 IFR=10RN=12THENPRINT"HERE IS
WRITING ON IT"ELSEIFN=2THENPRIN
T"IT IS TORN. IT IS A LIST OF SO
MESORT.":MID$(N$,5,4)="LIST":O$(
2)="LIST"ELSEIFN=4THENPRINT"RIFL
E SHELLS"ELSEIFN=5THENPRINT"OXYG
EN MASK"ELSEIFN=6THENPRINT"OXYGE
N BOTTLE MARKED FULL"
27 IFR=9THENPRINT"HAND OPERATED"
ELSEIFN=11THENPRINT"CONTAINS BAT
TERIES"ELSEIFN=13THENPRINT"SOFT
LENS"ELSEIFN=14THENPRINT"VERY SH
ARP"ELSEIFN=15THENPRINT"VERY STU
RDY IN APPEARANCE"
28 IFR=8THENIFP(0)=0THENPRINT"A
BOOK OF MATCHES WITH ONLY ONE MA
TCH IN IT"ELSEPRINT"COMPLETELY E
MPTY"
29 IFR=3THENIFP(1)=0THENPRINT"TH
E SHELL CHAMBER IS EMPTY"ELSEPRI
NT"THERE IS A SHELL IN THE CHAMB
ER"
30 GOTO37
31 IFO(N)<>R THENPRINT"THAT IS N
OT HERE":GOTO4ELSEIFN=16ORN=30TH
ENPRINT"HERE IS WRITING ON IT"EL
SEIFN=17AND(3)<0THENPRINT"ALAR
M PROTECTED. HOWEVER YOU SEEOE
RIFLE AGAINST THE WALL OUTS
IDE THE GUNCASE.":O(3)=29
32 IFR=19AND(4)<0THENPRINT"FULL
OF AMMO AND APPEARS VERY HEAV
Y":O(4)=R ELSEIFN=21THENPRINT"A
REMBRANDT ORIGINAL I BELIEVE"EL
SIFN=25AND(9)<0AND(10)<0THENPR
INT"THE DRILL AND SHOVEL LOOK",
"INTERESTING":O(9)=R:O(10)=R
33 IFR=24THENIFP(4)=0THENPRINT"IT
IS EMPTY"ELSEIFP(5)=0THENPRINT
"IT IS OPEN"ELSEPRINT"TOTAL DARK
NESS."
34 IFR=26THENPRINT"SLOW EASY RAP
IDS"ELSEIFN=27THENPRINT"LOOKS IN
VITING"ELSEIFN=29THENPRINT"HERE
IS A SIGN ON IT":O(30)=R ELSEIF
N=31THENPRINT"SHE IS HUGE AND UG
LY. ALSO SHE IS DROOLING AS SHE
APPROACHES"ELSEIFN=35THENPRINT"
TYPE <A> I THINK."
35 IFR=23THENPOKE178,88:LINE(0,0)
-(255,95),PSET,BF:POKE178,P(14)
:LINE(100,50)-(156,95),PSET,BF:P
OKE178,0:POKE179,0:PRINT"WONDER
WHAT IS BEHIND IT?"
36 IFR=33THENPRINT"A STURDY BRAN
CH ARCHES ACROSS THE PIT OPENI
NG ABOVE"ELSEIFN=34THENR=R+10:GO
TO14ELSEIFN=37THEN180
37 IFPEEK(1440)=96THENPRINT"YOU
SEE NOTHING SPECIAL"
38 GOTO4
39 IFO(N)=150THENPRINT"YOU ALREA

```

```

DY HAVE THAT":GOTO4ELSEIFO(N)>=0
ANDO(N)<>R THENPRINT"THAT IS NOT
HERE":GOTO4ELSEIFN=37THEN4ELSE
IFN>150RO(N)<0THENPRINT"YOU CANN
OT GET THAT":GOTO4ELSEIFP(3)>6TH
ENPRINT"YOUR HANDS ARE FULL":GOT
O4
40 P(3)=P(3)+1:O(N)=150:PRINT"TA
KEN":IFN=6ANDR=50THENLINE(158,52)
-(166,43),PSET,BF ELSEIFN=5ANDR
=50THENLINE(158,40)-(170,34),PSE
T,BF
41 IFR=65THENIFO(9)=150THENDRAW"
BM110,41C5EL2ED2U7ER3EU7HL3HU5EH
EHEHEHEHEHEHC0"
42 IFR=65THENIFO(10)=150THENPOKE
178,99:LINE(222,24)-(236,70),PSE
T,BF:POKE178,0
43 IFR=100THENIFN=11THENPRINT"YO
U TRIPPED AND SPRAINED YOUR AN
KLE"
44 IFR=12THENGOTO14ELSEIFN=37THE
NPRINT"YOUR GREED HAS DEFEATED Y
OU. YOUARE NOW TRAPPED IN THE PO
OL AS GEORGE ESCAPES":GOTO178
45 GOTO4
46 IFR<080RO(N)<150THENPRINT"YOU
CANNOT DO THAT":GOTO4ELSEIFR<6
0THENPRINT"YOU HAVE WASTED YOUR
ONLY MATCH AND PROBABLY THE GAME
":P(0)=1:GOTO4
47 IFR(0)>0THENPRINT"YOUR MATCH
IS ALREADY SPENT":GOTO4ELSEP(0)=
2:PMODE2,1:PCLS5:PMODE4,1:PRINT"
A MESSAGE APPEARS ATTACHED TO
THE CASKET LID OVER HEAD. IT
SAYS..."
48 PRINT@0,"YOU FOOL. IT WAS I W
HO STOLE THEAFRICAN JEWEL. AND I
T WAS I WHO FRAMED YOU. ONLY YOU
WILL NEVER TELL BECAUSE ONLY I
KNOW YOU AREHERE AND I WILL NEVE
R TELL.", "GEORGE RUBIX":GOTO4
49 IFO(N)<>150THENPRINT"YOU DO N
OT HAVE THAT"ELSEPRINT"OK. YOU N
O LONGER HAVE IT.":O(N)=R:P(3)=P
(3)-1
50 GOTO4
51 IFR=36ANDR=36THENPRINT"USE TH
E FOUR ARROW KEYS TO", "POSITION
THE CURSOR AND PRESS ENTER TO
OPEN DESIRED CUPBOARD.":K=0:Z=2
:GOSUB134ELSE54
52 PRINT@416,STRING$(95,32):PRI
NT@416,"":POKE178,0:LINE(X,Y)-(
X+16,Y+22),PSET,BF:IFX=P(10)ANDY
=40THENX=X+5:DRAW"BM=X",60C5U6NR
6E2R6NG2D6G2NL6BM24,10":PAINT(X+
2,58),5,5:O(19)=R:PRINT"HERE IS
A SMALL BOX HERE"ELSEPRINT"NOTH
ING IMPORTANT"
53 GOTO4
54 IFR<18ANDR<50THEN58ELSEIFP(
11)=0THENPRINT"IT IS LOCKED":GOT
O4ELSEP(12)=1:LINE(154,6)-(176,5
4),PSET,BF:FORV=18TO54STEP12:DRA
W"BM154,-Y;C5ER20GL20":NEXT:IFO(
5)=-1THENO(5)=R
55 IFO(6)=-1THENO(6)=R
56 IFO(6)=R THENDRAW"BE2BR4HU5E2
U2R2D2F2D5GL4":POKE178,1:PAINT(1

```



```

62,50),,5:POKE178,0
57 IFO(5)=R THENDRAW"BM158,40C5E
6F6L12":POKE178,2:PAINT(162,38)
,5:POKE178,0:GOTO4
58 IFR<220RO(22)<52THEN60ELSEP
RINT"COMBINATION?":INPUT"1ST NU
MBER";N1:INPUT"2ND NUMBER";N2:IN
PUT"3RD NUMBER";N3:IFN1=P(15)AND
N2=P(16)ANDN3=P(17)THENPRINT@464
,"SAFE IS OPEN"ELSEPRINT@464,"IT
DID NOT WORK":GOTO4
59 LINE(139,8)-(165,36),PSET,BF:
DRAW"BM139,36C5E4U20NH4R18NE4D20
NF4L18C0":P(13)=1:IFO(7)=-1THENO
(7)=R
60 IFR=29ANDN=17THENPRINT"AN ALA
RM SOUNDS QUICKLY SIGNAL= ING TH
E GUARDS":GOTO178ELSE4
61 IFR<33ORR<135THENPRINT"WHEE
... THAT WAS FUN.":GOTO4ELSEPRI
NT"A SUCCESSFUL JUMP...":YOU HA
VE CLEARED THE PIT.":R=133:O(33)
=134:GOTO14
62 IFR(R)<>12THENPRINT"HOW IN TH
E WORLD WILL YOU DO THAT?":GO
TO4ELSEINPUT"WHAT TYPE OF STROKE
? <1> BUTTER=FLY <2> FREE STYLE
<3> DOG PADDLE":A$:A=VAL(A$)
:PRINT@416,STRING$(94,32):PRIN
T@416,"":ONA GOTO63,64,65:GOTO4
63 PRINT"YOU SHOULD GO OUT FOR T
HE OLYM= PICS":GOTO66
64 PRINT"A LIBERAL; NO DOUBT":GO
TO66
65 PRINT"HOW ABOUT SOME ALPO?"
66 PRINT"YOU MADE IT TO THE OTHE
R SIDE":IFD(R)=1THEND(R)=2ELSE(D
(R))=1
67 GOTO4
68 IFR=36ANDR=36THENFORX=24TO238
STEP24:FORV=10TO40STEP30:POKE178
,2:LINE(X,Y)-(X+16,Y+22),PSET,BF
:CIRCLE(X+4,Y+10),2,0:NEXTV,X:O(
19)=-1:IFO(4)=36THENO(4)=-1
69 IFR<180RR<50THEN71ELSEP(12)
=0:POKE178,3:LINE(154,6)-(176,54)
,PSET,BF:DRAW"BM159,21C0U4R4U4R
4D4R4D4L4D4L4U4L4":CIRCLE(158,30)
,2:POKE178,2:PAINT(165,18),,0:P
OKE178,0:IFO(5)=50THENO(5)=-1
70 IFO(6)=50THENO(6)=-1
71 IFR=24ANDP(4)=1THENPMODE2,1:P
CLS0:PMODE4,1:PRINT"OK":O(24)=0:
P(5)=1ELSE4
72 PMODE2,3:PCLS5:PMODE4,1:PRINT
@256,"AFTER A WHILE; THE CASKET
BEGINSTO MOVE...":FORK=1TO900:NE
XT:PRINT"THEN YOU HEAR A TRUCK E
NGINE...":PRINT:A$="L25501;1;2;3
;6;4;8;9;11;3;2;12;4;6;1;":FORK=
1TO50:PLAYA$:NEXT:FORK=1TO500:NE
XT
73 PRINT:PRINT"AFTER A FEW KNOCK
S AND BANGS ALLMOTION CEASES AND
YOU HEAR DIRT HITTING THE CASKE
T.":FORK=12TO0STEP-1:PLAY"V=K;"+
LEFT$(A$,21)+"V12P"+STR$(RND(10)
):NEXT
74 IFO(5)<>150RO(6)<>150THENPRI
NT@416,"SINCE YOU DO NOT HAVE TH
E OXYGENMASK OR BOTTLE; YOU HAVE
QUICKLYSUFFOCATED FROM LACK OF
OXYGEN":GOTO178
75 IFINSTR(1,"NSEW",LEFT$(AN$,1)
)>0THENA$=LEFT$(AN$,1):GOTO8ELSE
IFN=27ORN=24ORN=29THEN76ELSE PRI
NT"YOU CANNOT GO THERE AT THIS T
IME":GOTO4
76 IFR=33ANDR=134THENPRINT"GOOD
TRY BUT THE WALLS ARE TOO SLIPP
ERY."ELSEIFAN$="WALL"THENPRINT"IT
HIS IS DRIVING ME CRAZY TOO"
77 IFR(5)=0THENIFN=24ANDR=60THEN
IFP(4)=0THENPRINT"OK. YOU ARE IN
THE CASKET":P(4)=1:D(60)=0:GOTO
14ELSEPRINT"OK. YOU ARE OUT OF T
HE CASKET":P(4)=0:D(60)=1:GOTO14

```

```

78 IFN=27THENIFR=111THENPRINT"TH
AT WAS REFRESHING":R=110:GOTO14E
LSEPRINT"AH... JUST WHAT THE DOC
TOR", "ORDERED":R=111:GOTO14
79 IFN=15ANDR=134ANDP(8)=1THENPR
INT"IT WAS QUITE A STRUGGLE BUT
YOU PULLED YOURSELF OUT":R=135:O
(33)=135:GOTO14
80 IFN=29ANDR=140THENPRINT"THE F
ENCE WAS CHARGED AND NOW SOARE Y
OU":GOTO178
81 GOTO4
82 IFN=18ANDR=50THENIFP(11)=1THE
NPRINT"IT IS ALREADY UNLOCKED"EL
SEIF(7)<150THENPRINT"YOU DO NOT
HAVE THE KEY"ELSEPRINT"YOUR KEY
WORKED. IT IS UNLOCKED":P(11)=1
ELSEPRINT"YOU CANNOT DO THAT"
83 GOTO4
84 IFN=18ANDR=50THENIFP(11)=0THE
NPRINT"IT IS ALREADY LOCKED"ELSE
IF(7)<150THENPRINT"YOU DO NOT H
AVE THE KEY"ELSEPRINT"OK":P(11)=
0ELSEPRINT"YOU CANNOT DO THAT"
85 GOTO4
86 IFO(10)<150THENPRINT"YOU HAVE
NO SHOVEL"ELSEIFR<60THENPRINT"
NO EFFECT"ELSEIFP(4)=1ANDP(5)=1T
HENIFP(6)=1THENPRINT"YOU MADE A
HOLE IN THE CASSET WALL. THE L
OOSE DIRT ALLOWED YOU TO DIG TO T
HE SURFACE.":R=71:GOSUB14ELSEPRI
NT"THE SIDES ARE TOO HARD."
87 GOTO4
88 IFO(9)<150THENPRINT"YOU HAVE
NO DRILL"ELSEIFP(4)<10R(5)<10
RR<60THENPRINT"WHAT A NICE ROUN
D HOLE YOU HAVE MADE"ELSEIFP(0)=
2THENPRINT"A DRAFT OF AIR EXTING
UISHES YOURMATCH":P(6)=1ELSEPRIN
T"TOO DARK"
89 GOTO4
90 IFO(N)<150THENPRINT"YOU DO N
OT HAVE IT TO THROW"ELSEIFN<150
RR<134THENPRINT"GOOD THROW.":GO
TO49ELSEPRINT"YOU HAVE APPARENTL
Y HAD PRACTICETHE ROPE IS TIGHTL
Y HOOKED ON THE BRANCH ABOVE.":
P(8)=1:O(15)=134:P(3)=P(3)-1
91 GOTO4
92 IFN<16ANDO(N)<150THENPRINT"YO
U DO NOT HAVE THAT":GOTO4ELSEIFN
>15ANDO(N)<R THENPRINT"THAT IS
NOT HERE":GOTO4
93 IFN=1THENPMODE2,1:PCLS5:PMODE
4,1:PRINT"IT IS A NOTE FROM GEOR
GE RUBIX..A NERVOUS BUT FRIENDLY
GUARD.":PRINT0,"I KNOW YOU ARE
INNOCENT AND HAVE PLAN FOR YOU
R ESCAPE.":ELSE95
94 PRINT" GET THE OXYGEN MASK FR
OM THE INFIRMARY. THEN GO TO THE
MORGUE AND HIDE IN AN OPEN CAS
KET. YOU WILL BE BURIED BUT I W
ILL COME DIG YOU UP. YOUR CELL
DOOR IS UNLOCKED. PLEASE DESTROY
THIS NOTE.":GOTO4
95 IFN=2THENPMODE2,1:PCLS5:PMODE
4,1:PRINT"THE LIST SAYS...":P=0:
FORK=1:TOLN(V5)STEP3:PRINT0,MID
$(V5,K,3):P=P+4:NEXT:GOTO4
96 IFN=16THENIFU=3THENPRINT"ETCH
ED ON THE WINDOW IS...":PRINT010
9,"WARDEN":GOTO4ELSEPRINT"YOU A
RE NOT FACING IT"
97 IFN=12THENPRINT"IT SAYS... ME
DICALERT. I WEAR CONTACT LENSE
S.", "GEORGE RUBIX"ELSEIFN=30THEN
PRINT"IT SAYS... DANGER. HIGH VO
LTAGE."ELSEPRINT"HERE IS NOTHIN
G TO READ."
98 GOTO4
99 IFAN$<>"GUN "ANDN<3THENPRINT
"YOU CAN ONLY SHOOT GUNS":GOTO4E
LSEIF(3)<150THENPRINT"YOU HAVE
NO GUN.":GOTO4ELSEIFP(1)=0THENP
RINT"YOUR GUN IS EMPTY":GOTO4ELS

```

```

EP(1)=0
100 INPUT"AT WHAT ARE YOU AIMING
":A$:IFA$="LOCK"ANDR=69THENIFU<>
3THENPRINT"YOU ARE NOT FACING IT
":P(1)=1:GOTO4ELSEPRINT"THE PADL
OCK FALLS TO THE FLOOR AND THE
DOOR SWINGS OPEN":P(18)=1:D(69)=
3:GOTO14
101 IFA$="SPIDER"THENPRINT"YOUR
SHOT GREATLY ALARMED THE SPIDE
R WHO INJECTED YOU WITH A DEADL
Y POISON":GOTO178
102 PRINT"YOU SHOULD NOT PLAY WI
TH GUNS":GOTO4
103 INPUT"ARE YOU SURE ":A$:A$=L
EFT$(A$,1):IFA$="Y"THEN179ELSE4
104 INPUT"SAVE TO DISK OR TAPE O
R ABORT ":A$:A$=LEFT$(A$,1):IFA$
="A"THEN4ELSEIFA$="D"THENDN=1ELS
EDN=-1:PRINT@448,"POSITION TAPE.
PRESS PLAY=RECORDAND PRESS <ENT
ER>.:INPUTA$
105 PRINT0,"SAVING":GOSUB187
106 OPEN"O",DN,"ZONEDATA":PRINT#
DN,R;O$(2):PRINT#DN,MID$(N$,5,4)
:FORK=0TO18:PRINT#DN,P(K):NEXT:
FORK=1TO140:PRINT#DN,D(K):NEXT:
FORK=1TO37:PRINT#DN,O(K):NEXT:C
LOSE
107 EXEC32211:GOTO4
108 IFAN$="GUN "ORN=3THENIF(3)<
150THENPRINT"YOU HAVE NO GUN":GO
TO4ELSEIF(4)<150THENPRINT"YOU H
AVE NO AMMO":GOTO4ELSEP(1)=1:PRI
NT"CHAMBER IS LOADED":GOTO4
109 INPUT"LOAD FROM DISK OR TAPE
OR ABORT":A$:A$=LEFT$(A$,1):IFA
$="A"THEN4ELSEIFA$="D"THENDN=1EL
SEDN=-1:PRINT@448,"POSITION TAPE
.PRESS PLAY AND PRESS ENTER.":
INPUTA$
110 PRINT0,"LOADING":GOSUB187
111 OPEN"O",DN,"ZONEDATA":INPUT#
DN,R;O$(2),O$:FORK=0TO18:INPUT#D
N,P(K):NEXT:FORK=1TO140:INPUT#DN
,D(K):NEXT:FORK=1TO37:INPUT#DN,O
(K):NEXT:CLOSE:MID$(N$,5,4)=O$
112 EXEC32211:GOTO14
113 ONRND(4)GOTO114,115,116,117
114 PRINT"YOU ARE ON YOUR OWN.":
GOTO4
115 PRINT"ARE YOU SURE YOU ARE C
UT OUT FORTHIS.":GOTO4
116 PRINT"YOU ARE TRAVELING IN A
DIFFERENTDIMENSION.":GOTO4
117 PRINT"YOUR SOLUTION IS THE K
EY TO THE IMAGINATION.":GOTO4
118 X=0:PMODE2,1:PCLS5:PMODE4,1:
PRINT0,"YOU HAVE":FORK=1TO36:IF
FO(K)>149THENPRINTTAB(11)O$(K):X
=1
119 NEXT:IFX=0THENPRINTTAB(11)"N
OTHING."
120 GOTO7
121 IFO(N)<150THENPRINT"YOU DO N
OT HAVE THAT":GOTO4ELSEIFN<11TH
ENPRINT"YOU CANNOT TURN "AV$" TH
AT":GOTO4ELSEIFV=26THENIF(11)=2
00THENPRINT"OK":O(11)=150ELSEPRI
NT"ALREADY OFF"
122 IFV=27THENIF(11)=150THENPRI
NT"OK":O(11)=200ELSEPRINT"ALREA
DY ON"
123 GOTO14
124 IFN=21ANDR=52THENLINE(139,8)
-(165,36),PSET,BF:CIRCLE(152,22)
,3,5:O(22)=R ELSEPRINT"THAT HAD
NO EFFECT"
125 GOTO4
126 IFO(N)<150THENPRINT"YOU DO
NOT HAVE THAT"ELSEIFN=1THENPRINT
"GOOD IDEA. THEY WILL NEVER FIND
THE "AN$" ON YOU.":O(1)=145ELSE
PRINT"THAT COULD BE DANGEROUS. B
ESIDESYOU ARE NOT EVEN HUNGRY."
127 GOTO14
128 IFN=23ANDR=53THENPRINT"A DOO

```

```

RWAY IS REVEALED!":D(53)=9:P(14)
=0:GOTO35
129 PRINT"BE CAREFUL. YOU MIGHT
HURT YOUR BIG TOE.":GOTO4
130 IFO(N)<150THENPRINT"YOU DO
NOT HAVE IT":GOTO4ELSEIFN<120RR
<131THEN132ELSEIF(14)=150THENP
RINT"THE SPIDER WAS DISTRACTED B
Y ITSSHINE LONG ENOUGH FOR YOU T
O CUTYOURSELF LOOSE AND ESCAPE":
R=132:O(12)=-1:P(3)=P(3)-1:GOTO1
4
131 PRINT"THE SPIDER WAS MOMENTA
RILY DIS=TRACTED; BUT WITHOUT A
KNIFE YOU CANNOT ESCAPE":GOTO178
132 PRINT"I WOULD NOT ADVISE THA
T":GOTO4
133 P(2)=P(2)-1:PRINT"YOUR SCORE
IS"P(2)" MOVES.":GOTO4
134 Q$=INKEY$:IFQ$="^"THENY=Y-30
ELSEIFQ$=CHR$(10)THENY=Y+30ELSEI
FQ$=CHR$(8)THENX=X-24ELSEIFQ$=CH
R$(9)THENX=X+24ELSEIFQ$=CHR$(13)
THENRETURN
135 IFY<10THENY=10ELSEIFY>40THEN
Y=40
136 IFX<24THENX=24ELSEIFX>216THE
NX=216
137 POKEL78,K:LINE(X,Y)-(X+4,Y+4
),PSET,BF:POKEL78,Z:LINE(X,Y)-(X
+4,Y+4),PSET,BF:GOTO134
138 IFX<1THEN180ELSEPOKE491,X:IF
P(5)=1ANDR=60THENPMODE2,1:PCLS0:
PMODE4,1ELSEEXEC32714
139 GOTO16
140 FORX=40TO220STEP180:CIRCLE(X
,80),16,0,.3,0,.5:CIRCLE(X,80),8
,0,.3,0,.5:CIRCLE(X,64),16,0,.3,
0,.5:CIRCLE(X,60),16,0,.3:PAINT(
X,60),0,0:N=X-14:DRAW"BM=N;80E2
R2F2M-8,-16NU4BR32NU4M-8,+16E2R
2F2":NEXT
141 DRAW"C0BM128,90R5E5U32G5L5H5
D32NF5BU4M-48,-24D2M-4,-2U26M+4,
+2ND24M+48,+20BR15M+45,-27D24NM-
45,+32D2M+4,-2U25NM-4,+2U2H2M-54
,-14L2M-57,+20G2D2BR16M+40,+16R8
M+38,-22M-40,-12L2M-44,+16D2H2U2
M+46,-17RM+42,+12D2G2"
142 DRAW"BM0,60M+70,-20BR118BU8M
+68,+6":POKEL78,1:PAINT(2,90),,0
:PAINT(128,30),,0:POKEL78,60:PAI
NT(100,50),,0:PAINT(150,60),,0:P
AINT(130,60),,0:POKEL78,2:PAINT(
40,70),,0:PAINT(220,70),,0:PAINT
(130,55),,0:PAINT(2,90),,0:POKE1
78,64:PAINT(2,2),,0
143 GOTO16
144 PMODE1,1:PCLS6:PMODE4,1:DRAW
"BM0,85C0M+101,-25NU60M+155,+30"
:IFO(12)=R THENCIRCLE(100,70),9,
5,.5
145 GOTO16
146 DRAW"BM0,85C0M+100,-25NU60M+
136,+35"
147 FORX=0TO100STEP8:LINE(X,85-X
/4)-(100,X*.6),PSET:LINE(X+4,85-
(X+4)/4)-(0,85-(X+4)*.6),PSET:NE
XT:FORX=0TO112STEP8:LINE(0,X*.7)
-(X*.85,0),BSET:NEXT:FORX=20TO0S
TEP-5:LINE(100,X+35)-(0,X),PSET:
NEXT:FORX=0TO100STEP12:LINE(X,0)
-(100,35-.35*X),PSET:NEXT

```



```
148 FORX=0TO96STEP8:LINE(100, .6*
X)-(100+X,0),PSET:LINE(X+100,60+
X/4)-(100,60-.6*X),PSET:NEXT:FOR
X=104TO136STEP8:LINE(X+100,60+X/
4)-(X,0),PSET:NEXT:FORX=142TO155
STEP6:LINE(X+100,95)-(X,0),PSET:
NEXT:FORX=160TO205STEP6:LINE(X,0
)-(255,250-X),PSET:NEXT
149 FORX=4TO60STEP6:LINE(X+100,6
0+X/4)-(198+X,0),PSET:LINE(X+160
,74+X/4)-(255,X),PSET:NEXT:FORX=
0TO24STEP6:LINE(X+224,90+X/4)-(2
55,62+X),PSET:NEXT:POKE178,2:PAI
NT(128,90),,0:FORX=0TO6STEP3:POK
E178,X/3:LINE(30+X,20+X)-(60-X,4
0-X),PSET,BF:NEXT
150 GOTO16
151 FORX=0TO255STEP2:DRAW"BM=X;,
25C0U10U"+STR$(RND(15))+"BM=X;,3
5RUI0U"+STR$(RND(15))+"BM=X;,95R
M"+MID$(STR$(RND(3)),2)+",-"+MI
D$(STR$(RND(20)+25),2)+"BM=X;,95
RUI0U"+STR$(RND(10)):NEXT
152 POKE178,1:PAINT(2,50),,0:FOR
X=1TO100:PSET(RND(255),RND(10)
+25):NEXT
153 GOTO16
154 DRAW"BM0,75C0R255U50H20L10G1
0L70M-100,+30L45":CIRCLE(180,65)
,40,0,1,.46,.05:PAINT(2,2),0,0:P
OKE178,1:PAINT(2,70),,0:POKE178,
2:PAINT(2,90),,0
155 FORX=1TO100:CIRCLE(40,20),X,5
:NEXT:FORX=1TO100:PSET(RND(255)
,RND(20)+75):PSET(RND(255),RND(
75),5):NEXT:PAINT(180,65),0,0:PA
INT(2,70),0,0
156 DRAW"BM0,75C0R255":CIRCLE(18
0,65),40,0,1,.46,.05:GOTO16
157 DRAW"BM0,65C0M+73,-10E2R2U2E
2R2U2E2R2U2E44R4F44F2R2D2F2R2D2F
2R2D2M255,75":CIRCLE(128,75),60,
0,2:CIRCLE(128,78),60,0,.2,.5,1
:POKE178,2:PAINT(128,90),,0
158 POKE178,1:PAINT(128,75),,0:P
AINT(2,2),0,0:PAINT(250,2),0,0:P
MODE3,1:FORK=0TO7:D=8*K+4:X=126-
K*8:Y=136+K*8:DRAW"BM=X;,0C6D=D;
BM=Y;,0D=D;":NEXT:DRAW"BM+20,+4U
65BR30D74BM-224,-12U70BR30D60":P
MODE4,1
159 GOTO16
160 PMODE2,1:PCLS0:PMODE4,1:DRAW
"BM128,48C5NU48NE48NR96NF48ND48N
G48NL96NH48":FORK=0TO48STEP8:Y=4
8-K:DRAW"BM128,-K;R=Y;F=Y;G=Y;L=
Y;L=Y;H=Y;E=Y;R=Y;":NEXT
161 GOTO16
162 PMODE2,1:PCLS0:PMODE4,1:GOTO
16
163 DRAW"BM128,70C0M-50,-30M+16,
-6F36U40NM-36,+4M+36,+4G36M+50,-
30NH10M-16,-6H12NM+18,+8M-22,-2N
D10M-22,+2NG12M-20,+8G10M":POKE17
8,1:PAINT(2,2),,0:PMODE4,1
164 GOTO16
165 PRINT037,1;TAB(69)2;TAB(70)"
B";:FORK=18TO66STEP24:CIRCLE(50,
K),10,0:NEXT:PRINT@266,R$(R):
PRINT@298,STRING$(10,32):PRINT@2
98,"";:IFR=31THENPRINT"N"ELSEPRI
NT"E"
166 PRINT@416,"WHICH FLOOR <1;2;
B>";:INPUTA$:IFA$="2"THENPRINT"A
HOST OF GUARDS AWAITS YOU.":GOT
O178ELSEIFA$="1"THENIFR=32THENR=
31:PRINT"UP WE GO..."ELSEPRINT"Y
OU WERE ON THE FIRST FLOOR"
167 IFA$="B"THENIFR=31THENR=32:P
RINT"DOWN WE GO..."ELSEPRINT"YOU
WERE ALREADY IN THE BASEMENT"
168 GOTO4
169 H=0:PMODE2,1:PCLS0:PMODE3,1:
DRAW"BM0,95C6R255BM5,90U50E40R16
5F40D50L245":PAINT(2,2),6,6:X=12
8:Y=86
170 PRINT@266,R$(R))
```

```
171 PRINT@416,"YOU HAVE ENTERED
THE LASER OB= STACLE COURSE. YO
U MUST TRAVERSE THIS DANGEROUS CO
URSE.":FORK=1TO2000:NEXT:PRINT@4
16,"USE THE 4 ARROW KEYS TO GET
THE WHITE CURSOR TO THE TOP OF T
HE SCREEN. PRESS <ENTER>";:INPU
TA$
172 A=RND(165)+45:B=RND(165)+45
173 A$=INKEY$:IFA$=CHR$(8)THENX=
X-2ELSEIFA$=CHR$(9)THENX=X+2ELSE
IFA$=CHR$(10)THENY=Y+2ELSEIFA$=C
HR$(94)THENY=Y-2
174 IFY<4THENPRINT@416,STRING$(6
4,32)"CONGRATULATIONS; YOU MADE
IT":R=22:GOTO14ELSEIFPPOINT(X,Y)
<>5THEN172ELSEPSET(X,Y,8):DRAW"B
M=A;,0C7D=Y;BM=B;,0D=Y;":PSET(X
,Y)
175 DRAW"BM=A;,0C5D=Y;BM=B;,0D=Y
;":IFA=X THEN177
176 GOTO172
177 H=H+1:IFH=1THENPRINT@224,"WO
UNDED":GOTO172ELSEIFH=2THENPRINT
@224,"FATALLY WOUNDED":GOTO178
178 FORK=1TO5:PLAYT$:NEXT
179 PCLS5:POKE491,10:EXEC32714:P
RINT@288,"FOR YOU THIS ADVENTURE
IS OVER.",,,:INPUT"CARE TO PLAY
AGAIN";A$:IFLEFT$(A$,1)="N"THEN
POKE113,0:EXEC40999ELSE2
180 PMODE2,3:PCLS5:PMODE4,1:PRIN
T@256,"THE JEWEL BRINGS ONLY DOO
M TO THOSE WHO POSSESS IT. GEO
RGE IS TRAPPED FOR ETERNITY IN T
HE POOL":FORK=1TO5:PLAYT$:NEXT
181 PRINT"YOU NOW RETURN TO CIVI
LIZATION WITH THE CASKET NOTE P
ROVING YOUR INNOCENCE":FORK=1
TO5:PLAYT$:NEXT
182 PMODE2,3:PCLS5:PMODE4,1:PRIN
T@256,"GEORGE WAS NEVER FOUND. I
T WAS ASSUMED HE ESCAPED TO SOU
TH","AMERICA.":FORK=1TO5:PLAYT$:
NEXT
183 PRINT:PRINT"YOU KNOW GEO
RGE HAS ESCAPED TO A PLACE BEYON
D SPACE AND","TIME. A PLACE KNOW
N AS THE...":FORK=1TO5:PLAYT$:NE
XT
184 PCLS5:POKE491,10:EXEC32714:P
RINT@288,"YOU HAVE WON THE ADVEN
TURE","IN" P(2)"MOVES":PRINT:PRIN
T"CARE TO PLAY AGAIN?"
185 A=RND(235)+10:B=RND(25)+10:F
ORK=5TO0STEP-5:FORK=1TO10:DRAW"B
M=A;, -B;C=K;NU=X;NE=X;NR=X;NF=X;
ND=X;NG=X;NL=X;NH=X;":NEXTX,K
186 A$=INKEY$:IFA$="Y"THEN2ELSEI
FA$="N"THENPOKE113,0:EXEC40999EL
SE185
187 POKE360,PEEK(492):POKE361,PE
EK(493):POKE363,PEEK(494):POKE36
4,PEEK(495):POKE32211,48:RETURN:
'get initial ram hooks
188 FORK=0TO3:Z=84-12*K:FORW=2TO
0STEP-1:ONW+1GOSUB189,192,195:NE
XTW,K:GOTO16
189 IF(D(R+K*C)ANDW(0))<>W(0)THE
NX=223:GOSUB197:X=192-64*K:DRAW"
BD6L=X;D=Z;R=X;U=Z;":K=4
190 RETURN
191 'right walls
192 X=254:GOSUB197:IF(D(R+K*C)AN
DW(1))=W(1)THENDRAW"BD6L32D=Z;R3
2"ELSEDRAW"M-32,+6;D=Z;M+32,+6"
193 RETURN
194 'left walls
195 X=0:GOSUB197:IF(D(R+K*C)ANDW
(2))=W(2)THENDRAW"BD6R32D=Z;L32"
ELSEDRAW"M+32,+6;D=Z;M-32,+6"
196 RETURN
197 DRAW"BM"+STR$(ABS(X-32*K))+",
"+STR$(6*K):RETURN
198 '
199 FORK=0TO24:READR$(K):NEXT:DA
TAA JAIL CELL,A HALL,AN EMPTY RO
```

```
OM, ARSENAL, STORAGE ROOM, INFIRMAR
Y, WARDEN'S OFFICE, SHOP, KITCHEN, I
SOLATION WARD, MORGUE, FIELD, RIVER
BANK, SAUNA ROOM, HOT TUB, A LONEL
Y ROAD
200 DATA CAVE OPENING, TUNNEL, PIT
, CLEARING, A CAVERN, WEB, POOL, ELEV
ATOR, LASER TEST ROOM
201 FORK=0TO140:READD(K),R(K):NE
XT:DATA2,,2,,3,1,3,1,3,1,7,1,3,1
,3,1,7,1,3,1,0,0,2,1,7,1,7,1,1,1
,14,1,3,1,2,1,14,1,3,1,5,1,2,24,
11,1,11,1,1,8,14,1,3,1,5,1,10,1,
1,3,12,1
202 DATA8,23,2,23,3,1,3,1,15,1,1
,4,10,1,3,1,3,1,9,1,6,1,3,1,7,1,1
,7,1,11,1,3,1,7,1,3,1,7,1,1,5,8,1
,2,6,8,2,12,1,2,2,3,1,15,1,1,2,1
0,1,1,10,2,2,3,1,3,1,9,1,2,7,3,1
,11,1,3,1,1,1,1,9
203 DATA6,11,7,11,7,15,7,11,7,11
,7,11,7,11,1,12,7,11,5,11,14,11,
15,11,15,15,15,11,15,11,15,11,15
,11,1,12,15,11,13,11,10,11,11,11
,11,15,11,15,11,15,11,15,11,11,1
,12,15,11,13,11,2,17,3,17,3,17,7
,17,7,17,3,16,3,15,3,15,11,15,13
,13
204 DATA,14,2,17,7,17,15,17,13,1
7,6,17,5,17,6,17,1,20,12,15,6,17
,3,17,11,17,11,17,9,17,10,17,13,
17,12,17,22,12,15,,21,2,17,2,17
,,18,3,17,9,17,10,17,11,17,1,17,
8,19
205 FORK=1TO37:READO$(K),O(K):NE
XT
206 DATANOTE,1,PAPER,7,RIFLE,-1,
AMMO,-1,MASK,-1,BOTTLE,-1,KEY,-1
,MATCHES,24,DRILL,-1,SHOVEL,-1,F
LASHLIGHT,100,BRACELET,111,CONTA
CT,106,KNIFE,112,ROPE,123
207 DATAWINDOW,9,GUNCASE,29,CABI
NET,50,BOX,-1,DOOR,9,PICTURE,52,
SAFE,-1,PANEL,53,CASKET,60,TOOLS
,65,RIVER,78,HOT TUB,110,GAS,-1,
LINK FENCE,140,SIGN,-1,SPIDER,13
1,WEB,131,PIT,134,POOL,119,BLOOD
,135,CUPBOARDS,36,JEWEL,129
208 V$="LOOXAGETAKSTRPUTDROPE
JUMSWICLOGO CLIUNLLOCDIGDRITHRE
ASHOQUISAVLOAHELINVOFFON MOVEATK
ICGIVSCO"
209 N$="NOTEPAPERIFLMMOMASKBOTT
KEY MATCDRILLSHOVFLASBRACCONTKNIF
ROPEWINDGUNCCABIBOX DOORPICS SAFE
PANECASKTOOLRIVETUB GAS FENCSIGN
SPIDWEB PIT POOLBLOOCUPBJEWE"
210 V(1)=8:V(2)=4:V(3)=2:V(4)=1:
S$="ARSSTOINFMORWARSHOKIFIEA LS
AUHOTCLERIVCAVA CWEBPITPOOLELAS
":K=RND(-TIMER):FORK=15TO17:P(K)
=RND(35):NEXT
211 PMODE4,1:COLOR0,5:SCREEN1,1:
T$="T5L804GP8G#P8GP8EP8":FORK=1T
O6:PLAYT$:NEXT:P(10)=RND(9)*24:P
(14)=2:PCLS5:CLS:R=2:A$="W":GOTO
8
```



# The Maze of Moycullen

Thomas Riley

**AS**

Muhra, leader of the Catella Village, you and your entire family have been captured and taken prisoner by the vile Doghedra and his barbaric warriors.

Knowing you as peaceful people who never carry defense weapons, the capture was harmless. But, the intentions were not, for Doghedra has planned your imprisonment to help his village prosper. And, if the scheme wrks, he will reign supreme.

Since the people of your village are extremely agricultural, and Doghedra's are not, his first demand is for your people to grow sufficient grain and produce to feed his pernicious tribe through the winter. Second, the first-born daughters from each family of your village must be brought to his camp to become subservient wives for his warriors. Finally, 20 of your strongest men will have to go to work in Doghedra's mines, and for each man that dies, one will be chosen to replace him. If all demands are met, you and your family will be freed. If not, all of you will encounter a torturous death.

Your family, although distressed, knows since you are a proud representative of your village, that Doghedra's demands will never be met. But, they do not know that you are only one of a few who remembers the long-forgotten custom of these warriors. Fortunately, Doghedra is another who knows that a condemned man is permitted to request a challenge. To honor the custom, your plea is granted.

Of course, the request does not disturb the fearless leader; if you survive the challenge he has in mind you will make him the richest and most powerful leader of the land, for you are to enter the maze of Moycullen to recover the treasure within.



The maze was built generations ago by a powerful race of people to safeguard their riches. But, the entire race was annihilated during war, leaving no one to pass on the secret of the maze.

Doghedra has you just where he wants you. Since hundreds of his strongest warriors were never seen again after entering the maze, he knows whatever happens, satisfaction will be his reward. However, surviving the maze and recovering the treasure is the only way to save your family.

## Loading and Playing Instructions

*The Maze of Moycullen* is a text Adventure that runs on a 16K Color Computer by entering `POKE25,6:NEW` before loading. The command is needed to clear sufficient memory for the program.

Traveling through the maze and performing the necessary task is simplified with the use of one-letter directional commands, as well as verb and noun responses. For example, enter 'N' to GO NORTH, or 'T' for TAKE, then 'A' for APPLE to take the apple when prompted with TAKE WHAT?

A bird's-eye view of each room is graphically displayed in the upper left corner of the screen. In all the rooms, the floor is shown in black, while the walls are depicted in blue and buff. By viewing the graphics display, an available exit can easily be chosen.

All valid verbs are displayed in the upper right section of the screen as are the items in the player's possession. The bottom half of the screen prints the activities of the occupied room, any objects within the room and in the player's possession, and prompts the player for the one-letter commands.

100	.....124	3430	.....195
1250	.....114	4220	.....229
1430	.....9	4250	.....206
2100	.....26	4285	.....143
2170	.....1	5100	.....16
2302	.....187	5225	.....121
2368	.....206	5295	.....119
2460	.....237	50000	.....211
3140	.....120	50940	.....209
3302	.....17	END	.....253

16K users, before loading: POKE25,6:NEW

The listing: MOYCULEN

```

1 CLS:PRINT@108,"THE MAZE":PRINT
@170,"OF MOYCULEN":PRINT@234,"B
Y TOM RILEY"
2 TD$=CHR$(128):FOR Y=331TO338:P
RINT@Y,TD$;:NEXTY:PRINT@340,TD$;
:PRINT@365,TD$;:PRINT@369,TD$;:P
RINT@372,TD$;:PRINT@395,TD$;:PRIN
T@397,TD$;:PRINT@399,TD$;:PRINT@
401,TD$;:PRINT@403,TD$;TD$
3 PRINT@427,TD$;:PRINT@431,TD$;:
PRINT@436,TD$;:FOR Y=459 TO 468:
PRINT@Y,TD$;:NEXTY
4 PLAY"O3L8CDL4.EL8DL4CCDCO2L2AL
4GL2EL4GALL6AGAO3CL2DL8CDL2EL4DL
2CL4EL2O2AO3L4CO2L2GL4O3CL2O2EL4
DL2CO3L4CO2L2GO3L4CO2L2EO3L8CO2B
O3L2CO2L4BO3L2.CL2C"
5 READ D:PRINT@D,"*";:PLAY"L4B":
PRINT@D," ";:IF D=339THENGOTO10E
LSEGOTO5
7 DATA363,364,396,428,429,430,39
8,366,367,368,400,432,433,434,40
2,370,371,339
10 CLS:GOSUB50000:PRINT@7,"N";:P
RINT@97,"W";:PRINT@109,"E";:PRIN
T@231,"S";:PRINT@258,"I SEE";
100 GG=0:BB=0:JJ=0:RR=0:AA=0:OO=
0:MM=0:FF=0:UU=0:NSS="NOTHING SP
ECIAL":IM$="NOT A VALID MOVE":W$
=CHR$(175):P$=CHR$(207):YC$="YOU
CAN'T TAKE THAT":YK$="YOU CAN'T
KILL THAT"
1100 GOSUB50000
1102 GOSUB50000
1104 GOSUB50000
1106 GOSUB50000
1108 GOSUB50000
1110 PRINT@290,NSS:GOSUB60000
1120 IF Z$="N"THENGOTO210ELSEIF
Z$="E"THENGOTO120ELSEPRINT@353
,IM$:PLAY"L4AB":PRINT@352," ":GO
TO1110
1200 GOSUB50000
1202 GOSUB60000
1204 GOSUB50000
1206 GOSUB50000
1210 IF FF=0THENPRINT@290,"FLAMI
NG TORCH"ELSEPRINT@290,NSS
1215 GOSUB60000
1220 IF Z$="N"THENGOTO220ELSEIF
Z$="E"THENGOTO130ELSEIF Z$="W"
THENGOTO110
1230 IF Z$="T"ANDFF=0THENPRINT@3
53,"TAKE":PRINT@385,"what";:GOTO
1250ELSEIF Z$="T"THENPRINT@353,"
THERE IS NOTHING TO TAKE":PLAY"L
4AB":PRINT@353," ":GOTO1210
1240 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO1210
1250 GOSUB60000
1260 IF Z$="F"THENPRINT@385,"FLA
MING TORCH":PRINT@417,"FLAMING T
ORCH TAKEN":FF=1:PLAY"L4CD":GOTO
1270ELSEPRINT@385,YC$:PLAY"L4AB"
1270 PRINT@353," ":PRINT:PRINT:G

```

```

OTO1204
1300 GOSUB50000
1302 GOSUB50000
1304 GOSUB50000
1306 GOSUB50000
1310 PRINT@290,NSS:GOSUB60000
1320 IF Z$="E"THENGOTO140ELSEIF
Z$="W"THENGOTO120ELSEPRINT@353
,IM$:PLAY"L4AB":PRINT@353," ":GO
TO 1310
1400 GOSUB50000
1402 GOSUB50000
1404 GOSUB50000
1406 GOSUB50000
1408 GOSUB50000
1410 IF UU=0THENPRINT@290,"UGLY
CYCLOPS"ELSEPRINT@290,"DEAD CYCL
OPS"
1415 GOSUB60000
1420 IF Z$="N"ANDUU=1THENGOTO240
ELSEIF Z$="N"THENGOTO1450
1425 IF Z$="W"ANDUU=1THENGOTO130
ELSEIF Z$="W"THENGOTO1450
1430 IF Z$="K"ANDUU=0THENGOTO146
ELSEIF Z$="K"THENPRINT@353,"SHE
IS ALREADY DEAD":PLAY"L4AB":PRI
NT@353," ":GOTO1410
1440 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO1410
1450 PRINT@353,"BEFORE YOU CAN E
SCAPE, THE CYCLOPS SEPERATES
YOUR HEAD FROM YOUR NECK. T
OO BAD";:PLAY"L104CO3CO2C":CLS:G
OTO2260
1460 PRINT@353,"KILL":PRINT@385,
"what";:GOSUB60000
1470 IF Z$="U"THENGOTO1480ELSEPR
INT@385,YK$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO1410
1480 PRINT@385,"UGLY CYCLOPS":PR
INT@417,"WITH what";:GOSUB60000
1490 IF Z$="B"ANDBB=1THENGOTO149
5ELSEIF Z$="B"THENPRINT@417,"YOU
DON'T HAVE THAT":PLAY"L4AB":PRI
NT@353," ":PRINT:PRINT:GOTO1410
1492 IF Z$="F"AND FF=1THEN GOTO1
495ELSE IF Z$="F"THEN PRINT@417,
"YOU DON'T HAVE THAT":PLAY"AB":P
RINT@353," ":PRINT:PRINT:GOTO141
0 ELSE PRINT@417,"THAT WON'T WOR
K":PLAY"AB":PRINT@353," ":PRINT:
PRINT:GOTO1410
1495 IF Z$="B"THEN PRINT@422,"BR
OADSWORD":UU=1:PLAY"O3CO1CO2C":P
RINT@353," ":PRINT:PRINT:GOTO141
0 ELSE PRINT@353,"YOU MANAGE TO
START HER CLOAK ON FIRE. A VER
Y ANGRY CYCLOPS PROCEEDS TO EAT
YOU ALIVE":PLAY"L104AO1AO2A":CL
S:GOTO2260
2100 GOSUB50000
2102 GOSUB60000
2104 GOSUB50000
2110 IF RR=0THENPRINT@290,"RATS"
ELSEPRINT@290,"BURNED RATS"
2115 GOSUB60000
2120 IF Z$="N"ANDRR=0THENGOTO215
ELSEIF Z$="N"THENGOTO310ELSEIF
Z$="S"ANDRR=0THENGOTO215ELSEIF
Z$="S"THENGOTO110ELSEIF Z$="E"
THENGOTO220
2140 IF Z$="K"ANDRR=0THENGOTO216
ELSEIF Z$="K"THENPRINT@353,"THE
RE IS NOTHING TO KILL":PLAY"L4AB
":PRINT@353," ":GOTO2110
2145 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO2110
2150 PRINT@353,"BEFORE YOU CAN L
EAVE, THE RATS SURROUND YOU. AS
YOU ARE DEVoured, YOU THI
NK - 'MY POOR FAMILY!";:PLAY"L1
GFL4GFG":CLS:GOTO2260
2160 PRINT@353,"KILL":PRINT@385,
"what";:GOSUB60000
2170 IF Z$="R"THENGOTO2180ELSEPR
INT@385,YK$:PLAY"L4AB":PRINT@352
," ":PRINT:GOTO2110

```

```

2180 PRINT@385,"RATS":PRINT@417,
"WITH what";:GOSUB60000
2190 IFZ$="F"ANDFF=1THENPRINT@42
2,"FLAMING TORCH":RR=1:PLAY"L2O1
CO2C":PRINT@353," ":PRINT:PRINT:
GOTO2110ELSEPRINT@417,"YOU DON'T
HAVE THAT":PLAY"L4ABA":PRINT@35
2," ":PRINT:PRINT:PRINT:GOTO2115
2200 CLS:PRINT:PRINT"THE WALL SW
INGS AWAY AT YOUR TOUCH. AS
YOU STEP INTO THE ROOM, YOUR
FOOT FAILS TO STRIKE ANY FLOOR.
TOO LATE YOU REALIZEIT IS AN OP
EN SHATE!"
2220 FOR X=1TO500:NEXTX
2230 CLS(0):PLAY"L3O5CO4CO3CO2CO
1CL1005AO2"
2240 CLS:PRINT:PRINT"YOUR SHATTE
RED BODY LIES AT THE BOTTOM OF A
THIRTY-FOOT SHAFT. YOUR LAST T
HOUGHT IS OF YOUR FAMILY - WH
O WILL BE EXECUTED ATSDOWN."
2260 PRINT:PRINT:INPUT"DO YOU WI
SH TO TRY AGAIN (Y/N)";:ANS
2280 IF AN$="Y"THENCLS:GOTO10ELS
EEND
2300 GOSUB50000
2302 GOSUB60000
2304 GOSUB50000
2306 GOSUB50000
2310 IF AA=1THENPRINT@290,NSSELS
EPRINT@290,"ANGRY WIZARD"
2315 GOSUB60000
2320 IF Z$="W"THENGOTO220ELSEIF
Z$="N"ANDAA=1THENGOTO330ELSEIF
Z$="N"THENGOTO3355
2330 IF Z$="K"ANDAA=1THENPRINT@3
53,"THERE IS NOTHING TO KILL":PL
AY"L4AB":PRINT@353," ":GOTO2310E
LSEIF Z$="K"THENPRINT@353,"KILL"
:PRINT@385,"what";:GOTO2370
2340 IF Z$="C"ANDMM<1THENPRINT@
353,"YOU HAVE NOTHING TO CAST":P
LAY"L4AB":PRINT@353," ":GOTO2310
ELSEIF Z$="C"THENPRINT@353,"CAST
":PRINT@385,"what";:GOTO2360
2350 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO2310
2360 GOSUB60000
2365 IF Z$="M"THENPRINT@385,"MAG
IC SPELL":MM=-1:PRINT@417,"THE W
IZARD IS IMMobilized.":PLAY"L1CD
E":PRINT@353," ":PRINT:PRINT:PR
INT@241," ":GOTO2306
2368 PRINT@385,"YOU CAN'T CAST T
HAT":PLAY"L4AB":PRINT@353," ":PR
INT:GOTO2310
2370 GOSUB60000
2375 IF Z$="A"THENPRINT@385,"ANG

```



```

RY WIZARD":PRINT@417,"WITH what"
;:GOTO2380ELSEPRINT@385,YK$:PLAY
"L4AB":PRINT@353," ":PRINT:GOTO2
310
2380 GOSUB6000
2385 IF Z$="B"ANDMM=-1THENGOTO23
90ELSEPRINT@353,"BEFORE YOUR WEA
PON FINDS ITS MARK, BOLTS OF E
NERGY FROM THE WIZARD'S FINGERS
TURN YOU AND ALL YOUR POSSESS
IONS INTO ASH.":PLAY"L101B02BL20
1B02BL401B02B":CLS:GOTO2260
2390 PRINT@417,"BROADSWORD":AA=1
:PRINT@439,"WIZARD VANISHES AS
BLADE STRIKES HIM":PLAY"L205EO
3E01E04E02E":PRINT@353," ":PRINT
:PRINT:GOTO2310
2400 GOSUB5000
2402 GOSUB5000
2404 GOSUB5000
2410 PRINT@290,"LIZARD":GOSUB6000
2420 IF Z$="N"THENGOTO3400ELSEIF
Z$="S"THENGOTO1400
2430 IF Z$="K"THENGOTO2450
2440 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO2410
2450 PRINT@353,"KILL":PRINT@385,
"what";:GOSUB6000
2460 IF Z$="L"THENGOTO2470ELSEPR
INT@385,YK$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO2410
2470 PRINT@385,"LIZARD":PRINT@41
7,"WITH what";:GOSUB6000
2480 IF Z$="B"THENPRINT@422,"BRO
ADSWORD":GOTO2490ELSEIF Z$="F"TH
ENPRINT@422,"FLAMING TORCH":GOTO
2490ELSEPRINT@417,"THAT WON'T WO
RK":PLAY"L4AB":PRINT@353," ":PRI
NT:PRINT:GOTO2410
2490 PLAY"L2CL105BBBL402C":CLS:P
RINT:PRINT"THE LIZARD SCREAMS CA
USING A STONE TO FALL FROM TH
E CEILING - YOU ARE CRUSHED BE
NEATH IT.":PLAY"LIABGFE":CLS:GOT
02260
3100 GOSUB5000
3102 GOSUB5000
3104 GOSUB5000
3110 IF BB=0THENPRINT@290,"BROAD
SWORD"ELSEPRINT@290,NS$
3115 GOSUB6000
3120 IF Z$="N"THENGOTO4100ELSEIF
Z$="S"THENGOTO2100
3130 IF Z$="T"ANDBB=0THENGOTO314
0ELSEPRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO3110
3140 PRINT@353,"TAKE":PRINT@385,
"what";:GOSUB6000
3150 IF Z$="B"THENGOTO3160ELSEPR
INT@385,YC$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO3110
3160 PRINT@385,"BROADSWORD":PRIN
T@417,"BROADSWORD TAKEN":BB=1:PL
AY"CCC":PRINT@353," ":PRINT:PRIN
T:GOTO3102
3200 GOSUB5000
3202 GOSUB6000
3204 GOSUB5000
3206 GOSUB5000
3210 IF JJ=0THENPRINT@290,"JEWEL
"ELSEPRINT@290,NS$
3215 GOSUB6000
3220 IF Z$="E"THENGOTO3300ELSEIF
Z$="S"THENGOTO2200
3230 IF Z$="T"ANDJJ=0THENGOTO324
0ELSEPRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO3215
3240 PRINT@353,"TAKE":PRINT@385,
"what";:GOSUB6000
3250 IF Z$="J"THENGOTO3260ELSEPR
INT@385,"YOU CAN'T TAKE THAT":PL
AY"L4AB":PRINT@352," ":PRINT:GOT
03215
3260 PRINT@385,"JEWEL":PRINT@417
,"JEWEL TAKEN":PLAY"CDA":JJ=1:PR
INT@353," ":PRINT:PRINT:GOTO3206

```

```

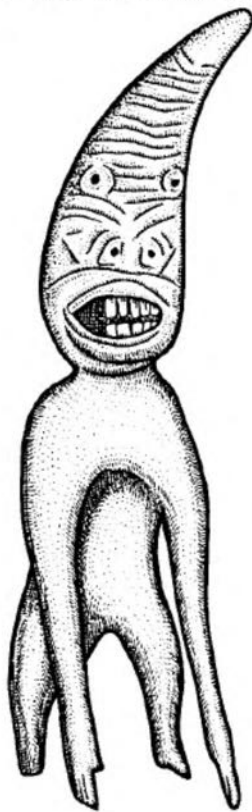
3300 GOSUB5000
3302 GOSUB5000
3304 GOSUB5000
3306 GOSUB5000
3308 GOSUB5000
3310 IF JJ=1ANDAA=0THENPRINT@290
,"ANGRY WIZARD":GOTO3340ELSEPRIN
T@290,NS$
3315 GOSUB6000
3320 IF Z$="N"THENGOTO4300ELSEIF
Z$="S"THENGOTO2300ELSEIF Z$="E"
THENGOTO3400ELSEIF Z$="W"THENGOT
03200
3330 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO3315
3340 GOSUB6000
3350 IF Z$="K"THENGOTO3370ELSEIF
Z$="C"THENGOTO3360
3355 PRINT@353,"THE WIZARD SAYS
'OLAN COLAIN' YOU FIND YOURSELF
THE SIZE OF A FLY AND TRAPPED I
N THE WEB OF A VERY HUNGRY SPIDE
R.":PLAY"L103A04A01A02A":CLS:GOT
02260
3360 IF MM<>1THENPRINT@353,"YOU
HAVE NOTHING TO CAST":PLAY"L4AB"
:PRINT@353," ":GOTO3340ELSEPRINT
@353,"CAST":PRINT@385,"what";:GO
SUB6000
3365 IF Z$="M"THENPRINT@385,"MAG
IC SPELL":MM=-1:PRINT@417,"THE W

```

```

IONS INTO ASH.":PLAY"L101B02B03B
02B":CLS:GOTO2260
3390 PRINT@422,"BROADSWORD":AA=1
:PRINT@439,"WIZARD VANISHES AS
BLADE STRIKES HIM.":PLAY"L205EO
1E04E03E02E":PRINT@353," ":PRINT
:PRINT:GOTO3308
3400 GOSUB5000
3402 GOSUB5000
3404 GOSUB5000
3410 PRINT@290,NS$:GOSUB6000
3420 IF Z$="N"THENGOTO4400ELSEIF
Z$="S"THENGOTO2400ELSEIF Z$="W"
THENGOTO3300
3430 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO3410
4100 GOSUB5000
4102 GOSUB5000
4110 IF OO=0THENPRINT@290,"ORANG
E SPHERE"ELSEPRINT@290,NS$
4115 GOSUB6000
4120 IF Z$="S"THENGOTO3100ELSEIF
Z$="T"ANDOO=0THENGOTO4140
4130 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO4115
4140 PRINT@353,"TAKE":PRINT@385,
"what";:GOSUB6000
4150 IF Z$="O"THENGOTO4160ELSEPR
INT@385,YC$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO4115
4160 PRINT@385,"ORANGE SPHERE":P
RINT@417,"ORANGE SPHERE TAKEN":P
LAY"L2DCD":OO=1:PRINT@353," ":PR
INT:PRINT:GOTO4102
4200 GOSUB5000
4202 GOSUB5000
4204 GOSUB5000
4206 GOSUB5000
4210 PRINT@290,"STAIRS LEADING D
OWN":GOSUB6000
4220 IF Z$="W"THENGOTO4225ELSEPR
INT@353,IM$:PLAY"L4AB":PRINT@353
," ":GOTO4210
4225 GOSUB5000
4230 PRINT@290,"A SMALL OPENING
AND STAIRS":GOSUB6000
4235 IF Z$="E"THENGOTO4200ELSEIF
Z$="W"ANDGG=1THENGOTO4245ELSEIF
Z$="W"THENGOTO4250
4240 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO4230
4245 CLS:PRINT" AS YOU COME THR
OUGH THE OPEN- ING DOGHEDRA'S GU
ARDS SEIZE YOU AND STRIP YOU OF
ALL POSSESSIONS YOU ARE BROUGHT B
EFORE DOGHEDRA.BEHIND HIM THE SU
N IS BISECTED BY THE HORIZON."
4246 PRINT" DOGHEDRA SPEAKS 'YO
U HAVE RETURNED WITH THE TRE
ASURE FROM THE MAZE OF MOYCULLEN
. YOU ARE A GRAND WARRIOR."
4247 PRINT" AS I PROMISED, YOU
AND YOUR FAMILY MAY GO FREE.
FURTHER, NO LONGER WILL WE INV
ADE YOUR LANDS, YOU AND YOUR P
EOPLE MAY LIVE IN PEACE.'"
4248 PLAY"02L4G03CCL8DCL402AGG03
CCDL2.EL4DDDDGAGEL2.GO2L4AL16AB
03C02BL4AGEG03CO2BL2AL4GCEGL8AGL
2EL8.CL16EL4DCCL2C"
4249 FOR Q=1TO2000:NEXTQ:CLS:GOT
02260
4250 CLS:PRINT" AS YOU COME THR
OUGH THE OPEN- ING, DOGHEDRA'S G
UARDS SEIZE YOU AND STRIP YOU OF
ALL POSSESSIONS YOU ARE BROUGHT B
EFORE DOGHEDRA.BEHIND HIM THE SU
N IS BISECTED BY THE HORIZON."
4251 PRINT" DOGHEDRA SPEAKS 'YO
U HAVE RETURNED FROM THE MAZ
E OF MOYCULLEN ALIVE. YOU
ARE A GREATWARRIOR. HOWEVER, YOU
FAILED TO RETURN WITH THE GOLDE
N TREASURE AS YOU WERE CHARGED.
YOU AND YOUR FAMILY WILL BE E
XECUTED."
4252 PRINT" AS A GREAT WARRIOR

```



```

IZARD IS IMMOBILIZED":PLAY"L2CDE
":PRINT@353," ":PRINT:PRINT:PRIN
T@241," ":GOTO3340
3367 PRINT@385,"YOU CAN'T CAST T
HAT":PLAY"L4AB":PRINT@353," ":PR
INT:GOTO3340
3370 PRINT@353,"KILL":PRINT@385,
"what";:GOSUB6000
3375 IF Z$="A"THENGOTO3380ELSEPR
INT@385,YK$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO3340
3380 PRINT@385,"ANGRY WIZARD":PR
INT@417,"WITH what";:GOSUB6000
3385 IF Z$="B"ANDMM=-1THENGOTO33
90ELSEPRINT@353,"BEFORE YOUR WEA
PON FINDS ITS MARK, BOLTS OF E
NERGY FROM THE WIZARD'S FINGERS
TURN YOU AND ALL YOUR POSSESS

```

```

IT IS YOUR HONOR TO DIE FIRST SO
AS NOT TO SEE YOUR LOVED ONES S
OFFER. ";
4253 FOR X=1TO1000:NEXTX
4260 FOR X=1TO25:PLAY"01V20L4BL8
V10BBB":NEXTX
4275 CLS:PRINT:PRINT" YOU ARE S
TRAPPED TO THE SACRIFICIAL
STONE. A LARGE SWORD IS RA
ISED ABOVE YOUR HEAD AND...
"
4280 FOR X=1TO10:PLAY"01V20L4BL8
V10BBB":NEXTX
4285 CLS(0):FOR X=1TO1000:NEXTX
4290 GOTO2260
4300 GOSUB50700
4302 GOSUB50400
4304 GOSUB50300
4306 GOSUB50900
4308 GOSUB50500
4310 IF MM=0THENPRINT@290,"MAGIC
SPELL"ELSEPRINT@290,NS$
4320 GOSUB60000
4330 IF Z$="N"THENGOTO5300ELSEIF
Z$="S"THENGOTO3300
4340 IF Z$="T"ANDMM=0THENGOTO435
0ELSEPRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO4320
4350 PRINT@353,"TAKE":PRINT@385,
"what";:GOSUB60010
4360 IF Z$="M"THENGOTO4370ELSEPR
INT@385,YC$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO4320
4370 PRINT@385,"MAGIC SPELL":PRI
NT@417,"MAGIC SPELL TAKEN":PLAY"
L203CO2C":MM=1:PRINT@353," ":PRI
NT:PRINT:GOTO4300
4400 GOSUB50700
4402 GOSUB50300
4404 GOSUB50900
4410 PRINT@290,"COBWEBS":GOSUB60
000
4420 IF Z$="N"THENGOTO5400ELSEIF
Z$="S"THENGOTO3400
4430 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO4410
4440 GOSUB50300
4442 GOSUB50900
4444 GOSUB50900
4410 PRINT@290,"COBWEBS":GOSUB60
000
4420 IF Z$="N"THENGOTO5400ELSEIF
Z$="S"THENGOTO3400
4430 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO4410
4440 GOSUB50300
4442 GOSUB50900
4444 GOSUB50900
5110 IF GG=0THENPRINT@290,"GOLD
COINS"ELSEPRINT@290,NS$
5120 GOSUB60000
5130 IF Z$="E"THENGOTO5200ELSEIF
Z$="T"ANDGG=0THENGOTO5150
5140 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO5120
5150 PRINT@353,"TAKE":PRINT@385,
"what";:GOSUB60010
5160 IF Z$="G"THENGOTO5170ELSEPR
INT@385,YC$:PLAY"L4AB":PRINT@353
," ":PRINT:GOTO5120
5170 PRINT@385,"GOLD COINS":PRIN
T@417,"GOLD COINS TAKEN":PLAY"L3
CO3L2CO4L1C":PRINT@353," ":PRINT
:PRINT:GG=1:GOTO5100
5200 GOSUB50200
5202 GOSUB50800
5204 GOSUB60300
5206 GOSUB50900
5210 PRINT@290,"IDOL ON SOUTH WA
LL":GOSUB60000
5220 IF Z$="E"THENGOTO5300ELSEIF
Z$="W"THENGOTO5240ELSEIF Z$="P"
THENGOTO5250
5225 IF Z$="T"THENPRINT@353,"IDO
L IS ATTACHED TO THE WALL":PLAY
"L4AB":PRINT@353," ":GOTO5210
5230 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO5210
5240 GOSUB50700
5245 PRINT@353,"THE WALL SWINGS
AWAY AT YOUR TOUCH TO REVEAL
A HIDDEN ROOM":PLAY"L201BEG":PRI
NT@353," ":PRINT:GOTO5100
5250 PRINT@353,"PUT":PRINT@385,"
what";:GOSUB60010
5260 IF Z$="J"ANDJJ=1THENGOTO528
ELSEIF Z$="O"ANDOO=1THENGOTO527

```

```

0ELSEPRINT@385,"YOU CAN'T PUT TH
AT ANYWHERE":PLAY"L4AB":PRINT@35
3," ":PRINT:GOTO5210
5270 PRINT@385,"ORANGE SPHERE":P
RINT@417,"IN what";:GOSUB60010
5275 IF Z$="I"THEN PRINT@420,"ID
OL":PRINT@448,"SHARPENED STONES
FLY FROM IDOL'SMOUTH PIERCING YO
UR HEART";:PLAY"L1004ABAL201BAG"
:CLS:GOTO2260ELSE GOTO5280
5280 PRINT@417,"YOU CAN'T PUT AN
YTHING THERE":PLAY"L4AB":PRINT@3
53," ":PRINT:PRINT:GOTO5210
5285 PRINT@385,"JEWEL":PRINT@417
,"IN what";:GOSUB60010
5290 IF Z$<>"I"THENGOTO5280ELSEPR
INT@420,"IDOL":JJ=-1:GOSUB50600
:PRINT@449,"A HIDDEN ROOM APPEAR
S";:PLAY"L103AO2BO1AO2B":PRINT@3
53," ":PRINT:PRINT:PRINT:GOSUB60
000
5295 IF Z$="S"THENGOTO4200ELSEPR
INT@353,"THE WALL SWINGS BACK QU
ICKLY":PLAY"01C":PRINT@384,"AS I
T SLAMS SHUT THE IDOL BREAKS LOOS
E AND TOPPLES UPON YOU. YOU ARE
PINNED BENEATH IT. MY GUESS IS
YOU WILL STARVE TO DEATH.";
5297 PLAY"04L1AO5L5AO2L1BGEC":CL
S:GOTO2260
5300 GOSUB50100
5302 GOSUB50400
5304 GOSUB50600
5306 GOSUB50900
5308 GOSUB50700
5310 PRINT@290,NS$:GOSUB60000
5320 IF Z$="W"THENGOTO5200ELSEIF
Z$="S"THENGOTO4300
5330 PRINT@353,IM$:PLAY"L4AB":PR
INT@353," ":GOTO5310
5400 GOSUB50100
5402 GOSUB50900
5410 PRINT@290,"FUNNY HOLES IN W
ALLS":PLAY"L4AB"
5420 PRINT@352,"YOUR FOOT STRIKE
S A STONE THAT SINKS SLIGHTLY.
INSTANTLY, POISON-TIPPED AR
ROWS SHOOT FROM THE WALLS. YOU
ARE DEAD.":PLAY"L101AAO2CO1AAO2C
":CLS:GOTO2260
50000 F$=CHR$(128):FOR Y=32 TO 1
92 STEP 32
50010 FOR X=2 TO 12:PRINT@(Y+X),
F$;:NEXTX:NEXTY:RETURN
50100 FOR X=34 TO 44:PRINT@X,W$;
:NEXTX:RETURN
50200 FOR X=194 TO 204:PRINT@X,W
$;:NEXTX:RETURN
50300 FOR X=34 TO 194 STEP 32:PR
INT@X,W$;:NEXTX:RETURN
50400 FOR X=44 TO 204 STEP 32:PR
INT@X,W$;:NEXTX:RETURN
50500 FOR X=35 TO 43:PRINT@X,F$;
:NEXTX:RETURN
50600 FOR X=195 TO 203:PRINT@X,F
$;:NEXTX:RETURN
50700 FOR X=66 TO 162 STEP 32:PR
INT@X,F$;:NEXTX:RETURN
50800 FOR X=76 TO 172 STEP 32:PR
INT@X,F$;:NEXTX:RETURN
50900 IF GG=1 THEN PRINT@145,"GO
LD";
50910 IF BB=1 THEN PRINT@177,"BR
OADSWORD";
50920 IF JJ=1 THEN PRINT@152,"JE
WEL";
50930 IF OO=1 THEN PRINT@273,"OR
ANGE SPHERE";
50940 IF MM=1 THEN PRINT@241,"MA
GIC SPELL";
50950 IF FF=1 THEN PRINT@209,"FL
AMING TORCH";
50960 PRINT@17,"NORTH SOUTH";:P
RINT@49,"EAST WEST";:PRINT@81,
"TAKE PUT";:PRINT@113,"KILL
CAST";:RETURN
60000 PRINT@352,"*";

```

```

60010 Z$=INKEY$
60020 IF Z$="" THEN GOTO 60010
60030 RETURN
60100 FOR X=34 TO 44:PRINT@X,P$;
:NEXTX:RETURN
60200 FOR X=194 TO 204:PRINT@X,P$
;:NEXTX:RETURN
60300 FOR X=34 TO 194 STEP 32:PR
INT@X,P$;:NEXTX:RETURN
60400 FOR X=44 TO 204 STEP 32:PR
INT@X,P$;:NEXTX:RETURN

```



# FANTASY



by *Dereck Powell.*

(Due to the complex nature of the disk files for this program, we have decided not to list this program in the magazine. However the series of programs which make up "Fantasy" can be found on this month's Rainbow on Tape or Disk. G.)

As an avid reader of Australian CoCo and a disk drive owner of some 6 months I thought I ought to repay some of the enjoyment gained by submitting a disk. Recently I saw a programmed 'Apple' Disk by a Mr Donald Brown of USA who gave his permission for the free distribution of the same.

The disk programs I submit is based on Mr Brown's disk and much of the dialogue is similar to his, although the methods of achieving similar results vary somewhat, mainly because of the dissimilarity of computer language.

The programs are of the 'Dungeon and Dragon' theme and permit characters to be created (and destroyed if necessary); character variables are stored in sequential files, and are updated from time to time. The main adventure is not written but I hope that with all the preliminaries completed, others will submit adventures to suit.

"Fantasy" is a role playing game where characters are created and assigned certain attributes. They are given 200 gold pieces which they may use to buy or sell weapons or armour and to purchase spells. Gold may be kept on the person or placed in a bank for safe keeping. Weapons (the law allows one per person) include axe, bow, mace, spear or sword and the player is allocated an expertise rating for each weapon.

Armour consists of 'skin' (no armour), leather, chain or plate (trade in values for weapons and armour). Ability to 'kit' an opponent depends upon the player's expertise with the weapon and his/her agility rating. This is offset by the type of armour - the heavier the armour the greater the resistance to using the weapon to its maximum effect. However the more armour the less chance of the player being hit.

Spells include power, speed, heal and blast.

Cost of weapons and armour (also spells) are effected by the persons 'charisma' value which also effects trade in values. Charisma also affects how 'monsters' will react to the player.

Hardiness affects how many 1 point hits the player can receive before death.

As a player successfully defeats an opponent his expertise rating with the weapon used, is increased. His armour expertise is adjusted to ward off more blows made against him. Each weapon has

its own hit points rating e.g. if the hit rate is 1-5 then the damage done will be RND(5) (if the opponent is hit)

## SYNOPSIS OF PROGRAM

### "Fantasy"

Gets players name and puts it into sequential file PLAYER/DAT  
Checks to see if player has a previous adventure by looking for name in NAMES/DAT file, if not found will place name in that file list.  
If player is a new player will run 'NEWCHAR' program else will run 'MAINHALL' if a previous player.

### "Newchar"

Gets the player's name from 'PLAYER/DAT' and gives the player his or her characteristics and informs the player of these. Will inform the Player of the rules and how variables will alter as experiance is gained. Opens sequential file for the player & runs 'MAINHALL'.

### "Mainhall"

Gets player's name from 'PLAYER/DAT'  
Gets player's variables from player's name file. Permits player to buy spells, to buy or exchange armour or weapons, deposit or withdraw gold from the banker, go on an adventure or quit. If he quits, puts new variables into his name file.

### "Names/Dat"

Contains names of all previous and the current player.

### "Player/Dat"

Contains the name of the current Player.

### "EDIT"

Permits editing or deletion of names from the Names/Dat file. ALWAYS leave one name at least on file.

### "LOOK"

Permits a look into sequential files.

EDIT and LOOK files are not a requirement for the game but are useful from time to time when writing the programs.

Obviously a disk system is necessary run this program. I trust that someone may find a use for it.



# THE MIS-ADVENTURES OF MARTHA



by Andrew White

Are you sick of adventures with lines like "AS YOU RUN THE MONSTER PICKS YOU UP AND USES YOU AS A TOOTHPICK. YOU ARE DEAD."? Are you getting tired of page after page of eye-straining green and black writing? Are you tired of the same old D&D themes?

Well I certainly was, so for a change of pace, I wrote "The Goldsoft Adventure". The scenario is as follows:

You are a Tandy computer representative, travelling down to the CoCoConf on the Gold Coast in your trusty utility truck. On the passenger seat, wrapped in several layers of bubble pack and strapped down with the seatbelt, is the prize for the winner of the games competition. You pull the truck up into the carpark to the West of the office, and as you do so fail to notice that you puncture the fuel line on a protruding rock. Hopping out of the car, you shout and look all around the office, only to note that the staff must already be at the 'Conf, on the other end of the coast. Unfortunately your ute now will not start - you eventually trace down the problem and repair it with masking tape - but there's no fuel left! Panic sets in, without it there is no way you can get to the 'Conf on time. Suddenly, inspiration! You recollect that Graham keeps a can of fuel somewhere deep within the bowels of the office, for use with the backup generator for power supply trouble. So you pull out of the tray of the ute your trusty (if somewhat unsteady) old ladder and lean it against the house ... and so the adventure begins. However, it is a race against time (you have only 100 units) or else a fate worse than death comes your way. Martha flies over to the office to see if you're there. (GULP - maybe a D&D death isn't so bad after all...)

## PLAYING THE GAME

Nearly all commands in the game should take the

form "VERB NOUN" (e.g. CLIMB TREE, DROP WALLET), with exceptions noted below the verb table. The table is one of all the verbs understood by the program, and if at any stage at the program you're stuck in a place or situation, refer back to this table and think HARD!

GET	DROP	SOUTH	WEST	EAST	NORTH	DRINK
CLIMB	EAT	READ	LOOK	EXAMINE	ENTER	HELP
OPEN	INSERT	LIFT	QUIT	BREAK	HOOK	TURN
GO	TAKE	PUT	TIME	INVENTORY		

\* Abbreviated forms of the commands "NORTH", "SOUTH", "EAST", "WEST", and "INVENTORY" exist. They are the following followed by <ENTER>: "N", "S", "E", "W", "I".

\* To exit from an inventory press <CLEAR>

\* To exit from the game type "QUIT" and then respond "Y" to the "Are You Sure?" query. This will warmstart the CoCo, a "N" will return you to the game, any other response will poll the keyboard again.

\* That's enough information or I will ruin the fun of finding out through swearing and patience. Remember to be devious, logical, and ingenious!

## INITIALLY SETTING UP THE GAME (USE THE ONE DISK)

1) Type in or load from CoCoOz on tape or disk, a copy of PROGRAM 1 (In the REM statements in the beginning of the program this should have some

reference to "CHARACTER GENERATOR".) Save a copy to disk and then run the program, it will save a machine code file ("C.DAT") to disk for use by GOLDSOFT.

2) Type in or load from CoCoOz on tape or disk, a copy of PROGRAM 2 (In the REM statements in the beginning of the program this should have some reference to "SYMBOL GENERATOR".) Save a copy to disk and then run the program, it will save three data files ("S.DAT", "S1.DAT", "S2.DAT") for use by GOLDSOFT.

3) Type in or load from CoCoOz on tape a copy of GOLDSOFT (that's the big one folks!). Save a copy to disk under the filename "GOLDSOFT.ADV" or "GOLDSOFT.BAS" (it doesn't matter which.) When you run the program it should start with a blue text screen then change to a PMODE 4 title screen. If it does not, check the program, then depending on the error message you have then check the data in PROGRAM 1 and PROGRAM 2.

4) Unless you are a hacker by heart (like moi) make an effort to buy this month's CoCoOz on Tape or Disk. I should know how easy it is to make a small mistake in the data and then have to debug the whole thing.

#### SUBSEQUENT PLAYING OF THE GAME

Type RUN"GOLDSOFT.ADV" or RUN"GOLDSOFT" followed by <ENTER>, that's all there is to it!

#### WHEN TYPING THE GAME

\* Many of the lines are over the length accepted by normal BASIC entry. To overcome this on long lines if you can type no more hit <ENTER> then use the "EDIT" command on that line to finish the line.

#### EXAMPLE:

```
200 etc.etc.etc AND SO I SAID 'WELCO <ENTER>
OK
EDIT 200
200 etc.etc.etc AND SO I SAID 'WELCO
200 etc.etc.etc AND SO I SAID 'WELCOME. WHAT CAN I
DO YOU FOR ME MATEY?'
OK
```

\* Please don't try to modify the game UNTIL the original game is totally debugged and running bug-free. Otherwise you'll end up with an evil snare of "?OM ERROR"'s.

\* Lastly, some people have wondered aloud to me why I coded the program so tightly. Here it is, the RENUM 0,0,1 and reduction of the number of lines by hand-compressing saved me 1.8K of memory!

Well, if you're still with me after all of the above you deserve this: Good Luck! (And watch out for M-m-m-artha!)

# Computer Hut Software

## MAY MADNESS PACKAGE SPECIALS

64K Disk \$86-10	
War of the Worlds	\$44-95
Wizards Castle	\$29-95
Alcatraz	\$17-95
Atlantis	\$21-95
NORMAL PRICE	\$114-80

16K Tape or Disk \$54-60	
Atlantis	\$21-95
Four Mile Island	\$17-95
Grenada Invasion	\$14-95
Cube Adventure	\$17-95
NORMAL PRICE	\$72-80

32K Disk \$84-60	
Blackbeards Island	\$29-95
To Preserve Quandic	\$39-95
Kingdom of Bashan	\$17-95
Doomsday at 2100	\$24-95
NORMAL PRICE	\$112-80

32K Tape \$82-35	
The Vortex Factor	\$29-95
Shenanigans	\$29-95
Witch's Knight	\$24-95
Stone of Rokan	\$24-95
NORMAL PRICE	\$109-80

### 10% DISCOUNT ON ALL ADVENTURES

We accept :-  
 \* BANKCARD  
 \* MASTERCARD  
 \* VISA CARD

VIATEL NUMBER  
 778622200

Computer Hut Software  
 21, Williams Street.  
 Bowen, Qld. 4805.  
 Phone (077) 86-2220

ALL ORDERS SHIPPED SAME DAY

LISTING 1: CHAR GEN

```

0 *****
1 *
2 * GENERATOR FOR THE MACHINE *
3 * CODE CHARACTER SET USED BY*
4 * "THE GOLDSOFT ADVENTURE." *
5 *
6 *****
7 CLS:PRINT" LOADING MACHINE
LANGUAGE"
8 CLEAR1480,31563
9 FOR I= 31564TO 32383
10 READ Q$:POKEI,VAL("&H"+Q$):NE
XTI
11 LINEINPUT"INSERT DISK, THEN P
RESS <ENTER>":I$
12 VERIFYON:PRINT"AMOUNT OF FREE
SPACE IS";FREE(0):PRINT"GRANULE
S."
13 LINEINPUT"DO YOU STILL WISH T
O GO AHEAD? ";I$
14 IFLEFT$(I$,3)="YES"ORLEFT$(I$
,3)="Y"THEN17
15 IFLEFT$(I$,2)="NO"ORIS="N"THE
N11
16 GOTO13
17 PRINT"SAVING TO DISK":SAVEM"C
.DAT",31564,32383,40999
18 CLS:PRINT"SAVED TO DISK":DIR:
PRINT"FREE SPACE IS";FREE(0);"GR
ANULES.":END
19 DATA BD,B3,ED,ED,8C,1,39,1,C6
,0,A6,84,A7,8C,FA,10,AE,2,EC,8C,
F2,8E,E,80,10,83,0,20,25,9,83,0,
20,30,89,1,0,20,F1,30,8B,8C,25,9
F,25,1,39,A6,A0,81,21,25,8,81,7A
,22,4,80,21,20,2,86,3F,C6,8,3D,1
F,53,33,CB,33,C8,20,C6,80
20 DATA A6,C0,43,A7,85,CB,20,C1,
80,26,F5,1F,10,C1,9F,27,7,30,1,6
A,8C,A8,26,CC,39,20,20,20,20,20,
0,20,0,48,48,48,0,0,0,0,0,50,50,
F8,0,F8,50,50,0,20,78,80,70,8,F0
,40,0,C8,C8,10,20,40,98,98,0,10,
78,80,60,80,78,10,0,20,20
21 DATA 20,0,0,0,0,0,10,20,40,40
,40,20,10,0,40,20,10,10,10,20,40
,0,0,88,50,F8,50,88,0,0,0,20,20,
F8,20,20,0,0,0,0,30,30,10,20,0
,0,0,0,F8,0,0,0,0,0,0,0,0,30,3
0,0,8,8,10,20,40,80,80,0,30,48,4
8,48,48
22 DATA 48,30,0,20,60,20,20,20,2
0,70,0,70,88,8,30,40,80,F8,0,70,
88,8,30,8,88,70,0,10,30,50,90,F8
,10,10,0,F8,80,F0,8,8,88,70,0,70
,80,80,F0,88,88,70,0,F8,8,10,20,
40,80,80,0,70,88,88,70,88,88,70,
0,70,88,88,78,8,8,70,0
23 DATA 0,20,20,0,20,20,0,0,0,30
,30,0,30,10,20,0,8,10,20,40,20,1
0,8,0,0,0,F8,0,F8,0,0,0,80,40,20
,10,20,40,80,0,70,88,8,10,20,0,2
0,0,70,88,8,68,9A,88,70,0,20,50,
88,88,F8,88,88,0,F0,48,48,70,48,
48,F0,0,70,88,80
24 DATA 80,80,88,70,0,F0,48,48,4
8,48,48,F0,0,F8,80,80,F0,80,80,F
8,0,F8,80,80,F0,80,80,0,78,80
,80,98,88,88,78,0,88,88,88,F8,88
,88,88,0,70,20,20,20,20,70,0,
8,8,8,8,8,88,70,0,88,90,A0,C0,A0
,90,88,0,80,80,80,80,80,80

```

```

25 DATA F8,0,88,D8,A8,A8,88,88,8
8,0,88,C8,A8,98,88,88,88,0,F8,88
,88,88,88,88,F8,0,F0,88,88,F0,80
,80,80,0,70,88,88,A8,A8,90,68,0,
F0,88,88,F0,A0,90,88,0,70,88,40,
20,10,88,70,0,F8,20,20,20,20,20,
20,0,88,88,88,88,88,88,70,0,88
26 DATA 88,88,50,50,20,20,0,88,8
8,88,A8,A8,D8,88,0,88,88,50,20,5
0,88,88,0,88,88,50,20,20,20,20,0
,F8,8,10,20,40,80,F8,0,70,40,40,
40,40,40,70,0,80,80,40,20,10,8,8
,0,70,10,10,10,10,10,70,0,20,70,
A8,20,20,20,20,0,0,20,40,F8
27 DATA 40,20,0,0,0,0,0,0,0,0,0,0,
0,0,0,18,8,78,88,78,0,80,80,80,F
0,88,88,B0,0,0,0,70,88,80,88,70,
0,8,8,8,78,88,88,78,0,0,0,70,88,
F8,80,70,0,30,20,20,F8,20,20,60,
0,0,0,78,88,78,8,30,0,80,80,B0,C
8,88,88,88
28 DATA 0,0,20,0,20,20,20,70,0,0
,10,0,10,10,90,60,0,80,80,90,A0,
C0,A0,90,0,40,40,40,40,40,40,60,
0,0,0,88,D8,A8,88,88,0,0,0,B0,C8
,88,88,88,0,0,0,70,88,88,88,70,0
,0,0,F0,88,F0,80,80,0,0,0,70,88,
88,78,8,0,0,0
29 DATA 98,E0,80,80,80,0,0,0,70,
80,70,8,70,0,0,20,F8,20,20,20,30
,0,0,0,88,88,88,98,68,0,0,0,88,8
8,50,50,20,0,0,0,88,88,A8,D8,88,
0,0,0,88,50,20,50,88,0,0,0,88,88
,F8,8,38,0,0,0,F8,10,20,40,F8,0

```

LISTING 2: SYM GEN

```

0 *****
1 *
2 * "SYMBOL FILES" GENERATOR *
3 * GENERATES ASCII FILES FOR *
4 * "THE GOLDSOFT ADVENTURE" *
5 * COPYRIGHT (C) 1985 *
6 * BY ANDREW WHITE *
7 * HUBRIS SOFTWARE *
8 *
9 *****
10 GOTO14
11 VERIFYON:CLS3:PRINT@256,"FREE
SPACE ON THIS DISK IS ";FREE(0)
:PRINT@288,"";:INPUT"GRANULES. G
O AHEAD (YES/NO)";ANS$
12 IFANS$="NO"THENCLS:END:ELSEIF
ANS$="YES"THEN13ELSE11
13 SAVE"SYM GEN":DIR:PRINT"FREE
SPACE =";FREE(0):END
14 VERIFYON:CLS3:PRINT@256,"FREE
SPACE ON THIS DISK IS";FREE(0):
PRINT@288,"";:INPUT"GRANULES. GO
AHEAD (YES/NO)";ANS$
15 IFANS$="YES"THEN16ELSEIFANS$=
"NO"THENCLS:ENDELSE14
16 CLS3:PRINT@232,"HUBRIS SOFTWA
RE":OPEN"O",#1,"S.DAT":FORI=1TO
396:READX$:PRINT#1,X$:NEXTI:CLOS
E#1
17 DATA 7F,FF,F8,FF,E2,3F,CF,87
18 DATA C2,13,C8,D9,DD,DD,DB,9D
19 DATA D7,5D,CE,D9,DD,D3,CB,87
20 DATA E2,3F,F8,FF,FF,FF,FF,FF,
0
21 DATA FF,FF,FF,FF,E0,3,EF,3
22 DATA EF,7,E0,83,E1,C3,E3,E3
23 DATA E1,C3,E0,83,E0,3,E0,3
24 DATA E4,13,FF,FF,FF,FF,FF,FF,

```

```

0
25 DATA FF,FF,C0,1,DF,FD,D0,7D
26 DATA DF,FD,DE,5,DF,FD,D0,5
27 DATA DF,FD,D0,5,DF,FD,DF,85
28 DATA DF,FD,C0,1,FF,FF,FF,FF,0
29 DATA FF,FF,F8,E3,E2,21,CF,B9
30 DATA DF,9B,CF,CB,E1,E3,FC,7F
31 DATA FE,F,F7,EF,E7,E7,EF,EF
32 DATA E2,F,F8,7F,FF,FF,FF,FF,0
33 DATA FF,FF,FB,FF,E1,FF,EC,FF
34 DATA CD,FF,E0,FF,F6,7F,FF,3F
35 DATA FF,9F,FF,8F,FF,27,FF,73
36 DATA FF,E1,FF,ED,FF,FF,FF,FF,
0
37 DATA FF,FF,C0,1,DF,7D,D8,D
38 DATA D3,E5,C7,F1,D7,F5,D3,E5
39 DATA D8,D,DF,7D,C0,1,FD,DF
40 DATA FD,DF,FC,1F,FF,FF,FF,FF,
0
41 DATA FF,FF,FF,FD,FF,FB,FF,F7
42 DATA FF,EF,FF,DF,FE,3F,FE,BF
43 DATA FC,3F,F9,FF,F3,FF,E7,7F
44 DATA E4,7F,F0,FF,FF,FF,FF,FF,
0
45 DATA FF,FF,FF,7F,FE,BF,FD,DF
46 DATA FD,BF,FD,7F,FD,BF,FD,DF
47 DATA FD,BF,FD,7F,FD,7F,FD,7F
48 DATA FD,7F,FD,BF,FF,FF,FF,FF,
0
49 DATA FF,3F,FE,3F,FC,1F,F8,F
50 DATA FB,EF,F8,F,F9,4F,FA,AF
51 DATA F9,4F,FA,AF,F9,4F,F8,F
52 DATA FB,EF,F8,F,FF,FF,FF,FF,0
53 DATA FF,FF,C0,1,DF,FD,CA,A9
54 DATA DF,FD,CA,A9,DF,FD,CA,A9
55 DATA DF,FD,C0,1,FF,FF,FF,FF,0
56 DATA FF,FF,F0,1,E0,1,C0,1
57 DATA DF,F1,D8,31,DB,F1,DB,F1
58 DATA D8,71,DB,F1,DB,F1,DB,F1
59 DATA DF,F3,C0,7,FF,FF,FF,FF,0
60 DATA FF,FF,FF,FF,FF,FF,FF,FF,0
61 DATA FE,FF,FE,3F,F0,7,E3,43
62 DATA CE,39,C0,1,CE,39,E3,43
63 DATA F0,7,FE,3F,FF,FF,FF,FF,0
64 DATA OPEN"O",#1,"S1.DAT":FORI=0TO3
90:READY$:PRINT#1,Y$:NEXTI:CLOSE
#1
66 DATA FF,FF,7F,FF,FF,FF,FF,FF,FF
67 DATA FF,FF,BF,FF,FF,FF,FF,FF,FF
68 DATA FF,FF,DF,FF,FF,FF,FF,FF,FF
69 DATA FF,F0,0,1F,FF,FF,FF,1
70 DATA FF,FB,FF,EF,FF,FF,FF,30
71 DATA FF,FD,0,17,FF,FF,FE,67
72 DATA 7F,FE,BF,EB,FF,FF,FC,CF
73 DATA BF,FF,FF,FF,FF,FF,F9,9F
74 DATA DF,FF,AF,FA,FF,FF,FC,1F
75 DATA EF,FF,D0,1,7F,FF,FE,EF
76 DATA F7,FF,EB,FE,BF,FF,FF,57
77 DATA FB,FF,F4,0,5F,FF,FF,AB
78 DATA FD,FF,FA,FF,AF,F0,0,15
79 DATA FE,FF,FD,0,10,7,FF,CA
80 DATA FF,3,FE,BF,EB,FE,55,55
81 DATA 7F,8D,FF,40,5,F5,55,5A
82 DATA BF,9E,FF,BF,FE,F5,55,59
83 DATA 5F,3F,7F,0,0,77,FF,DA
84 DATA AE,7F,BC,7F,FF,FD,0,DB
85 DATA 74,FF,D9,FF,FF,FD,FF,F3
86 DATA 81,FF,E0,0,0,0,0,0
87 DATA 3,FF,F3,FF,FF,F7,FF,FF
88 DATA FD,FF,F9,FF,FF,FF,FF,FF
89 DATA FE,FF,FC,FF,FF,FF,FF,FF
90 DATA FF,7F,FE,7F,FF,FF,FF,FF
91 DATA FF,BF,FF,0,3,0,0,0,0

```

```

92 DATA 7,DF,FF,9E,FF,FF,DF,FF
93 DATA FB,EF,FF,CF,7F,FF,EF,FF
94 DATA FD,F7,FF,E7,BF,CF,F7,FC
95 DATA FE,FB,FF,F3,DF,E7,FB,FC
96 DATA FF,7D,FF,F9,EF,FF,FD,FF
97 DATA FF,BE,FF,FC,F7,FF,FE,FF
98 DATA FF,DF,7F,FE,78,0,0,0
99 DATA 0,F,BF,FF,3D,FF,FB,FF
100 DATA FF,F7,DF,FF,9E,FF,FD,FF
101 DATA FF,FB,EF,FF,CF,7F,FE,FF
102 DATA FF,FD,F7,FF,E7,FB,FF,FF
103 DATA FF,FE,FB,FF,F3,DF,FF,BF
104 DATA FF,FF,7D,FF,F9,EF,FF,DF
105 DATA FF,FF,BE,FF,FC,F7,FF,E0
106 DATA 0,0,1F,7F,FC,7B,FF,CF
107 DATA FF,FF,EF,BF,F9,BD,FF,9F
108 DATA FF,FF,F7,DF,F3,DE,FF,3F
109 DATA FF,FF,FB,EF,E7,EF,7E,7F
110 DATA FF,FF,FD,F7,EF,F7,BC,FF
111 DATA FF,FF,FE,FB,CF,FB,D9,FF
112 DATA FF,FF,FF,7D,9F,FD,E3,FF
113 DATA FF,FF,FF,BE,3F,FE,7,FF
114 DATA FF,FF,FF,C0,7F,FF,0
115 OPEN"O",#1,"S2.DAT":FORI=0TO
191:READZ$:PRINT#1,Z$:NEXTI:CLOS
E#1:CLS:DIR:PRINT"FREE SPACE =";
FREE(0):EXEC41393:END
116 DATA FF,83,FF,FF,FF,73,FF,FF
117 DATA FE,EB,FF,FF,FD,DE,FF,FF
118 DATA FB,BB,FF,FF,F7,7B,FF,FF
119 DATA EE,FB,FF,FF,DD,FB,FF,FF
120 DATA BB,FB,FF,FF,7,FB,FF,FF
121 DATA 77,FB,FF,FF,77,FB,FF,FF
122 DATA 77,FB,FF,FF,77,FB,FF,FF
123 DATA 77,FB,FF,FF,77,FB,FF,FF
124 DATA 77,FB,FF,FF,77,FB,FF,FF
125 DATA 77,FB,FF,FF,77,FB,FF,FF
126 DATA 77,F8,0,3F,77,F7,FF,3F
127 DATA 77,EF,96,BF,77,DB,6D,BF
128 DATA 77,B6,DB,BF,77,6D,F7,3F
129 DATA 76,D3,EE,BF,75,BB,DD,BF
130 DATA 73,FF,BB,BF,70,0,73,BF
131 DATA 77,F7,6B,BF,77,F7,5B,BF
132 DATA 77,F7,3B,BF,70,7,7B,BF
133 DATA 77,D7,7B,BF,77,D7,7B,BF
134 DATA 77,D7,7B,BF,77,D7,7B,BF
135 DATA 77,D7,7B,BF,70,7,0,0
136 DATA 77,F7,7F,FF,77,F7,7F,FF
137 DATA 77,F7,7F,FF,77,F7,7F,FF
138 DATA 77,F7,7F,FF,77,F7,7F,FF
139 DATA 77,F7,7F,FF,7,F0,7F,FF,
0

```

LISTING 3: GOLDSOFT/ADV

```

0 CLS3:PRINT@232,"HUBRIS SOFTWARE";
LOADM"C.DAT":DEFUSR0=31564:DEFUSR1=31574:
CLEAR1480,31563:PMODE4,1:PCLS1:COLOR0,5:GOTO3
1 FORDLAY=1TO2300:NEXTDLAY:RETURN
2 IPPEEK(339)<>191THEN2ELSERBTUR
N
3 CLEAR405:GOSUB158:POKE65495,0:
DIMOB$(30,5),IN$(5),NN$(30),DE$(28):
POKE280,PEEK(275):HM=0:SM=0:MH=0:
FC=0:CN=0:LF=0:SS=0:LC=1:TM=1004:
OB$(3,1)="PRINTER(Dot matrix)":OB$(4,1)=
"PHOTOCOPIER":OB$(5,1)="DISK(With a red label)":
OB$(6,1)="SIGN":OB$(7,1)="PIECE OF PAPER":
OB$(8,1)="COCO(Unconnected)":OB$(9,1)=
"MEMO(From Graham)"

```

```

5 OB$(10,1)="COCO WITH MESSAGE "+CHR$(34)+
"Insert Disk"+CHR$(34):OB$(11,1)="STICKERS
SET":OB$(12,1)="KEY":OB$(14,1)="FLASHBULB":
OB$(15,1)="ROPE(Knotted at intervals)":OB$(16,1)=
"CHAIR(Made of wood)"
6 OB$(19,1)="AIR FRESHENER":OB$(20,1)=
"MUG(Full of coffee)":OB$(21,1)="POSTER(For Aust.
Coco)":OB$(22,1)="LINE(Made of nylon)":OB$(28,1)=
"FUEL CAN":OB$(24,1)="NOTICE(To staff)":OB$(13,1)=
"SANDWICHES(A plate full!)"
7 GOSUB109
8 PCLS1:GOSUB163:LINE(0,0)-(256,94),PSET,B:
GOSUB145:P=384:S$="You see":GOSUB134:IFLEN(DE$(LC))>
32THEN9ELSE10
9 DL$="":DL$=LEFT$(DE$(LC),32):P=416:S$=DL$:
GOSUB134:DR$="":FORL=33TOLEN(DE$(LC)):DR$=DR$+MID$(DE$(LC),L,1):
NEXTL:P=448:S$=DR$:GOSUB134:GOTO101:GOTO11
10 P=416:S$=DE$(LC):GOSUB134:GOSUB101
11 IFLC=1THENFORI=1TO5:IFLEFT$(OB$(1,1),3)<>"FUE"
THENNEXTELSE122
12 CM$="":NN$="":P=704:S$="Input your command":
GOSUB134:P=736:S$=STRING$(32,32):GOSUB134:P=736:S$=""
13 IK$=INKEY$:IFIK$=""THEN13ELSEBPOKE282,255
14 IF(ASC(IK$)<65ANDASC(IK$)>32)OR(ASC(IK$)<32ANDASC(IK$)>13)OR(ASC(IK$)<13ANDASC(IK$)>8)ORASC(IK$)<8ORASC(IK$)>90THEN13:ELSEIFIK$=CHR$(8)THENCM$="":GOTO12
15 P=P+1:S$=IK$:GOSUB134:IFIK$=CHR$(13)THENSOUND100,1:GOTO16:ELSEIFLEN(CM$)>25THENCM$="":GOTO12:ELSECM$=CM$+IK$:GOTO13
16 IFLEN(CM$)=1ANDCM$="N"THENCM$="NOR"ELSEIFLEN(CM$)=1ANDCM$="S"THENCM$="SOU"ELSEIFLEN(CM$)=1ANDCM$="E"THENCM$="EAS"ELSEIFLEN(CM$)=1ANDCM$="W"THENCM$="WES"ELSEIFLEN(CM$)=1ANDCM$="I"THENCM$="IN V"
17 IFLC=12THENFORE=1TO5:IFLEFT$(IN$(E),3)="AIR"THEN18ELSENEXT:GOTO135
18 IFSM=0ANDLC=12THENP=704:S$="The smell in here is bad but you":GOSUB134:P=736:S$="realise and spray the freshener":GOSUB134:GOSUB1:GOSUB1:SM=1:P=704:S$=STRING$(32,32):GOSUB134:P=736:GOSUB134
19 RESTORE:CM$=LEFT$(CM$,3):FORL=1TOLEN(CM$):IFMID$(CM$,L,1)<>" "THENNEXTELSENN$=MID$(CM$,L+1,3)
20 TM=TM-1:IFTM<=0THENGOTO128:ELSEIFTM<30THENP=0:S$="Time is running out - so hurry!":GOSUB134
21 FORI=1TO27:READDAT$:IFDAT$=CM$ THEN ON I GOTO 22,27,29,29,29,29,35,37,39,44,46,10,51,55,68,74,82,85,87,89,91,93,97,55,22,27,100:ELSENEXTI:P=736:S$="Eh? I don't quite understand you":GOSUB134:GOSUB1:GOTO12

```

```

22 FORD=1TO5:IFIN$(D)=""THEN23EL
SENEXTD:P=736:S$="You can't carry anything more.":GOSUB134:GOSUB1:GOTO12
23 FORE=1TO5:IFLEFT$(OB$(LC,E),3)=NN$THEN24ELSENEXTE:IFNN$="LAD"THEN24ELSEIFNN$="TRU"THEN24:ELSEP=736:S$="I don't see that here, do you?":GOSUB134:GOSUB1:GOTO12
24 IFNN$="ACT"ORNN$="PRI"ORNN$="BRO"ORNN$="PHO"ORNN$="COP"ORNN$="SIG"ORNN$="UNC"ORNN$="COC"ORNN$="WOO"ORNN$="CHA"ORNN$="PAL"ORNN$="POS"ORNN$="TRU"ORNN$="LAD"THEN25ELSE26
25 P=736:S$="That's too heavy for you to lift":GOSUB134:GOSUB1:GOTO12
26 IFNN$="KEY"ANDLC=12THENP=736:S$="I can't reach the key.":GOSUB134:GOSUB1:GOTO12:ELSEIFNN$="T"THENP=736:S$="GET what?":GOSUB134:GOSUB1:GOTO12:ELSEIN$(D)=OB$(LC,E):OB$(LC,E)=""GOTO8
27 FORD=1TO5:IFOB$(LC,D)=""THEN28ELSENEXTD:P=736:S$="This area full of junk already.":GOSUB134:GOSUB1:GOTO12
28 FORE=1TO5:IFNN$=""THENP=736:S$="PUT what?":GOSUB134:GOSUB1:GOTO12:ELSEIFLEFT$(IN$(E),3)=NN$THENOBS$(LC,D)=IN$(E):IN$(E)=""GOTO8:ELSENEXTI:P=736:S$="You aren't carrying it":GOSUB134:GOSUB1:GOTO12
29 RESTORE:FORI=1TO27:READDT$:NEXTI:IFLC=17ANDDAT$="WES"ANDMH<>1THENP=736:S$="The door is shut and locked.":GOSUB134:GOSUB1:GOTO12
30 IFLC=24ANDDAT$="SOU"ANDHM<>1THENP=704:S$="The opening is covered by boards":GOSUB134:P=736:S$="padlocked across it.":GOSUB134:GOSUB1:GOSUB1:GOTO12:ELSEIFLC=10AND(DAT$="NOR"ORDAT$="SOU")THEN31ELSE32
31 IFN1<>1THENPCLS1:GOTO139
32 IFLC=4ANDCN=0ANDDAT$="NOR"THENP=736:S$="You cannot get past the copier.":GOSUB134:GOSUB1:GOTO12
33 FORI=1TOLC:READKL:READN:READS:READE:READW:NEXTI:IFDAT$="NOR"ANDN>0THENLC=N:GOTO8ELSEIFDAT$="SOU"ANDS>0THENLC=S:GOTO8ELSEIFDAT$="EAS"ANDE>0THENLC=E:GOTO8ELSEIFDAT$="WES"ANDW>0THENLC=W:GOTO8

```





34 P=736:S\$="You can't go in that direction.":GOSUB134:GOSUB1:GOTO12  
35 FORE=1TO5:IFLEFT\$(IN\$(E),3)="MUG" THEN36ELSENEXT E:P=736:S\$="You don't have anything to drink":GOSUB134:GOSUB1:GOTO12  
36 P=704:S\$="Urgh! This is Graham's coffee, &":GOSUB134:P=736:S\$="it's 3 weeks old! You lose 5 min":TM=TM-5:GOSUB134:IN\$(E)="" :GOSUB1:GOTO12  
37 IFLC=27ORLC=25ORLC=26ORLC=28THENP=704:S\$="The dirt is soft and digs easily":GOSUB134:P=736:S\$="but you find nothing.":GOSUB134:GOSUB1:GOTO12  
38 IFLC=1THENP=736:S\$="You can't dig on a metal floor!":GOSUB134:GOSUB1:GOTO12:ELSEP=736:S\$="You can't dig on a wooden floor!":GOSUB134:GOSUB1:GOTO12  
39 IFLC<>25ANDLC<>27ANDLC<>1THENP=736:S\$="You can't climb that.":GOSUB134:GOSUB1:GOTO12ELSEIFLC=1ANDNN\$="LAD" THENP=736:S\$=" (HINT: Try "+CHR\$(34)+"GO LADDER"+CHR\$(34)+"":GOSUB134:GOSUB1:GOTO12  
40 IFLC=1ANDNN\$="TRU" THENP=736:S\$="You climb & slip, losing 5 min!":TM=TM-5:GOSUB134:GOSUB1:GOTO12  
41 IFLC=25AND(NN\$="BOX" ORNN\$="BAC" ORNN\$="STE" ORNN\$="STA") THENP=736:S\$="You climb & fall, losing 5 min!":GOSUB134:TM=TM-5:GOSUB1:LC=27:GOTO8  
42 IFLC=27AND(NN\$="BOX" ORNN\$="BAC" ORNN\$="STE" ORNN\$="STA") THENP=704:S\$="Argh! You are knocked out by a":GOSUB134:P=736:S\$="falling box! You lose 30 minutes":GOSUB134:TM=TM-30:GOSUB1:LC=25:GOTO8  
43 P=736:S\$="CLIMB What?":GOSUB134:GOSUB1:GOTO12  
44 IFNN\$<>"PLA" ANDNN\$<>"SAN" THENP=736:S\$="Nong! You can't eat that!":GOSUB134:GOSUB1:GOTO12:ELSEFORI=1TO5:IFLEFT\$(IN\$(I),3)="SAN" THEN45ELSENEXT I:P=736:S\$="You don't have them to eat.":GOSUB134:GOSUB1:GOTO12  
45 P=736:S\$="Annette's sandwiches! Mmmm-mmm.":GOSUB134:IN\$(I)="" :GOSUB1:GOTO12  
46 IFLC<>5ANDLC<>6ANDLC<>7ANDLC<>9ANDLC<>24THENP=736:S\$="You can't read that, I'm afraid.":GOSUB134:GOSUB1:GOTO12:ELSEIFLC=5AND(NN\$="RED" ORNN\$="LAB") THENP=736:S\$=CHR\$(34)+"Insert into activated Coco"+CHR\$(34):GOSUB134:GOSUB1:GOTO12

47 IFLC=6ANDNN\$="SIG" THENP=736:S\$=CHR\$(34)+"Goldsoft 33 North St Southport"+CHR\$(34):GOSUB134:GOSUB1:GOTO12  
48 IFLC=7AND(NN\$="PIE" ORNN\$="PAP") THENP=736:S\$=CHR\$(34)+"3 Pizzas to go, Rugatinis.":CHR\$(34):GOSUB134:GOSUB1:GOTO12  
49 IFLC=9ANDNN\$="MEM" THENP=736:S\$=CHR\$(34)+"HINT: Try to DIG up clues"+CHR\$(34):GOSUB134:GOSUB1:GOTO12  
50 IFLC=24ANDNN\$="NOT" THENP=736:S\$=CHR\$(34)+"If key does not fit, use a ..."+CHR\$(34):GOSUB134:GOSUB1:GOTO12:ELSEP=736:S\$="Eh? I can't read that!":GOSUB134:GOSUB1:GOTO12  
51 IFNN\$="CHA" ORNN\$="COP" ORNN\$="PHO" ORNN\$="LIN" ORNN\$="PRI" THEN52 ELSEP=736:S\$="I see nothing special at all.":GOSUB134:GOSUB1:GOTO12  
52 IFLEFT\$(OB\$(16,1),3)="CHA" ANDNN\$="CHA" ANDLC=16THENP=736:S\$="Hey! There's a crack in the lug!":GOSUB134:GOSUB1:GOTO12ELSEIFNN\$="PHO" ORNN\$="COP" ANDLC=4THENP=736:S\$="You notice that it's on a swivel":GOSUB134:GOSUB1:GOTO12  
53 IFNN\$="LIN" THENFORI=1TO5:IFLEFT\$(IN\$(I),3)<>"LIN" THENNEXT ELSEP=704:S\$="There's a dirty great big hook":GOSUB134:P=736:S\$="on the end of it ... Hmmm ...":GOSUB134:GOSUB1:GOTO12  
54 IFNN\$="PRI" THENP=704:S\$="There's a printout of something":GOSUB134:P=736:S\$="called "+CHR\$(34)+"Forest"+CHR\$(34):GOSUB134:GOSUB1:GOTO12:ELSEP=736:S\$="I don't see that here - do you?":GOSUB134:GOSUB1:GOTO12  
55 IFNN\$="KNO" ORNN\$="ROP" ORNN\$="LAD" THEN56ELSEIFNN\$="EAS" ORNN\$="WES" ORNN\$="NOR" ORNN\$="SOU" THENDATS=NN\$:GOTO29ELSEP=736:S\$="How can I GO to such a thing?":GOSUB134:GOSUB1:GOTO12  
56 IFNN\$="KNO" ORNN\$="ROP" THEN57 ELSE58  
57 FORI=1TO5:IFLEFT\$(OB\$(LC,1),3)="ROP" THEN61ELSENEXT I:P=736:S\$="I can't see a knotted rope here.":GOSUB134:GOSUB1:GOTO12  
58 IFLC=1ORLC=4ANDNN\$="LAD" ANDLF=0THEN59ELSE60  
59 IFLC=1THENLC=4:GOTO8:ELSEIFLC=4THENLC=1:GOTO8  
60 IFLC=1ORLC=4THENP=736:S\$="You can't the ladder has fallen!":GOSUB134:GOSUB1:GOTO12:ELSEP=736:S\$="The ladder isn't here.":GOSUB134:GOSUB1:GOTO12  
61 IFLC=1THENLC=4:GOTO62:ELSEIFLC=4THENLC=1  
62 IFLC=1THENOB\$(4,I)="" :GOTO63 ELSEGOTO65  
63 FORI=1TO5:IFOB\$(1,I)="" THEN64 ELSENEXT I:GOTO65  
64 OB\$(1,I)="ROPE (Knotted at intervals)":GOTO8  
65 IFLC=4THENOB\$(1,I)="" :FORI=1TO5:IFOB\$(4,I)<>" THENNEXT ELSEOB\$(

(4,I)="ROPE (Knotted at intervals)"  
66 IFLC<>4ANDLC<>1THENP=736:S\$="Nothing happens.":GOSUB134:GOSUB1:GOTO12  
67 GOTO8  
68 ON RND(5)GOTO69,70,71,72,73  
69 P=736:S\$="Open your eyes!":GOSUB134:GOSUB1:GOTO12  
70 P=736:S\$="I'm sorry, but I cannot help you":GOSUB134:GOSUB1:GOTO12  
71 P=736:S\$="I'm not in a helpful mood!":GOSUB134:GOSUB1:GOTO12  
72 P=736:S\$="You could examine the listing!":GOSUB134:GOSUB1:GOTO12  
73 P=736:S\$="I know nothink, Her Commandant!":GOSUB134:GOSUB1:GOTO12  
74 IFNN\$="DOO" ANDMH=0ANDLC=17THENFORI=1TO5:IFIN\$(I)<>"KEY" THENNEXT ELSEP=736:S\$="You use the key & the door opens":GOSUB134:GOSUB1:GOTO80  
75 IFNN\$="DOO" ANDLC=17THENP=736:S\$="Sorry, it's locked.":GOSUB134:GOSUB1:GOTO12  
76 IFNN\$="OPE" ANDLC=24ANDMH<>1THENFORI=1TO5:IFLEFT\$(IN\$(1),3)<>"HAI" THENNEXT ELSE81  
77 IFNN\$="DOO" ANDLC=17ANDMH=1THENP=736:S\$="It's already opened":GOSUB134:GOSUB1:GOTO12  
78 IFNN\$="OPE" ANDLC=24ANDMH=1THENP=736:S\$="It's already opened":GOSUB134:GOSUB1:GOTO12  
79 P=736:S\$="I can't open that!":GOSUB134:GOSUB1:GOTO12  
80 MH=1:GOTO8  
81 P=704:S\$="Using the hair pin you pick the":GOSUB134:P=736:S\$="lock and the opening opens.":GOSUB134:GOSUB1:GOSUB1:HM=1:GOTO8  
82 IFNN\$="DIS" THENFORI=1TO5:IFLEFT\$(IN\$(E),3)="DIS" THEN83ELSENEXT I:P=736:S\$="You don't have it on you.":GOSUB134:GOSUB1:GOTO12ELSEIFNN\$<>"DIS" THENP=736:S\$="You can't insert that!":GOSUB134:GOSUB1:GOTO12  
83 IFLC=10THEN84ELSEP=736:S\$="There is nothing to insert it in":GOSUB134:GOSUB1:GOTO12  
84 P=704:S\$="You insert the disk, the Coco":GOSUB134:P=736:S\$="clunks and you then remove it.":GOSUB134:NI=1:GOSUB1:GOSUB1:GOTO12  
85 FORI=1TO5:IFNN\$=LEFT\$(OB\$(LC,1),3) THEN86ELSENEXT I:P=736:S\$="That's weigh (heh-heh) too heavy":GOSUB134:GOSUB1:GOTO12





```
86 IFNN$="PHO"ORNN$="COP"ORNN$="
COC"ORNN$="POS" THENP=736: S$="Tha
t's weigh (heh-heh) too heavy":G
OSUB134:GOSUB1:GOTO12:ELSEP=736:
S$="There's nothing underneath."
:GOSUB134:GOSUB1:GOTO12
87 PCLS0:P=361:S$=" Are you sure
?":GOSUB134
88 IK$=INKEY$: IFIK$="" THEN88ELSE
IFIK$="Y" THENPOKE113,0: EXEC40999
ELSEIFIK$="N" THEN8ELSE88
89 IFNN$<>"CHA" THENP=736: S$="You
can't break that!":GOSUB134:GOS
UB1:GOTO12ELSEIFLEFT$(OB$(LC,1),
3)="CHA" THEN90ELSEP=736: S$="I ca
n't see a chair.":GOSUB134:GOSUB
1:GOTO12
90 P=704:S$="CRACK! It splits, a
nd something":GOSUB134:P=736:S$=
"falls out. ":GOSUB134:GOSUB1:G
OSUB1:OB$(16,1)="HAIR PIN":OB$(1
6,2)="BROKEN CHAIR":GOTO8
91 PCLS:P=384:S$="You are carryi
ng the following":GOSUB134:P=44
8:FORD=1TO5: IFIN$(D)<>" THENSS$="
* "+IN$(D):GOSUB134
92 P=P+64:NEXTD:GOSUB2:GOTO8
93 FORI=1TO5: IFLEFT$(OB$(LC,I),3
)=NN$ THEN94ELSENEXT: P=736: S$="I
can't see that here.":GOSUB134:G
OSUB1:GOTO12
94 IFNN$<>"KEY" THENP=736: S$="Sor
ry, I can't hook that!":GOSUB134
:GOSUB1:GOTO12
95 FORD=1TO5: IFLEFT$(IN$(D),3)="
LIN" THEN96ELSENEXT: P=736: S$="You
've got zilch to hook it with":G
OSUB134:GOSUB1:GOTO12
96 OB$(LC,I)="" :FORI=1TO6: IFIN$(
I)<>" THENNEXTELSEIN$(I)="KEY": P
=736: S$="You got the key! Well d
one!":GOSUB134:GOSUB1:GOTO8ELSEP
=736: S$="You are carrying too mu
ch.":GOSUB134:GOSUB1:GOTO12
97 IFNN$="PHO"ORNN$="COP" THEN98E
LSEIFNN$="" THENP=736: S$="TURN wh
at?":GOSUB134:GOSUB1:GOTO12ELSEP
=736: S$="I can't turn that I'm a
fraid.":GOSUB134:GOSUB1:GOTO12
98 IFLC<>4 THENP=736: S$="I don't
see it here.":GOSUB134:GOSUB1:GO
TO12ELSEIFCN=1 THENP=736: S$="It's
locked in the turn position":GO
SUB134:GOSUB1:GOTO12
99 CN=1:P=704:S$="CRIPES! When y
ou turned the cop":GOSUB134:P=7
36: S$="ier it pushed the ladder
```

```
down!":GOSUB134:GOSUB1:GOSUB1:LF
=1:GOTO8
100 P=736:S$="You have"+STR$(TM)
+" turns left.":GOSUB134:GOSUB1:
GOTO12
101 IFLF=0AND(LC=1ORLC=4) THENP=4
80: S$="There is a Ladder here.":
GOSUB134:ELSEP=480: S$="Visible i
tems":GOSUB134
102 FORD=1TO5: IFOB$(LC,D)="" THEN
NEXTELSEP=P+32: S$=OB$(LC,D):GOSU
B134:NEXT
103 P=672: S$="Possible exits":G
OSUB134:RESTORE:FORI=1TO27: READD
T$:NEXTI:FORI=1TOLC: READKL: READN
:READS: READE: READW: NEXTI: IFMH=0A
NDLC=24 THENS=0
104 IFW>0 THENP=688: S$="N":GOSUB1
34
105 IFS>0 THENP=692: S$="S":GOSUB1
34
106 IFE>0 THENP=696: S$="E":GOSUB1
34
107 IFW>0 THENP=700: S$="W":GOSUB1
34
108 RETURN
109 SCREEN1,1:P=0: S$="Welcome to
...":GOSUB134:P=70: S$="THE GOLDS
OFT ADVENTURE":GOSUB134:P=714: S$
="Press Enter":GOSUB134
110 P=647: S$="Copyright (C) 1985
":GOSUB134:P=137: S$="by Andrew W
hite":GOSUB134:P=192: S$=STRING$(
32,"*"):GOSUB134:P=576:GOSUB134:
S$="*":FORX=224TO544STEP32:P=X:G
OSUB134:P=X+31:GOSUB134:NEXTX:FO
RX=16TO224STEP16:PUT(X,88)-(X+15
,103),FL,PSET:NEXTX
111 DE$(1)="A parked utility tru
ck, under an open window.":DE$(2)
="The bottom of steps leading awa
y to West, an open door to Nort
h.":DE$(3)="Top of flight of ste
ps leading to East, an open doo
r to North.":DE$(4)="The South e
nd of Graham's room."
112 DE$(5)="The South end of a l
ong hallway, open doors to South
and East.":DE$(6)="A long verand
ah stretching form West to East,
open door to North":DE$(7)="The
South-east corner of the ver
andah."
113 DE$(8)="Sonya's end of Graha
m's room, an open door to the Eas
t.":DE$(9)="A long hallway, with
an open door to the West.":D
E$(10)="Storage end of tape-copy
ing room with an open door to Sou
th":DE$(11)="Middle of Eastern v
erandah."
114 DE$(12)="Alex's room, with a
n open door to the East.":DE$(1
3)="Middle of a long hallway, wi
th doors to East and West.":DE$(
14)="Michael's end of tape-copy
ing room, an open door to the
East."
115 DE$(15)="End of Eastern vera
ndah.":DE$(16)="The entrance end
of the kitchen, a door to the Ea
st.":DE$(17)="Middle of a long h
allway.":DE$(18)="An almost empt
y room.":DE$(19)="A narrow bathr
```

```
oom."
116 DE$(20)="The end of the kitc
hen.":DE$(21)="The Northern end
of the hallway, an open door to N
orth.":DE$(22)="An almost empty
room.":DE$(28)="A dark, almost e
mpty room, with a dirt floor and
plenty of dust."
117 DE$(23)="Top of steps leadin
g away to East, open door to
the South.":DE$(24)="Bottom of s
teps to West, opening to South.":
DE$(25)="It is dark, but it feel
s like there are steps made of
boxes.":DE$(26)=DE$(25):DE$(27)
=DE$(25)
118 IFINKEY$<>CHR$(13) THEN118ELS
ERETURN
119 DATA"GET", "DRO", "SOU", "WES",
"EAS", "NOR", "DRI", "DIG", "CLI", "E
AT", "REA", "LOO", "EXA", "ENT", "HEL
", "OPE", "INS", "LIF", "QUI", "BRE",
"INV", "HOO", "TUR", "GO ", "TAK", "P
UT", "TIM"
120 DATA1,0,0,0,0,2,25,0,0,3,3,5
,0,2,0,4,8,0,0,0,5,9,3,6,0,6,10,
0,7,5,7,11,0,0,6,8,0,4,9,0,9,13,
5,0,8,10,14,6,0,0,11,15,7,0,0,12
,0,0,13,0,13,17,9,14,12,14,0,10,
0,13,15,0,11,0,16,20,0,17,0,17
,21,13,18,16,18,22,0,19,17,19,0,
0,0,18,20,0,16,0,0
121 DATA21,23,17,22,0,22,0,18,0,
21,23,0,21,24,0,24,0,28,0,23,25,
0,2,0,26,26,27,0,25,0,27,0,26,0,
0,28,24,0,0,0
122 PCLS1:P=64: S$="BEWDY, BOTTLE
R BONZER! You got":GOSUB134:P=12
8: S$="the fuel & made it on time
.":GOSUB134:P=192: S$="(Martha ev
en smiles at you!)":GOSUB134:P=
256: S$="The winner of the prize
thanks":GOSUB134:P=320:GOTO125
123 PCLS0:P=259: S$=" Thank you f
or visiting the":GOSUB134:P=322:
S$=" Goldsoft Office. Better luc
k":GOSUB134:P=385: S$=" next time
, (if there is one.":GOSUB134:P
=611: S$=" Do you wish to play ag
ain?":GOSUB134
124 IFPEEK(339)=247 THENRUNELSEIF
PEEK(344)=253 THENPOKE113,0: EXEC4
0999 ELSEGOTO124
125 S$="you publicly, & you get
a cheer!":GOSUB134:P=384: S$="Tan
dy even give you a bonus!":GOSUB
134:P=448: S$="I bet you're glad
you went to":GOSUB134:P=524: S$="
GOLDISOFT":GOSUB134
126 P=714: S$="Press Enter":GOSUB
134
127 IFPEEK(338)<>191 THEN127ELSEP
CLS0:P=362: S$=" Play again?":GOS
UB134:GOTO124
128 PCLS1:P=32: S$="You ran out o
f time I'm afraid!":GOSUB134:P=6
4: S$="(and I would be afraid if
I were)":GOSUB134:P=96: S$="you .
. 'cause of M-m-m-martha!":GOSU
B134:GOSUB1:P=160: S$="Oh-oh! Mar
tha Gritwhistle has":GOSUB134:P=
192
129 S$="just arrived from Cococo
nf, she":GOSUB134:P=224: S$="left
```

```

unnoticed to see what was":GOSU
B134:P=256:S$="holding up the pr
ceedings. One":GOSUB134:P=288:S
$="look at you tells the story a
nd":GOSUB134:P=320
130 S$="she starts off on a tira
de con-":GOSUB134:P=352:S$="cern
ing Tandy Computer workers":GOSU
B134:P=384:S$="in generPAINT1 an
d you in particular":GOSUB134:P=
416:S$="Gee, I didn't know you
escaped":GOSUB134:P=448:S$="from
a home for permanent fools)"
131 GOSUB134:P=480:S$="At this s
tage Graham arrives,":GOSUB134:P
=512:S$="strait-jackets Martha &
gets the":GOSUB134:P=544:S$="pr
ize then goes back to Cocomf":
GOSUB134:P=576
132 S$="When you turn up, pale &
shaken,":GOSUB134:P=608:S$="to
work the next day, you're":GOSUB
134:P=640:S$="fired. You never s
hould've gone":GOSUB134:P=672:S$
="to the Goldsoft Office!":GOSUB
134:P=746:S$="Press Enter":GOSUB
134
133 IFPEEK(338)<>191THEN133ELSE1
23
134 G=USR0(P):G$=USR1(S$):RETURN
135 PCLS:P=32:S$="ARRGGHH!! The
smell in this room":GOSUB134:P=6
4:S$="is terrible! You pass out
and":GOSUB134:P=96:S$="wake up m
uch later. However, the":GOSUB13
4:P=128:S$="first face you see i
s Martha's":GOSUB134
136 P=160:S$="who, once she has
ascertained":GOSUB134:P=192:S$="
you are OK starts a tirade on":G
OSUB134:P=224:S$="your stupidity
, your doubtful":GOSUB134:P=256:
S$="lineage and Tandy in general
.":GOSUB134:P=388:S$="You pass b
ack out again.":GOSUB134
137 P=714:S$="Press Enter":GOSUB
134
138 IFPEEK(338)<>191THEN138ELSE1
23
139 P=32:S$="The Coco alerts the
police to":GOSUB134:P=64:S$="yo
ur presence in the office":GOSUB
134:P=96:S$="through Cocomconnect
ion. They":GOSUB134:P=128:S$="do
n't believe your story, and ":GO
SUB134:P=160:S$="take you to the
station."
140 GOSUB134:P=192:S$="By the ti
me they have contacted":GOSUB134
:P=224:S$="Graham the prize-givi
ng has been":GOSUB134:P=256:S$="
ruined. Graham sorts things out"
:GOSUB134:P=288:S$="But ... gulp
... sends over"
141 GOSUB134:P=352:S$="up. She a
rrives on her broom &":GOSUB134:
P=384:S$="immediately points out
to you":GOSUB134:P=416:S$="that
your old folk at home must":GOS
UB134:P=448:S$="have a diet of "
+CHR$(34)+"Doggy Chow"+CHR$(34)+
", and that"
142 GOSUB134:P=480:S$="you have
the IQ of a mentally":GOSUB134:P

```

```

=512:S$="retarded earthworm. You
then":GOSUB134:P=544:S$="flee f
or your sanity's sake to":GOSUB1
34:P=576:S$="the mountains, scre
aming loudly":GOSUB134:P=608
143 S$=CHR$(34)+"I surrender! I
surrender!"+CHR$(34):GOSUB134:P=
714:S$="Press Enter":GOSUB134:P=
320:S$="Martha Gritwhistle to pi
ck you":GOSUB134
144 IFPEEK(338)<>191THEN144ELSEG
OTO123
145 IFLC=1THENLINE(0,0)-(256,94)
,PSET,BF:DRAW"C5BM164,8D20M+16,+
8U20M-16,-8":PAINT(166,10),5,5:D
RAW"COBM167,30M+3,-6BM174,33M+3,
-6BM168,28M+6,+4":RETURN
146 LINE(48,0)-(208,72),PSET,B:L
INE-(256,94),PSET:LINE(48,72)-(0
,92),PSET:IFLC=2THENGOSUB157:GOS
UB167:RETURNELSEIFLC=3THENGOSUB1
57:GOSUB168:LINE(188,72)-(208,76
),PSET,B:DRAW"BM208,72M-8,-8L20N
M+8,+8D8M+7,+4":PAINT(182,70),0,
0:PAINT(183,73),0,0:RETURN
147 IFLC=4THENDRAW"COBM16,34D20M
+20,-9U20M-20,+9":PAINT(18,36),0
,0:IFCN=0THENDRAW"C5BM24,50U8D4M
+10,-5U4D8C0":LINE(104,40)-(152,
72),PSET,BF:P=205:S$="copier":GO
SUB134:LINE(124,72)-(132,74),PSE
T,BF:CIRCLE(128,76),7,0,.5:PAINT
(128,76),0,0:RETURN
148 IFLC=4THENLINE(56,40)-(104,7
2),PSET,BF:P=199:S$="copier":GOS
UB134:GOSUB169:LINE(76,72)-(84,7
4),PSET,BF:CIRCLE(80,76),7,0,.5:
PAINT(80,76),0,0:RETURN
149 IFLC=5THENGOSUB155:GOSUB169:
RETURNELSEIFLC=6THENPUT(84,28)-
(99,43),LT,PSET:GOSUB157:GOSUB16
7:GOSUB168:RETURNELSEIFLC=7THENG
OSUB169:GOSUB167:GOSUB170:RETURNE
LSEIFLC=8THENGOSUB155:PUT(32,33)
-(96,80),CC,PSET:RETURN
150 IFLC=9THENGOSUB156:GOSUB169:
RETURNELSEIFLC=10THENGOSUB169:PU
T(32,33)-(96,80),CC,PSET:RETURNE
LSEIFLC=11THENGOSUB169:GOSUB170:
RETURNELSEIFLC=12THENGOSUB155:PU
T(32,33)-(96,80),CC,PSET:RETURN
151 IFLC=13THENGOSUB155:GOSUB156
:GOSUB169:RETURNELSEIFLC=14THENG
OSUB156:RETURNELSEIFLC=15THENGOS
UB170:RETURNELSEIFLC=16THENGOSUB
155:GOSUB169:IFLEFT$(OB$(16,1),3
)="CHA"THENPUT(104,33)-(135,80),
CH,PSET:RETURNELSEPUT(104,33)-(1
34,80),CH,PSET:RETURN
152 IFLC=17THENGOSUB156:GOSUB169
:GOSUB168:RETURNELSEIFLC=18THENG
OSUB167:GOSUB155:GOSUB169:RETUR
NELSEIFLC=19THENGOSUB156:LINE(48,
0)-(208,72),PSET,BF:P=140:S$=" C
ENSORED":GOSUB134:RETURNELSEIFLC
=21THENGOSUB157:GOSUB168:LINE(64
,16)-(92,64),PSET,B:RETURN
153 IFLC=22THENGOSUB167:RETURNE
LSEIFLC=23THENGOSUB168:RETURNELSE
IFLC=24THENGOSUB167:RETURNELSEIF
LC=25ORLC=26ORLC=27THENLINE(0,0)
-(256,94),PSET,BF:RETURNELSEIFLC
=28THENGOSUB157:RETURN
154 RETURN

```

```

155 DRAW"COBM220,78U58M+20,+9D58
":PAINT(222,60),0,0:RETURN
156 DRAW"COBM16,86U58M+20,-9D58"
:PAINT(18,82),0,0:RETURN
157 LINE(112,20)-(144,72),PSET,B
F:RETURN
158 OPEN"1",#1,"S.DAT":DIMDK(6),
LT(6),RP(6),KY(6),FB(6),LN(6),HP
(6),AF(6),SK(6),FL(6),SD(6),CF(6
),CC(77)
159 FORX=32TO208STEP16:I=0:T$="
":P=0:Y=80
160 IFX=48THENP=VARPTR(DK(0))ELS
EIFX=64THENP=VARPTR(LT(0))ELSEIF
X=80THENP=VARPTR(RP(0))ELSEIFX=9
6THENP=VARPTR(KY(0))ELSEIFX=112T
HENP=VARPTR(FB(0))ELSEIFX=128THE
NP=VARPTR(LN(0))
161 IFX=144THENP=VARPTR(HP(0))EL
SEIFX=160THENP=VARPTR(AF(0))ELSE
IFX=176THENP=VARPTR(SK(0))ELSEIF
X=192THENP=VARPTR(FL(0))ELSEIFX=
208THENP=VARPTR(SD(0))ELSEIFX=32
THENP=VARPTR(CF(0))
162 FORI=0TO32:INPUT#1,T$:POKEP+
I,VAL("&H"+T$):NEXTI,X:CLOSE#1:G
OTO171
163 FORI=1TO5:IFOB$(LC,I)=""THEN
166ELSEF$=LEFT$(OB$(LC,I),3):IFF
$="ROP"THENPUT(80,80)-(95,95),RP
,PSETELSEIFF$="KEY"THENPUT(96,80)
-(111,95),KY,PSETELSEIFF$="FLA"
THENPUT(112,80)-(127,95),FB,PSET
164 IFF$="DIS"THENPUT(48,80)-(63
,95),DK,PSETELSEIFF$="MEM"ORF$="
PIE"ORF$="NOT"THENPUT(64,80)-(79
,95),LT,PSETELSEIFF$="LIN"THENPU
T(128,80)-(143,95),LN,PSETELSEIF
F$="HAI"THENPUT(144,80)-(159,95)
,HP,PSET
165 IFF$="AIR"THENPUT(160,80)-(1
75,95),AF,PSETELSEIFF$="STI"THEN
PUT(176,80)-(191,95),SK,PSETELSE
IFF$="FUE"THENPUT(192,80)-(207,9
5),FL,PSETELSEIFF$="SAN"THENPUT(
208,80)-(223,95),SD,PSETELSEIFF$
="MUG"THENPUT(32,80)-(47,95),CF,
PSET
166 NEXT:RETURN
167 P=163:S$="?":GOSUB134:RETURN
168 P=189:S$="?":GOSUB134:RETURN
169 P=175:S$="?":GOSUB134:RETURN
170 LINE(208,40)-(256,60),PSET:P
AINT(212,4),0,0:POKE178,9:PAINT(
222,50),4:POKE178,0:RETURN
171 OPEN"1",#1,"S1.DAT":I=0:T$="
":P=0:P=VARPTR(CC(0)):FORI=0
TO390:INPUT#1,T$:POKEP+I,VAL("&H
"+T$):NEXTI:CLOSE#1:OPEN"1",#1,"
S2.DAT":DIMCH(39):I=0:T$="":
P=0:P=VARPTR(CH(0)):FORI=0TO191:
INPUT#1,T$:POKEP+I,VAL("&H"+T$):
NEXTI:CLOSE#1:RETURN

```



Put your program up in lights . . .

# MARQUEE

By Chuck Poynter

You just wrote a computer program that will set the world on fire - but something is missing. Do your title and menu screens lack pizzazz? If that is what you need, then MARQUEE should help.

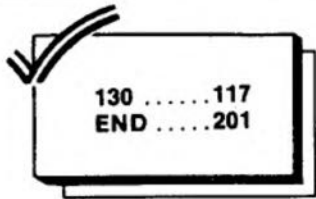
Listing 1 pokes a machine code program into memory that draws and moves a border around the screen. This is actually an illusion. What really happens is the graphics characters are changing back and forth, and it appears to be

moving. The 150 in Line 50 is the yellow graphics character you start with. It changes to 153 during the program, then changes back to 150. To get the following colours, change 150 to one of the following numbers: red, 182; buff, 198; cyan, 214; orange, 246; blue, 166; and magenta, 230. You can use any character code from zero to 252. Try these and see the effects.

There is a title and menu screen in the program starting at Line

170. Start your program here and include EXEC 32000 when you want to call the program. The ENTER, '1', '2' and '3' keys return you to the BASIC program. If you need more than three items in your menu, just break it into two menu screens.

Listing 2 is the source code for the machine language program. This is included so you can see how the program works. Both programs are well - documented so they don't need extensive explanation.



130 .....117  
END .....201

## Listing 1: MARQUEE

```
5 REM*CAN BE PLACED ANY WHERE IN
  RAM CHANGE 32000 TO ANY RAM
  LOCATION ADD 90 TO IT FOR THE
  SECOND NUMBER
10 FOR I=32000 TO 32099
20 READ X
30 POKE I,X
40 NEXT I
45 REM*CHANGE THE 150 IN LINE 50
  TO ANY CHARACTER CODE YOU WANT
50 DATA 198, 150, 247, 125, 89,
  173, 159, 160
60 DATA 0, 129, 13, 39, 57, 129,
  49, 39
70 DATA 53, 129, 50, 39, 49, 129
  , 51, 39
80 DATA 45, 142, 4, 0, 231, 128,
  140, 4
```

```
90 DATA 33, 38, 249, 142, 5, 223
  , 231, 128
100 DATA 140, 6, 0, 38, 249, 142
  , 4, 63
110 DATA 231, 128, 231, 128, 48,
  136, 30, 140
120 DATA 5, 223, 38, 244, 241, 1
  25, 89, 39
130 DATA 8, 241, 125, 90, 39, 11
  , 151, 135
140 DATA 57, 203, 3, 247, 125, 9
  0, 126, 125
150 DATA 5, 192, 3, 247, 125, 89
  , 126, 125
160 DATA 5, 18, 18, 255, 0, 255,
  0, 255
170 CLS 4
180 REM*PUT YOUR TITLE PAGE HERE
190 POKE 359,57:SCREEN0,1
200 PRINT@106,"YOUR TITLE";
210 PRINT@168,"BY";CHR$(191);"WH
  O";CHR$(191);"DONE";CHR$(191);"I
  T";
220 PRINT@452,"PRESS <ENTER> TO
  CONTINUE";
230 EXEC32000
240 CLS 5
250 REM*PUT YOU MENU HERE
260 PRINT@70,STRING$(19,32);
```

```
270 PRINT@70+32,"(1) MENU ITEM O
  NE ";
280 PRINT@70+64,"(2) MENU ITEM T
  WO ";
290 PRINT@70+96,"(3) MENU ITEM T
  HREE";
300 PRINT@70+128,STRING$(19,32);
310 EXEC32000
320 POKE359,126:SCREEN0,1
330 REM*CHANGE GOTO TO YOUR LINE
  NUMBERS
340 IF PEEK(135)=49 THEN GOTO 39
  0
350 IF PEEK(135)=50 THEN GOTO 41
  0
360 IF PEEK(135)=51 THEN GOTO 43
  0
370 IF PEEK(135)=13 THEN GOTO 31
  0
380 REM*YOUR PROGRAM STARTS HERE
390 CLS:PRINT"YOU HAVE SELECTED
  MENU ITEM (1)";
400 END
410 CLS:PRINT"YOU HAVE SELECTED
  MENU ITEM (2)";
420 END
430 CLS:PRINT"YOU HAVE SELECTED
  MENU ITEM (3)";
440 END
```



Listing 2:

7D00		00100	ORG	\$7D00	CAN BE ANY WHERE IN RAM
7D00 C6	96	00110	LDB	#150	LOAD BORDER CODE
7D02 F7	7D59	00120	STB	STORE	STORE BORDER CODE IN RAM
7D05 AD	9F A000	00130	START JSR	[\$A000]	CHECK KEYBOARD
7D09 81	0D	00140	CMPA	#13	IS ENTER KEY PRESSED
7D0B 27	39	00150	BEQ	END	IF SO RETURN TO BASIC PROGRAM
7D0D 81	31	00160	CMPA	#49	IS THE 1 KEY PRESSED
7D0F 27	35	00170	BEQ	END	IF SO RETURN TO BASIC
7D11 81	32	00180	CMPA	#50	IS 2 KEY PRESSED
7D13 27	31	00190	BEQ	END	IF SO RETURN TO BASIC
7D15 81	33	00200	CMPA	#51	IS 3 KEY PRESSED
7D17 27	2D	00210	BEQ	END	IF SO RETURN TO BASIC
7D19 8E	0400	00220	LDX	#\$400	LOAD START OF SCREEN
7D1C E7	80	00230	DRAW1 STB	,X+	STORE CHARACTER ON SCREEN
7D1E 8C	0421	00240	CMPX	#\$421	COMPARE FIRST LINE PLUS 1
7D21 26	F9	00250	BNE	DRAW1	IF NOT EQUAL DO MORE
7D23 8E	05DF	00260	LDX	#\$5DF	FIRST CHARACTER BOTTOM LINE
7D26 E7	80	00270	DRAW2 STB	,X+	STORE CHARACTER
7D28 8C	0600	00280	CMPX	#\$600	END OF TEXT SCREEN
7D2B 26	F9	00290	BNE	DRAW2	IF NOT END CONTINUE
7D2D 8E	043F	00300	LDX	#\$43F	LAST CHARACTER SECOND LINE
7D30 E7	80	00310	DRAW3 STB	,X+	STORE CHARACTER
7D32 E7	80	00320	STB	,X+	ONE MORE
7D34 30	88 1E	00330	LEAX	30,X	ADD 30 TO SCREEN POSITION
7D37 8C	05DF	00340	CMPX	#\$5DF	LAST POSITION LINE 15
7D3A 26	F4	00350	BNE	DRAW3	DO UNTILL EQUAL
7D3C F1	7D59	00360	CMPB	STORE	IS CHARACTER SAME AS ONE IN RAM
7D3F 27	08	00370	BEQ	CHR1	IF EQUAL GET NEW CHARACTER
7D41 F1	7D5A	00380	CMPB	STORE+1	IS CHR SAME AS ONE IN RAM
7D44 27	0B	00390	BEQ	CHR2	IF EQUAL GET NEW CHR
7D46 97	87	00400	END STA	135	STORE KEY PRESSED IN RAM
7D48 39		00410	RTS		RETURN TO BASIC PROGRAM
7D49 CB	03	00420	CHR1 ADDB	#3	CHANGE CHR BY +3
7D4B F7	7D5A	00430	STB	STORE+1	STORE NEW CHR IN RAM
7D4E 7E	7D05	00440	JMP	START	START OVER
7D51 C0	03	00450	CHR2 SUBB	#3	SUBTRACT 3 FROM CHR
7D53 F7	7D59	00460	STB	STORE	STORE NEW CHR IN RAM
7D56 7E	7D05	00470	JMP	START	START OVER
7D59 12		00480	STORE NOP		CHR STORAGE AREA 1
7D5A 12		00490	NOP		CHR STORAGE AREA 2
	0000	00500	END		

000000 TOTAL ERRORS

Computer Supplies & Secretarial Supplies

- \* Computer Furniture
- \* Printer Requisites
- \* Printer Wheels (most styles)
- \* Floppy Disks - all sizes.
- \* Binders & Disk Storage Boxes
- \* Business Software

All at VERY competitive prices!

# WORDPRO

Shop 1/115 Scarborough St.,  
Southport. Qld. 4215. 075-32-0355

**NEW!!** **\$39-95**  
PAIR

Score higher in all your games with -

## DRAGON JOYSTICKS

Have many features of expensive joysticks -

- \* Sturdy construction
- \* Potentiometers for maximum precision  
NOT the imprecise leaf switches used in cheaper models
- \* High quality anti-RFI cable
- \* Positive Fire Button control

NOTE: these joysticks do NOT have self centring or line tuning controls.

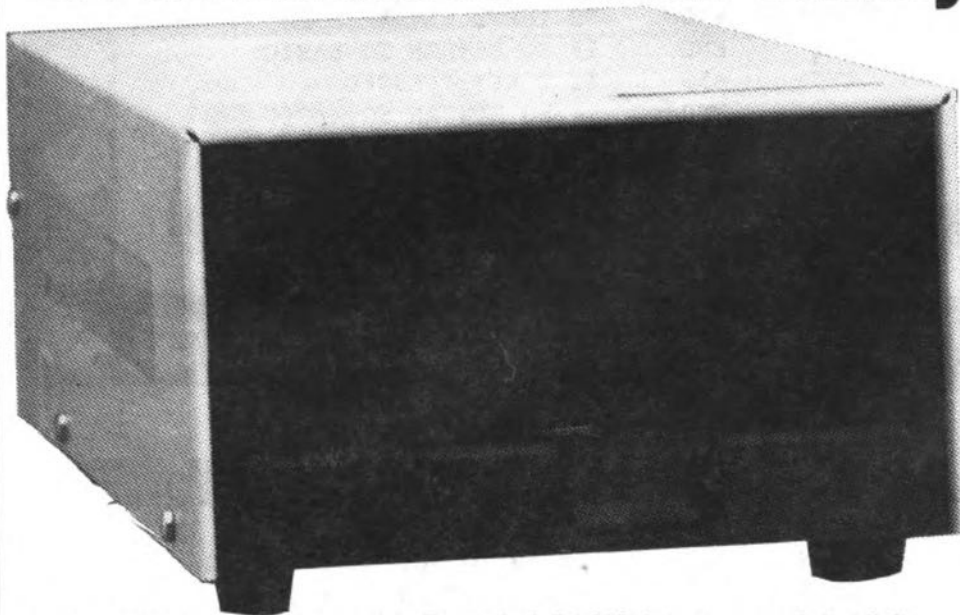
ORDERS with CHEQUE or MONEY ORDER TO:

EASE Services,  
19 Acacia Avenue,  
Blackburn, 3130

**EASE**  
MONEY BACK IF NOT SATISFIED

# Tandy ELECTRONICS

## Increases the Uses of Your Color Computer with Tandy Hardware



- Stores Over 156,00 Characters of Data
- Slashes Time Taken to See, Save and Load Programs

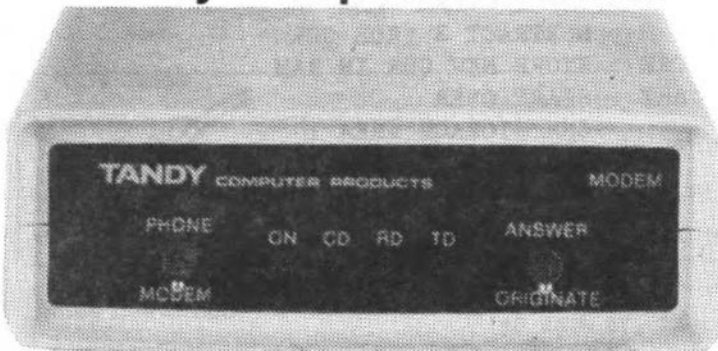
# Save \$150

Reg 599.00

Turn any Color Computer with Extended BASIC into a complete disk system. You can write your own sophisticated disk applications or add ready-to-run software from our huge range. The drive plugs into the Program Pak port or MultiPak Interface. Includes 13.3cm double-density, 35-track floppy disk drive, Program Pak cartridge, cable, one blank 13.3cm diskette as well as full instructions and a handy reference manual. Provision has been made for adding a second drive. 26-3129

# \$449

### Communicate with Tandy Computer Products



# Save \$60

Reg 259.95

# 199<sup>95</sup>

A modem translates the electronic impulses that make up computer information into tones that can be sent over telephone lines. The Tandy V-21 Modem plugs into any RS-232 equipped computer, the ideal "add-on" to any system. It connects you to such services as Stars Database Information Express, Bulletin Boards. 300 Baud and completely Telecom approved. 26-9403

### Gain Access to VIATEL\*



\*Optional Software Required

Telecom  
**VIATEL**  
AUSTRALIA'S NATIONAL VIDEOTELEX SERVICE

# 399<sup>95</sup>

Get into the biggest information service in Australia with our new Modem. Run banking, book airline tickets etc. from your living room.

26-9404

Ask Your Store About Our **VIATEL** Software for  
Your Color Computer, Model 4, Tandy 1000  
and Model 2000

## Tandy Touch Pad

Reg  
129.95

**79<sup>95</sup>** Save \$40

The Tandy Touch Pad allows you to use your computer like an easel. Trace over designs, write some "hand written" letters, even create your own masterpiece! Comes in diskette format with pad, tracing pen and full instructions. A must for anyone wishing to extend their computer's capabilities. 26-1185

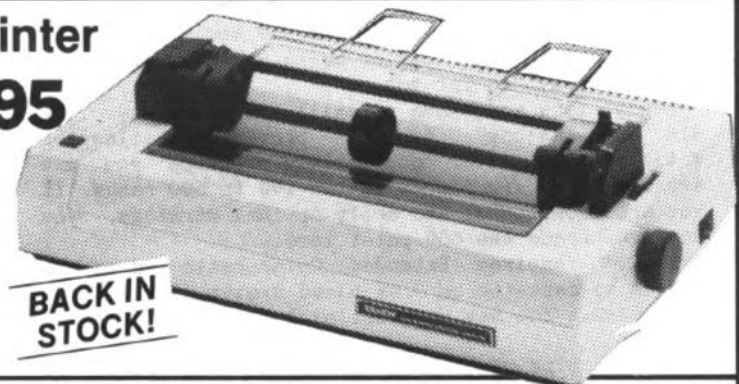


## Low-Priced Impact Matrix Printer

- DMP 105
- High Performance for Less
- 80 Characters per Second

**349<sup>95</sup>**

Low-Cost complement to your Color Computer or Model III/4. Print quality rivals printers at twice its price. Features elongated and condensed type, underline and bold face. Graphics mode allows 480-800 dots per line. Includes parallel and Color Computer-compatible serial interfaces. 26-1276

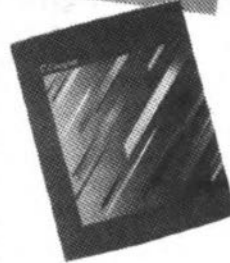


**BACK IN STOCK!**

## Ready-to-Run Disk Software for Your Color Computer



**OS-9** Accesses entire 64K memory of the computer. With documentation and reference manual. 26-3030 ..... **119.95**  
**Flight Simulator** Learn how to fly a plane. Do landings and take-offs too. 26-3108 ..... **49.95**  
**Color Profile** Your own personal filing system. Create up to 256 different data fields. 26-3253 ..... **109.95**  
**SCRIPSIT** Easy-to-use word processing system. Insert, duplicate, delete, move words in seconds. 26-3255 ..... **99.95**  
**Shamus** Try and find your way through a complete labyrinth of different passages. 26-3289 ..... **59.05**  
**Cookbook** Two programs in one! Menu planner suggests entire meals. An index file stores recipes for future reference. 26-3257 ..... **69.95**



**A. C-Compiler** allows the easy-to-use "C" language to be used with the OS-9. 26-3038 ..... **179.95**  
 Also available, **Pascal OS-9** 26-3034 ..... **179.95**  
**Dynacalc**, a powerful electronic spreadsheet. 26-3275 ..... **69.95**  
**B. Color Dictionary** Use with SCRIPSIT to check for spelling errors automatically. Makes the usual checking time completely obsolete. Quick and efficient. 26-3265 ..... **109.95**

**WE SERVICE WHAT WE SELL!**

**Tandy**  
**ELECTRONICS**

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

Nearly  
350 Stores  
Australia-  
Wide

Available from  
**350 Stores Australiawide**  
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*

## A CHALLENGING BANDY

## WORD GAME

By H. Allen Curtis

Do you like to toss words around? Would you like to become a superstar or maybe even the greatest? If so, "Bandy" is for you.

"Bandy" is a word game of skill and strategy. The aim of "Bandy" is to provide challenging entertainment. After considerable practice you may expect to attain scores in the 400 to 500 range. If you are exceptional and apply optimal strategy, you may even reach the 600 point level.

"Bandy" requires Extended Color Basic and can be played on cassette or disk-based system with 16K or 32K RAMs. You may save unfinished games on tape or disk.

The mechanics of playing "Bandy" and the rules of word formation and scoring will be presented in detail. You may prefer to glance over them or even skip them entirely until you have experimented with the game.

## THE MECHANICS

You play the game by forming interconnecting words crossword-like on the "Bandy" board. Each letter in every word you form adds one point to your score. Also, you can earn bonuses for forming words more than six letters long. Turn by turn you form words on the board by selecting letters from a "play-line" of letters. After each turn, the play-line is replenished with letters from a 63-letter "supply".

Having the "Bandy" letter supply in full view and knowing the order in which letters will be available for play lets you plan strategy for turns in advance.

The letter supply occupies the top two lines of your CoCo screen. A blue marker separates the letters that have been sent to the play-line from those that will be sent. Figure 1 shows an example letter supply as it would be located in a game of "Bandy".

The play-line is an eight-position line of letters located right of the screen and below the letter supply. The first seven positions contain letters from the letter supply and the eighth position contains a red marker. The eighth position is called the choice position because you may choose any letter to replace the red marker. Figure 2 shows an example of the play-line and the letter supply locations. The "Bandy" board occupies most of the left half of the screen. It consists of 180 spaces and each space is designated by a plus sign. The board has 12 rows and 15 columns. Thus, a 15-letter word is the largest you can form across. A 12-letter word is the longest word you can form down. Figure 3 shows the "Bandy" board relative to the letter supply and the play-line.

At the start of the game the cursor is found at the top left space of the board. At the beginning of each turn move the cursor to the position you wish to start forming a word. You control the cursor by typing the direction arrow keys; left, right, up, and down.

After you have moved the cursor to the desired starting position and are ready to form a word, type 'A' or 'D' depending on whether you wish to form the word across (from left to right) or down (from top to bottom). As you type 'A' or 'D' the word, across or down, will be displayed below the play-line.

If you type a letter other than 'A' or 'D' when neither across nor down is displayed, an alarm will be given. The alarm is in form of help instructions summarizing the correct order of play.

To place a letter from the play-line at the cursor position of the board, merely type that letter. After the letter has been placed, the cursor will automatically move to the next available position to the right or towards the bottom, depending on whether you are forming a word across or down. Above the play-line letter just placed on the board will appear a minus sign to indicate that the letter has been "played".

Even if a letter appears in only one position of the play-line, it can be played as many times as you need to use it in forming a word during a turn. This greatly enlarges the possibilities for words to be formed. After you complete the turn, all played letters will be replaced with letters from the letter supply.

There are two ways of completing a turn. Usually you complete a turn, after forming a word, by pressing the ENTER button. A turn is automatically completed when there is no available space for the cursor to move to the right or down for an across or down formed word.

To place a letter in the "choice" position of the play-line you must do the following: When neither across nor down is displayed depress the Space Bar. This will cause the cursor to replace the red marker. When you see the cursor there, type the letter of your choice and it will appear in the choice position. The letter will be "reversed," that is, it will be green with a dark background.

The reversed letter is placed on the board in the same way as any other play-line letter. Do *not* switch to the "upper-lowercase" mode to play the reversed letter! Also, like the other play-line letters, it may be played as many times per turn as is needed. The chosen letter will be replaced by the red marker when the turn is over.

While across or down is displayed, the direction arrow

EFRETOS | GMEDNLUBRHYTACUSKJEIQDET  
 LENOPXAUENZWBJTSEBOECFNRKTLIGNRE

Figure 1

keys can *not* be used to control the cursor. This does not mean that if you made a mistake in forming a word that you are "stuck" with it. You may erase a completed turn by pressing the CLEAR key. After completing a turn, you may see that you would have scored better by playing the turn differently. In such a case, erase the turn and replay it.

A beep alarm is sounded when you make certain illegal moves such as pressing the Space Bar when there is a letter in the choice position or typing a character not on the play-line when across or down is displayed.

The game automatically ends after the letter supply is exhausted. You may also terminate the game prematurely as follows: when neither across nor down is displayed, you can end the game by typing the '@' key.

When the game is terminated, you are given five options:

- 1) To replay the game that you just finished.
- 2) To play a game with another letter supply.
- 3) To end the program.
- 4) To save your game on disk or tape.
- 5) To view screen of just completed game.

At the very beginning of the game if the letter supply is not to your liking, merely press ENTER to change it.

At the start of *Bandy* you are asked whether you have a previously saved game to load. If you answer by typing 'Y', the program will load the game and set it to start at the point that you discontinued it.

### Rules For Word Formation

Now that you know the mechanics of play, you need to know how to interconnect and modify words to form new words. The rules for forming words are as follows:

EFRETOS | GMEDNLUBRHYTACUSKJEIQDET  
 LENOPXAUENZWBJTSEBOECFNRKTLIGNRE

EFRETOS |

Figure 2

1) On your first turn, anywhere on the board form any valid dictionary word consisting of two or more letters.

2) Form a valid dictionary word by adding one or more letters to a word already on the board. The letter or letters can be added at the beginning, the end, or both the beginning and end of a word already on the board.

3) Form a valid dictionary word by adding one or more letters at right angles to an existing word. The letter or letters can be added at right angles to either side or both sides of the existing word. If one of the letters adds to the beginning or end of the existing word, another new word is formed. It also must be a valid dictionary word and you will be given credit for both words formed.

4) Form a valid dictionary word adjoining or parallel to an existing word so that all adjoining letters form valid dictionary words. You receive credit for all words formed.

5) Form a valid dictionary word that touches no other word. This word like all other words must consist of at least two letters.

To crystallize in your mind the foregoing rules, some examples will now be presented.

**Example 1:** Suppose the word "FRIEND" is on the *Bandy* board, and the letters L, N, U and Y are among those on the play-line. You could, according to rule 2, add "LY" to the end of "FRIEND" to form the word "FRIENDLY." Better yet, you could add "UN" to the beginning of "FRIEND" and "LY" to the end of "FRIEND" to form the word "UN-FRIENDLY" in one turn. It has been assumed that there were available spaces at the places where the letters were played. Similar assumptions will be made in the ensuing examples.

**Example 2:** Suppose the word "ARK" is on the board and the letters I, N, E, D are on the play-line. By rule 3, you could form the word "INKED" by adding at right angles to "ARK" the letters I and N before K and the E and D after K. By rule 3, you could form two words, "DINE" and "DARK." If "ARK" had been across, the result of taking your turn could have been:

```
D A R K
I
N
E
```

**Example 3:** Suppose the words "FLOWER" and "TENT" are on the board as shown:

```
F
L
O
W
T E N T
R
```

Also, suppose the letters A, T and I are on the play-line. According to rule 4, the words "AWAIT," "AT," "AN" and "IT" can be formed. The result of playing according to rule 4 would be:

```
F
L
O
A W A I T
T E N T
R
```

After each turn, a new display temporarily replaces the game playing screen. The new display lets you see more easily whether or not you followed the rules of word formation. The display will list every word you formed during the just completed turn. If you see that any of the listed words are not valid, you should erase the turn and play it correctly.



```

540 DATA 32,66,1F,12,8E,4,F5,CC
550 DATA CF,6,A7,80,5A,26,FB,5A
560 DATA 36,4,BE,3B,3,EC,41,10
570 DATA BF,3B,7,FF,3B,3,A3,1E
580 DATA 1F,12,CE,3B,A,4D,26,15
590 DATA 5D,26,14,34,10,AE,3E,E6
600 DATA 1F,C1,5F,25,2,E6,1,35
610 DATA 10,C1,5F,25,6,20,68,C1
620 DATA 12,22,64,AE,1E,E6,82,C1
630 DATA 5B,25,FA,30,1,E6,80,C1
640 DATA 5A,22,F,8D,49,20,F6,35
650 DATA 2,4A,27,6,C6,20,E7,C0
660 DATA 20,6,C6,D,E7,C0,86,2,34
670 DATA 2,1F,21,E6,82,C1,FF,27
680 DATA 5B,30,1F,1F,12,4F,AE,84
690 DATA 30,88,20,E6,84,4C,C1,5B
700 DATA 25,F6,30,88,E0,E6,84,4C
710 DATA C1,5B,25,F6,81,3,27,D9
720 DATA 30,88,20,E6,84,C1,5A,22
730 DATA BD,8D,2,20,F3,C1,20,22
740 DATA 2,CB,40,E7,C0,39,AE,1E
750 DATA 30,88,E0,E6,84,C1,5B,25
760 DATA F7,30,88,20,E6,84,C1,5A
770 DATA 22,4,8D,E1,20,F3,C6,D
780 DATA E7,C0,86,2,34,2,1F,21
790 DATA E6,82,C1,FF,27,32,30,1F
800 DATA 1F,12,4F,AE,84,30,1,E6
810 IFB<>67098THENCLS:PRINT" DA
TA ENTRY ERROR IN LINES 430 TH
ROUGH 800.":STOP
820 FORI=X+611 TOX+918:READA$:A=
VAL("&H"+A$):POKEI,A:B=B+A:NEXT
830 DATA 84,4C,C1,5B,25,F7,E6,82
840 DATA 4C,C1,5B,25,F9,81,3,27
850 DATA DD,30,1,E6,84,C1,5A,22
860 DATA 4,8D,AA,20,F4,35,2,4A
870 DATA 27,C4,C6,20,E7,C0,20,C4
880 DATA 5F,E7,C0,35,2,8E,3B,A
890 DATA 8D,18,C1,7,25,FA,CB,5
900 DATA C1,14,25,F4,8D,C,CB,A
910 DATA C1,30,25,F8,8D,4,CB,F
920 DATA 20,FA,A6,80,27,E,81,21
930 DATA 25,2,5C,39,4F,ED,C1,32
940 DATA 62,5F,20,D4,32,62,4F,ED
950 DATA C1,33,5F,8E,FF,FF,30,1
960 DATA EC,C3,26,FA,33,41,E3,C1
970 DATA 30,1F,26,FA,FE,3B,5,36
980 DATA 6,FF,3B,5,BD,B4,F4,5F
990 DATA 86,4,1F,1,CE,3B,D1,BD
1000 DATA A5,9A,7E,A5,9A,8E,3B,9
1010 DATA BD,B9,9C,7E,B4,F4,8E
1020 DATA 3B,D1,5F,86,4,1F,3,20
1030 DATA E7,CE,4,3F,5F,8E,4,71
1040 DATA A6,C5,81,AF,26,13,A6
1050 DATA 80,8C,4,78,27,7,81,6D
1060 DATA 26,F5,5A,20,F2,2B,8,16
1070 DATA FD,73,5C,C1,7,26,E2,8E
1080 DATA 4,71,CE,4,1,C6,7,86,6D
1090 DATA A1,80,27,15,5A,26,F9
1100 DATA 86,CF,A7,84,86,BF,A7
1110 DATA 88,20,FE,3B,3,BE,3B,7
1120 DATA 16,FC,C3,86,AF,A1,C0
1130 DATA 26,FC,A6,C4,81,CF,26,7
1140 DATA A7,88,1F,33,5F,20,B,A7
1150 DATA 88,1F,A7,5F,86,AF,A7
1160 DATA C4,86,CF,A7,1F,86,6D
1170 DATA 20,C7,C6,6B,CE,3D,FF
1180 DATA FF,3B,3,33,5E,A6,5F,E7
1190 DATA D4,81,FF,26,F6,C6,8,CE
1200 DATA 4,71,86,CF,A7,C0,5A,26
1210 DATA FB,86,BF,B7,4,98,20,AE
1220 DATA 0,0,0
1230 IFB<>103022THENCLS:PRINT"
DATA ENTRY ERROR IN LINES 830
THROUGH 1220.":STOP

```

Listing 2: BANDY2

```

10 GOTO30
20 GOTO40
30 PCLEAR4:GOTO20
40 CLS:PRINT@225,"DO YOU WANT TO
BE A SUPER ...":V=60:GOSUB390:C
LS:PRINT@234,"IF SO ...":FORI=0T
O1500:NEXT
50 CLEAR190,15104:S=256*PEEK(27)
+PEEK(28)-919:DEFUSR=S:DEFUSR1=S
+752:DEFUSR2=S+761:DEFUSR3=S+771
:DEFUSR5=S+879:S=15871
60 F$="EFAETOETAETIESAETOEORETSE
IAEODENLEGREHTEYSECIEFAEBOEMDELN
EPUEBOEHYETSECFEIAGMEODENLEPBUR
HYTACURTISMNDLKJWVLDNMOGAFICSTYH
RUBPQKDNJMOWGAVLFICXSTYZHRUBP":C
LS:PRINT@37,"YOUR SCORE KEEPER N
EEDS":PRINT@67,"THE FOLLOWING IN
FORMATION.":PRINT
70 PRINT@132,"YOUR NAME? ";:A$="
"
80 GOSUB380:IFASC(X$)<>13THENA$=
A$+X$:PRINTX$;:IFLEN(A$)<14THEN8
0
90 A=RND(-TIMER):PRINT:PRINT:PRI
NT"DO YOU WISH TO LOAD A PREVIOU
SLYAVED GAME? (Y/N)"
100 GOSUB380:IFX$<"Y"THEN160ELS
E IFPEEK(188)=6THEN130
110 GOSUB570
120 GOSUB380:IFX$="D"THEN520ELSE
IFX$<"T"THEN120
130 CLS:MOTORON:PRINT@73,"POSITI
ON TAPE.":PRINT@101,"TYPE L & DE
PRESS play."
140 GOSUB380:IFX$<"L"THEN140
150 CLS:PRINT"LOADING":OPEN"1",-
1,"B":INPUT#-1,A$,E$,Z:CLOSE#-1:
CLOADM:GOTO530
160 Z=0:GOSUB360:PRINT@344-LEN(S
TR$(Y)),Y;:PRINT@305+INT(7-.5*LE
N(A$)),A$;
170 A=USR(0)
180 IFPEEK(S)=254THEN230ELSEIFA=
0THEN450ELSEGOSUB370:C=USR1(0)
190 GOSUB380:IFASC(X$)=12THEN220
ELSEIFASC(X$)<>13THEN190
200 C=USR2(0):Z=A+Z
210 PRINT@344-LEN(STR$(Z)),Z;:A=
USR3(0):GOTO180
220 C=USR2(0):A=USR5(0):GOTO180
230 CLS:PRINT@80-LEN(A$)/2,A$;:P
RINT@104,"YOUR SCORE IS ";Z;".":
PRINT" YOU";:IFZ<200THENV=20:
PRINT" EARNED A ...":ELSEIFZ<300T
HENV=35:PRINT"RE ALMOST A ...":E
LSEIFZ<400THENV=50:PRINT"RE A .
...":ELSEIFZ<500THENV=60:PRINT"RE
A SUPER ..."
240 IFZ>499THENV=70:PRINT"RE TH
E GREATEST ..."
250 GOSUB390
260 PRINT@101,"YOU HAVE 5 OPTION
S":PRINT" 1) REPLAY GAME JUST F
INISHED.":PRINT" 2) PLAY ANOTHER
GAME.":PRINT" 3) TERMINATE PLAY
.":PRINT" 4) SAVE GAME ON TAPE O
R DISK."
270 PRINT" 5) VIEW FINISHED GAME
.":PRINT@356,"TO TAKE AN OPTION
, TYPE THE ASSOCIATED N
UMBER."

```

```

280 GOSUB380:X=VAL(X$):IFX=1THEN
CLS:Z=0:PRINT@0,E$;:POKE1031,17
5:PRINT@145,LEFT$(E$,7)+CHR$(191
);:PRINT@305+INT(7-.5*LEN(A$)),A
$;:PRINT@344-LEN(STR$(Z)),Z;:FOR
I=97TO449STEP32:PRINT@I,STRING$(
15,43);:NEXT:GOTO170
290 IFX=2THENZ=0:GOTO160ELSEIFX=
3THENGOTO510ELSEIFX=5THEN490ELSE
IFX<4THEN280
300 IFPEEK(188)<>6THEN540
310 CLS:PRINT"TYPE M. POSITION
TAPE. THEN TYPE R & DEPRESS re
cord BUTTONS. TYPE S."
320 GOSUB380:IFX$<"M"THEN320
330 MOTORON:GOSUB380:IFX$<"R"TH
EN330
340 MOTOROFF:GOSUB380:IFX$<"S"
HEN340
350 CLS:PRINT"SAVING":OPEN"0",-
1,"B":PRINT#-1,A$,E$,Z:CLOSE#-1:F
ORI=0TO10:NEXT:CSAVEM"C",&H3B00,
&H3DFF,0:GOTO510
360 E$="":CLS5:FORI=1TO32:E$=E$+
MID$(F$,151-RND(75),1):E$=E$+MID
$(F$,76-RND(75),1):NEXT:PRINT@0,
E$;:POKE1031,175:PRINT@145,LEFT$(
E$,7)+CHR$(191);:FORI=97TO449ST
EP32:PRINT@I,STRING$(15,43);:NEX
T:RETURN
370 CLS:PRINT@5,"THE FOLLOWING I
S THE SET OFWORDS THAT YOU FORME
D THIS TURN.IF ALL ARE NOT VALID
DICTIONARYWORDS, PRESS clear.
OTHERWISE,PRESS enter TO CONTI
NUE PLAY.":PRINTSTRING$(32,207):
RETURN
380 X$=INKEY$:IFX$=""THEN380ELSE
RETURN
390 GOSUB480:PMODE4:PCLS1:SCREEN
1,1:G=5:M(0)=125:N(0)=100
400 P=.0174532925:H=G*SIN(P*72):
J=COS(P*36):K=SIN(P*36):L=G*COS(
P*72):M(1)=M(0)+G:N(1)=N(0):M(2)
=M(1)+L:N(2)=N(0)-H:M(3)=M(2)+L:
N(3)=N(0):M(4)=M(3)+G:N(4)=N(0):
M(5)=M(4)-H*J:N(5)=N(4)+H*K:M(6)
=M(5)+L:N(6)=N(5)+H
410 M(7)=M(6)-G*J:N(7)=N(6)-G*K:
M(8)=M(7)-H*J:N(8)=N(6):M(9)=M(8
)+G-H*J:N(9)=N(5):G=G+5:GOSUB430
:M(0)=M(0)-7:N(0)=N(0)-2:IFG<V
THEN400
420 FORT=1TOV STEP6:GOSUB440:NEX
T:CLS:RETURN
430 FORI=0TO8:LINE(M(I),N(I))-M
(I+1),N(I+1)),PRESET:NEXT:LINE-(
M(0),N(0)),PRESET:RETURN
440 PMODE3:SCREEN1,0:FORI=1TO150
:NEXT:SCREEN1,1:FORI=1TO150:NEXT
:RETURN
450 CLS:PRINT@41,"ORDER OF PLAY"
:PRINT@98,"1. USE ARROWS TO POSI
TION.":PRINT" 2. TO PUT OPTIONA
L LETTER IN RED CHOICE POSITIO
N, PRESS SPACE AND THEN LET
TER KEY.":PRINT" 3. TYPE 'A' FO
R across OR 'D' FOR down."

```

continued on Page 59

# Nouveau Rock 'N' Roll

By Bill Bernico

**S**urely everyone at one time or another has fooled around with the SOUND and PLAY commands. Maybe you've even gone so far as to compose a tune or two. OK, now what do you do with those tunes? You need a method of presenting them. Some way other than just as a musical program alone. If you run a program that simply plays a song, chances are after one or two runs, you've heard all you care to.

Suppose you have five or 10 or 20 songs. How do you get folks to listen? A menu-driven selection type program

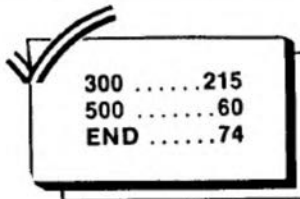
is one way. (BORING!) You could present five or 10 or 20 song programs individually. (BORING!)

How about putting your songs into a program like *RADIO*? Displayed are a radio and the needed instructions. The radio has a digital readout dial and pressing the up- or down-arrow keys helps you "tune in your favorite station."

Once tuned in, press ENTER and the program randomly plays any one of a number of your favorite tunes. Line 5000 states ZQ=RND(5):ON ZQ

GOSUB 6000,7000,8000,9000,10000. That leaves room for five of your own tunes. If you have more or less than five, change Line 5000 accordingly. Also, my lines 6000-10000 are there only as examples of how the RANDOM option selects a tune. Delete lines 6000-10000 and use them to store your own music.

The lines where each of your own selections start should be the same as the lines mentioned in Line 5000. Don't forget to include a RETURN statement after your music. □



300 .....215  
500 .....60  
END .....74

## The listing: RADIO

```

10 'RADIO
20 'BY BILL BERNICO
30 '708 MICHIGAN AVE.
40 'SHEBOYGAN, WI 53081
50 '(414) 459-7350
60 '
70 CLEAR 500
80 SP$="BR3
90 A$="BR3U5ER2FD2NL4D3
100 C$="BR3BR4BU5HL2GD4FR2EBD
110 D$="BR3RU6NLR2FD4GNL2BR
120 E$="BR3U6NR4D3NR3D3R4
130 H$="BR3U3NU3R4NU3D3
140 I$="BR3R2U6NL2NR2D6R2
150 M$="BR3U6F2DUE2D6
160 N$="BR3U6F4NU4D2
170 O$="BR3BRHU4ER2FD4GNL2BR
180 P$="BR3U6R3FDGL3D3BR4
190 R$="BR3U6R3FDGL3RF3
200 S$="BR3BUFR2EUHL2HUER2FBD5
210 T$="BR3BU6R4L2D6BR2
220 U$="BR3BUNU5FR2ENU5BD
230 W$="BR3NU6E2UDF2NU6
240 FOR G=0 TO 9
250 READ GX$(G)
260 NEXT G
270 DATA"BR3BRHU4ERFD4GNLBR2
280 DATA"BR3R2U6NGD6R2
290 DATA"BR3BU5ER2FDGL2GD2R4

```

```

300 DATA"BR3BU5ER2FDGNLFDGL2NHBR
310 DATA"BR3BR3U6G3R4BD3
320 DATA"BR3BUFR2EU2HL3U2R4BD6
330 DATA"BR3BU3R3FDGL2HU4ER2BD6B
R
340 DATA"BR3BU6R4DG3D2BR3
350 DATA"BR3BRHUER2EUHL2GDFR2FDG
NL2BR
360 DATA"BR3BRR2EU4HL2GDFR3BD3
370 PMODE 4,1:PCLS 1:SCREEN 1,1:
COLOR 0,1
380 DRAW"BM150,32R59U16L59D16BL2
0U25D130R100U130L100
390 FOR X=145 TO 225 STEP 23:CIR
CLE(X,45),8:PAINT(X,45),0,0:NEXT
X:CIRCLE(180,100),33
400 CIRCLE(180,100),13:PAINT(180
,100),0,0
410 DRAW"BM130,59
420 FOR X=1 TO 25:DRAW"R100D3L10
0":NEXT X
430 DRAW"BM130,59
440 FOR X=1 TO 33:DRAW"D78R3U78"
:NEXT X
450 DRAW"BM0,180S12"+R$+A$+D$+I$
+O$
460 DRAW"BM130,137S4L20HLHLHLHL5
HLHLHUHUHUHUEUEURURURURFRFRFR
RFDFFDFDGDGDGDGLGLGLGLGLGL4HL
2HL3HLHLHUHUHUHUHUHLHLHL3HL3
HL3GLGLGLGLGLGLDGDGDGDGD2GD2GD2
GD5FFDFD12
470 PAINT(35,165),0,0:DRAW"BM31,
170D10RU9R9DL9DR9D7LU7
480 DRAW"BM10,20S4"+T$+O$+SP$+T$
+U$+N$+E$+SP$+R$+A$+D$+I$+O$:DRA
W"BM8,30"+U$+S$+E$+SP$+T$+H$+E$+
SP$+U$+P$+SP$+A$+N$+D$:DRAW"BM12
,40"+D$+O$+W$+N$+SP$+A$+R$+R$+O$
+W$+S$
490 DRAW"BM10,80"+H$+I$+T$+SP$+E
$+N$+T$+E$+R$+SP$+T$+O$:DRAW"BM1
4,90"+H$+E$+A$+R$+SP$+M$+U$+S$+I
$+C$
500 SC=530
510 GOSUB 590
520 II$=INKEY$:IF II$=""THEN 520
530 IF II$=CHR$(94)THEN SC=SC+10
540 IF SC>1610 THEN SC=530
550 IF II$=CHR$(10)THEN SC=SC-10
560 IF SC<530 THEN SC=1610
570 IF II$=CHR$(13)THEN GOSUB 50
00
580 GOTO 510
590 GG$=""
600 SS$=STR$(SC)
610 LS=LEN(SS$):SS$=RIGHT$(SS$,L
S-1)
620 L=LEN(SS$)
630 FOR A=1 TO L
640 NN$=MID$(SS$,A,1)
650 V=VAL(NN$)
660 GG$=GG$+GX$(V)
670 NEXTA
680 DRAW"BM150,30C1S8"+OG$
690 EXEC 43345
700 OG$=GG$
710 DRAW"BM150,30C0S8"+GG$
720 RETURN
5000 ZQ=RND(5):ON ZQ GOSUB 6000,
7000,8000,9000:RETURN
6000 PLAY"T101CDEFGABAGFEDC":RE
TURN
7000 PLAY"T102CDEFGABAGFEDC":RE
TURN
8000 PLAY"T103CDEFGABAGFEDC":RE
TURN
9000 PLAY"T104CDEFGABAGFEDC":RE
TURN
10000 PLAY"T105CDEFGABAGFEDC":R
ETURN

```



# The Wishing Well Title Maker

By Fred B. Scerbo

Some of you may have noticed that during the last 12 months the opening credits of most of my "Wishing Well" programs have taken on a slightly different look. While in the past I have used the Hi-Res graphics screens to make title cards, I haven't taken a liking to using the CHR\$ colors in the text mode to create introductory title screens. Since many of you have written and requested a way to create text graphics of this style on your own, this month's "Wishing Well" will be dedicated to fulfilling this task.

## The Motivation

Over a year ago, the folks at RAINBOW asked me to create a new RAINBOW ON TAPE title card. I had made the original logo back when the tape format first came out and we were looking for something a little more classy. Since at the time I had just completed a "Wishing Well" series on creating additional colors in PMODE4, I decided to use those colors for the actual graphics.

As those of you who have followed this column will recall, creating these extra colors takes a little time since the pixel patterns must be set and placed in an array. Usually, I would tie up the text screen with CLS0 so the user would not see the colors being created.

When it came time to make the RAINBOW ON TAPE logo, I felt that the user should not have to stare at a black screen for what might seem like an eternity while the graphics being created on the Hi-Res screen was kept hidden from view. Therefore, I chose to have the text screen display the words "Falsoft Inc. presents" in large block letters while this graphics manipulation took place out of sight.

For this, I used a character set that I created for my math program, *Multi-Math Driller*. However, when I included the text generator that created these large, multicolored block letters,

the RAINBOW ON TAPE menu program was too long. To solve this, I used the generator to create my characters and then went through the slow process of examining the text screen memory locations to determine which CHR\$ codes made up the graphics I had just created. The job took a little longer than I wanted, but the result was satisfying. In fact, it brought to mind one of the rules of programming I learned years ago: the greater the time spent by the programmer, the less the time spent by the user. (The reverse of this is also true.)

As more programs for "The Well" required title cards, I started using this technique more often. Each time, however, I streamlined the process so it would take less time. By the time I got to creating *Tri-Planetary Hangmenoids*, the character set no longer was suitable since I needed smaller, more compact letters for longer words.

This led to my writing a short routine to allow me to use the arrow keys for a simpler drawer-type program on the text screen. However, each time I created a screen, I had to go about analyzing the memory locations one line at a time. It worked, but it was slow.

## The Wish

Meanwhile, many "Wishing Well" readers have kept a close watch on my title cards and asked over and over, "When will you give us a program to make title cards like these?" Some readers even wanted to use this technique for creating a string of titles to use for TV displays.

Since all of my efforts had been fractional up to this point, I finally decided to put all the pieces together in a usable program that would not only create the text and analyze the screen memory locations, but write the final BASIC program itself! To make the

program as flexible as possible, the program would have to use both a combination of arrow keys and a direct input format that would change text to large letter characters. The end result of these wishes is *The Wishing Well Title Maker*, which you will find listed here.

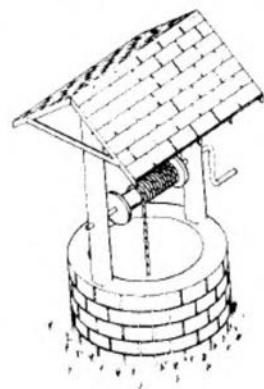
## The Program

Because of a few commands used by the program, I have written it to work with 16K Color Extended BASIC, and not just Color BASIC. The program it produces will work in Color BASIC, however, since the program consists only of DATA statements of the CHR\$ codes needed to create your screen.

The way the program works is actually quite simple. Let's say you have created a screen you wish to use. (I'll explain all the steps later.) You have two options. You may save the screen's memory locations in machine language to either disk or tape. This allows you to save a screen you have been working on and reload it later for other alterations.

Your other option, if you are satisfied with your results, is to use the "analyze" option, which will create the complete BASIC program to redraw your screen independently of the original program. The way the analyze option works is to start at the bottom line of the screen available and check all its locations to see if they are filled with black blocks of CHR\$(128). If the row is completely black, the program keeps moving up a row until it finds characters. This prevents us from having a really long program of DATA statements if only a few lines are needed.

Once the bottom row of characters is found, the program returns to the top row and analyzes each block to determine the CHR\$ code by peeking at the screen location in memory. Whatever value is found in that location, the



number will have 128 subtracted from it. Later, when the number is used from DATA, 128 is added to it. This saves a byte here and there by keeping our DATA numbers as low as possible. Also, in cases where the CHR\$ is 128 (a black box), the value will end up as zero for our DATA line.

Rather than eat up memory by having a zero in a DATA line, the program will leave nothing between the commas. The new program will thus later read a zero and add 128 to it, giving us our CHR\$(128). This also explains why you will often see my DATA lines with strings of commas. These are zero values and I am simply trying to save some space. Look at the beginning of this listing and you will see what I mean.

In the analyze mode, the program writes a file in ASCII to either disk or tape. This ASCII file can later be loaded directly into memory as a BASIC program that you can add to later. You may even merge this file if you have a disk drive using the MERGE command. Otherwise, create a screen and then add onto your program from there.

### Using the Program

Type in the program exactly as it is listed, making sure to leave out no lines or commas in the DATA statements. When the title screen comes up, you may press ENTER to proceed with the program. A new screen then comes up asking:

LOAD AN OLD FILE (Y/N) ?

You may press 'Y' if you wish to continue working on an old screen. If you do, you are asked to enter a filename with:

ENTER FILENAME:

which must be less than nine characters long. The program next asks:

FROM (D)ISK OR (T)APE ?

At this point, you should either have your disk in the disk drive or your tape in the cassette player with Play pushed. Failure to do this might cause an I/O Error, especially with disk.

If you have not chosen to load an old file, the screen will next say:

YOU MUST SELECT A FILENAME.

ENTER NEW FILENAME:

This filename is used later when you wish to save or analyze the screen you create. The next question to answer is:

(B)LANK SCREEN OR (A)UTOPRINT?

Autoprint allows you to enter up to four lines of text at a selected color. The internal character set in the program is used to create your screen. This method is not flawless, however, since your text cannot be over eight characters long. If the characters include M, N, W or X,

then you may have to use fewer than eight characters.

You may only choose to enter one or two lines of text. When the screen says:

ENTER TEXT:

enter the words or spaces you wish to use. To center a short word such as "hello," you may wish to insert one or two spaces before the word. A space only takes up half the space of a character. You will have to experiment to get the effect desired. If you enter no text, the program will proceed to create your screen. Otherwise, you are asked to enter four lines of text.

Next you need to select the color of the characters or text to be printed. You must select the color with the numbers 1 to 8. Use this guide:

- 1) Green
- 2) Yellow
- 3) Blue
- 4) Red
- 5) Buff
- 6) Aqua
- 7) Magenta
- 8) Orange

You may not use a zero or a number larger than eight. Use the numbers listed above to select the way you want your text to look. After the text has been created, you will be in the screen editor mode.

### Using the Screen Editor

If you select "blank screen" as your option, you will be in the screen editor mode. In this mode, you may use the arrow keys to draw or create your own characters. Here is a summary of the functions included:

Arrow keys — one space at a time

SHIFT-arrow — moves to that corner

1 to 8 — colors as listed earlier

9 — return to black cursor

SHIFT @ — clear the screen

\* — save screen in machine language

# — analyze and create BASIC file

When you choose to analyze the screen or to save the screen in machine language, you are again asked:

PREPARE (D)ISK OR (T)APE

At this point, prepare your disk or tape and press either 'D' or 'T'.

If you have used the autoprint option you will end up in the screen editor mode anyway. You may choose to add borders or other decorations to enhance the screen. Quite often, I will draw thin block letters using only the editor since most of my titles are over eight characters long. In fact, the actual title card for this program was created in this fashion as was the new text title page for RAIN-

BOW ON TAPE 1986.

A few hints are needed here. The bottom line of the screen is not included in your screen design. It contains a summary of your colors and commands so you do not have to refer to this article. Also, the program is designed to use only a black background, CLS@. This is due to the way the CHR\$ blocks are designed with black for the blank areas.

Also, when creating designs, it may take a little getting used to switching back and forth between a color and '9', which is black. With a little practice, you will get the hang of it. Remember, if you place some colors too close to each other, they may overlap. Once again, a little practice will help you avoid this structural limitation of the CoCo screen.

### Other Pointers

I did try to include the keyboard alphanumeric characters in the editor mode but found it caused too much trouble with overall screen control. Therefore, if you must add text to a screen, do it using PRINT@ after the DATA statements in the BASIC program this *Title Maker* creates.

Also, keep in mind that if you load an old screen filename, that filename will be used to write the new file to disk or tape. This does not cause a problem with the ASCII file or machine language file having the same name, but if you are altering an existing machine language file, the new one you save will overwrite the old. You may avoid this simply by using different disks for loading and saving, or by using RENAME independent of the program.

You can string a group of these title cards together using Disk BASIC's MERGE command. You may also do the same with tape by renumbering each file, resaving it in ASCII and loading it into a tape-based word processor. Be sure to use FOR/NEXT delays (i.e., FOR I=1 to 2000:NEXT) to keep your screens from flipping by too quickly. You be the judge of how you want these to work. Your only limitation is the amount of free memory left.

One last point should be included here. If you wish to have the program file written so the DATA statements will actually be the ASCII values of the CHR\$ that form the screen images, change the value of SW in Line 10 to zero (0). SW stands for "short way," which is using empty commas to save memory.

50	136	520	12
100	60	635	90
180	45	750	212
285	199	END	234
365	56		

The listing: TITLEMKR

```

1 REM *****
2 REM * THE WISHING WELL *
3 REM * TITLE MAKER *
4 REM * BY FRED B. SCERBO *
5 REM * 60 HARDING AVE. *
6 REM * NORTH ADAMS, MA 01247 *
7 REM * COPYRIGHT (C) 1985 *
8 REM *****
10 CLS:FORI=1TO384:READA:PRINTC
HR$(A+128);NEXT:SW=1
15 DATA14,14,10,13,,14,4,14,12,1
0,,20,26,,30,20,30,21,28,29,20,2
6,21,24,29,24,29,16,30,21,28,29
20 DATA,10,,5,3,10,,11,2,,26,2
6,26,,26,21,19,19,16,27,23,,21,,
21,25,26,21,17,19
25 DATA,10,,5,10,,10,,2,,26,26
,26,,26,17,16,21,,26,21,,21,,21,
,26,21,16,21
30 DATA4,12,,12,,12,4,12,12,8,,,
28,28,24,20,28,20,28,28,20,24,20
,24,28,24,28,20,28,20,28,28
35 DATA109,,101,104,109,108,106,
109,104,96,109,104,16,,126,125,
124,122,116,126,,126,125,124,122
,125,120,,125,124,125
40 DATA101,101,101,,101,99,,101,
,,101,96,,117,,122,,117,,
117,,117,115,114
45 DATA101,101,101,,101,98,101,
96,106,101,106,,117,,122,,
117,,117,,122,117,113
50 DATA100,108,108,,108,108,104,
108,108,104,108,108,104,,112,116
,124,124,,116,124,80,116,124,124
,112,124,124,124,124,124,124
55 DATA60,60,60,60,60,60,85,92,9
4,93,85,92,93,,93,,86,88,84,94,9
2,90,93,92,92,90,60,60,60,60,60
60
60 DATA51,51,51,51,51,51,85,80,9
0,85,85,83,87,,85,86,80,,91,82,
85,83,83,90,51,51,51,51,51,51
65 DATA48,,,,,85,,85,85,,85,,8
5,84,82,80,,90,,82,85,,89,,,,,
48
70 DATA60,60,60,60,60,60,84,48,,
84,84,,84,80,92,80,84,88,84,92,9
2,88,92,88,80,88,60,60,60,60,60
60
75 PRINT@422," BY FRED B. SCERBO
";
80 PRINT@454," COPYRIGHT (C) 198
5 ";
85 IFINKEY$<>CHR$(13)THEN85
90 CLS:PRINT@132,"LOAD AN OLD FI
LE (Y/N) ?"
95 X$=INKEY$:IFX$="Y"THEN100ELSE
IFX$="N"THEN125ELSE95
100 PRINT:PRINTTAB(4);"ENTER FIL
E NAME: ";:LINEINPUT G$:IF LEN(G
$)>8 THEN90
105 PRINT:PRINTTAB(4)"FROM (D) IS
K OR (T)APE ?";
110 X$=INKEY$:IFX$="T"THEN115ELS
EIFX$="D"THEN120ELSE110
115 CLOADM G$:F$=G$:GOTO530
120 LOADM G$:F$=G$:GOTO530
125 CLS:PRINT@130,"YOU MUST SELE
CT A FILE NAME."
130 PRINT:PRINT" ENTER NEW FILE
NAME: ";:LINEINPUT F$:IF LEN(F$
)>8 THEN 130
135 PRINT:PRINT" (B) LANK SCREEN
OR (A) UTO PRINT?"
140 X$=INKEY$:IFX$="B"THEN525ELS
EIFX$="A"THEN145ELSE140
145 CLS:PRINT@231," PLEASE STAND
BY ";
150 DIM A(45,9),B(4,12),K(8)
155 FORI=1TO8:K(I)=D:D=D+16:NEXT
I
160 FORI=2TO11:FORY=1TO9:READ A$

```

```

:A(I,Y)=ASC(A$)+63:NEXTY,I
165 FORI=19TO44:FORY=1TO9
170 READ A$:IFA$=""THEN A(I,Y)=0
:GOTO180
175 A(I,Y)=ASC(A$)+63
180 NEXTY,I
185 FORI=1TO4:FORY=1TO12:READ A$
:B(I,Y)=ASC(A$)+63:NEXTY,I
190 CLS:PRINT"FIRST TEXT LINE (8
CHARACTERS)."
195 FORY=1TO4
200 PRINT"ENTER TEXT: ";:LINEINP
UT W$(Y):IF W$(Y)=""THEN215
205 PRINT"ENTER COLOR #: ";:INPU
TQ(Y)
210 PRINT:NEXTY
215 K=32:CLS0
220 FORI=1TO Y-1:L=K:W$=W$(I):
C=K(Q(I)):GOSUB230:K=K+96:NEXTI
I
225 GOTO530
230 P=LEN(W$):FORZ=1TOP:I=ASC(MI
D$(W$,Z,1))-46
235 IFI=31THEN245ELSEIFI=32THEN2
50ELSEIFI=41THEN255ELSEIFI=42THE
N260ELSEIFI=-14THEN265
240 GOSUB275:GOTO270
245 I=1:GOSUB300:GOTO270
250 I=2:GOSUB300:GOTO270
255 I=3:GOSUB300:GOTO270
260 I=4:GOSUB300:GOTO270
265 L=L+2
270 NEXT:RETURN
275 PRINT@0+L,CHR$(A(I,1)+C)CHR$
(A(I,2)+C)CHR$(A(I,3)+C);
280 PRINT@32+L,CHR$(A(I,4)+C)CHR
$(A(I,5)+C)CHR$(A(I,6)+C);
285 PRINT@64+L,CHR$(A(I,7)+C)CHR
$(A(I,8)+C)CHR$(A(I,9)+C);
290 L=L+4:RETURN
295 GOTO295
300 PRINT@0+L,CHR$(B(I,1)+C)CHR$
(B(I,2)+C)CHR$(B(I,3)+C)CHR$(B(I
,4)+C);
305 PRINT@32+L,CHR$(B(I,5)+C)CHR
$(B(I,6)+C)CHR$(B(I,7)+C)CHR$(B(
I,8)+C);
310 PRINT@64+L,CHR$(B(I,9)+C)CHR
$(B(I,10)+C)CHR$(B(I,11)+C)CHR$(
B(I,12)+C);:L=L+5:RETURN
315 PRINT@Q,CHR$(154);:PRINT@Q+3
0,CHR$(145)CHR$(128)CHR$(154)CHR
$(145);:PRINT@Q+63,CHR$(153)CHR$
(155)CHR$(152);:PRINT@Q+96,CHR$(
152);:RETURN
320 FORI=1TO1500
325 IFPEEK(339)=254THEN330ELSEIF
INKEY$=""THENNEXT
330 RETURN
335 DATAH,M,L,P,A,P,E,M,I
340 DATAB,P,A,A,P,A,E,M,I
345 DATAO,M,L,D,M,B,M,M,M,I
350 DATAM,M,L,M,M,P,M,M,I
355 DATAP,F,K,M,N,O,A,E,I
360 DATAP,M,M,M,M,P,M,M,M,I
365 DATAP,M,M,M,P,M,M,M,M,I
370 DATAO,M,P,A,H,I,E,I,A
375 DATAP,M,P,P,M,P,M,M,M,I
380 DATAP,M,P,M,P,M,P,M,M,M,I
385 DATAH,M,L,P,M,P,M,A,M
390 DATAP,M,L,P,M,L,M,M,I
395 DATAP,M,M,P,A,A,M,M,M,I
400 DATAP,M,L,P,A,P,M,M,M,I
405 DATAP,M,M,P,M,M,M,M,M,I
410 DATAP,M,M,P,M,M,M,M,A,A
415 DATAP,M,M,P,E,P,M,M,M,M,I
420 DATAP,A,P,P,M,P,M,A,M,I
425 DATAE,P,I,A,P,A,E,M,I
430 DATAM,N,O,A,P,F,K,M,M,I
435 DATAP,B,O,P,N,C,M,A,M
440 DATAP,A,A,P,A,A,M,M,M,I
445 DATA,,,,,
450 DATAP,M,P,P,A,P,M,M,M,I
455 DATAP,M,P,P,M,M,M,A,A
460 DATAP,M,P,P,B,P,M,M,O
465 DATAP,M,P,P,N,C,M,A,M
470 DATAP,M,M,M,M,P,M,M,M,I
475 DATAM,P,M,A,P,A,M,A,M,I
480 DATAP,A,P,P,A,P,M,M,M,I

```

```

485 DATA,A,H,N,D,O,A,M,A
490 DATA,,,,,
495 DATA,A,H,E,P,I,A,M,A
500 DATAM,P,D,M,A,M,M,M,I
505 DATAP,C,B,P,E,I,P,M,A,A,M
510 DATAP,L,A,P,P,E,L,P,M,A,E,M
515 DATAP,A,A,P,P,G,J,P,E,I,E,I
520 DATAN,C,B,O,B,G,J,C,M,A,A,M
525 CLS0
530 R$=CHR$(128):PRINT@481,"":F
ORI=143TO255STEP16:PRINTCHR$(I)R
$;:NEXT
535 Q=48:FORI=1504TO1520STEP2:Q=
Q+1:POKEI,Q:NEXTI:POKEI,42:PRINT
@497,R$;
540 PRINT@499,"save"R$R$"analys"
;:POKE1528,35:POKE1535,5
545 A$="PAGE"
550 C=0:H=0:V=0
555 X$=INKEY$:IFX$=""THEN555
560 IFX$=CHR$(8)THENH=H-1
565 IFX$=CHR$(9)THENH=H+1
570 IFX$=CHR$(10)THENV=V+1
575 IFX$=CHR$(94)THENV=V-1
580 IFX$=CHR$(95)THENV=0
585 IFX$=CHR$(91)THENV=29
590 IFX$=CHR$(21)THENH=0
595 IFX$=CHR$(93)THENH=63
600 IFX$="*"THEN655
605 IFX$="#"THEN675
610 IFX$=CHR$(19)THEN525
615 IFH<0THENH=0
620 IFV<0THENV=0
625 IFH>63THENH=63
630 IFV>29THENV=29
635 X=VAL(X$):IFX<1 OR X>9THEN64
5
640 C=X
645 IFC=9 THEN RESET(H,V):GOTO55
5
650 SET(H,V,C):GOTO555
655 GOSUB680
660 X$=INKEY$:IFX$="D"THEN670ELS
EIFX$="T"THEN665ELSE660
665 CSAVEM F$,1024,1503,0:GOTO53
0
670 SAVEM F$,1024,1503,0:GOTO530
675 GOSUB680:GOTO700
680 PRINT@480,STRING$(31,32);
685 POKE1535,143
690 PRINT@484,"PREPARE (D) ISK OR
(T)APE ";
695 RETURN
700 X$=INKEY$:IFX$="D"THEN 705EL
SEIFX$="T"THEN710ELSE700
705 DV=1:F$=F$+"/BAS":GOTO715
710 DV=-1
715 PRINT@480," NOW ANALYSING SC
REEN LOCATIONS";
720 FOR L=1472TO1056STEP-32:ST=0
725 FORM=L TO L+31:RS=PEEK(M):ST
=ST+RS:NEXTM
730 IF ST=4096THEN NEXTL
735 REM OPEN FILE
740 OPEN"O",#DV,F$
745 PRINT#DV,"10 CLS:FORI=1TO";
L-1024+32;:READ A:;
750 IF SW=1 THEN 760
755 PRINT#DV,"PRINTCHR$(A);:NEXT
":GOTO765
760 PRINT#DV,"PRINTCHR$(A+128);:
NEXT"
765 LN=10:FOR N=1024TO L STEP32
770 LN=LN+10:W$=STR$(LN):QW=LEN(
W$):W$=RIGHT$(W$,QW-1)+" DATA":P
RINT#DV,W$;
775 FOR M=N TO N+31:RS=PEEK(M)
780 IF SW=0THEN790
785 RS=RS-128
790 RS$=STR$(RS):QW=LEN(RS$):RS$
=RIGHT$(RS$,QW-1):IF RS$="0"THEN
RS$=""
795 PRINT#DV,RS$;:IF M<>N+31 THE
N PRINT#DV,"";
800 NEXTM:PRINT#DV,"":NEXTN
805 PRINT#DV,"1000 GOTO1000"
810 PRINT@480,STRING$(31,32);
815 CLSE#DV
820 GOTO530

```

*Are you selling your home and need to find out a fair asking price? With a few simple questions answered, it's easy to figure*

# Assessing the Market Value of Your Home

By Harry W. Hallstrom

**B**ack a couple of years ago I was interested in selling my home, but not quite sure what I should be asking as a fair price. The first thing I did, as I'm sure just about everyone does, was get out the phone book. After a careful selection of three real estate agents, I gave each one a call. When all were met and given the grand tour each asked me, "What are you asking as a selling price range?" Now wait a minute, I thought, they are supposed to tell me what a good asking price is. In time they all did, but they were all different. The price spread was almost \$20,000 on a house that is about \$120,000 in value. That's when I decided there must be a better way to accurately determine the true market value of a home.

After thinking about it for several weeks, I realized that all real estate is assessed by the city or town to which you pay taxes. Most cities/towns also reassess properties every so often — 10 years seems to be the average time span between assessments. With that information, I wrote *House Value* to help determine the current market value of a piece of property. Three pieces of information are required to run the program: What year did you purchase your home or when was it last assessed, at how much was it assessed and what is the percentage of assessment?

Most homeowners know the answers to these three questions. If you do not know, there are several ways to find out. The first is to call the city hall and ask the tax collector. Another way is to call the bank holding the mortgage on your home and ask if they can help you with those questions. If you get an itemized tax bill, usually the assessed value and percentage of assessment is furnished on the bill.

*House Value* takes the assessed value

and determines the actual value based on the percentage of assessment. Once that is determined, the national inflation rate is added to each year from the year of purchase or last assessment to the present year.

## How the Program Works

Type in the listing and save a copy to disk or tape before running. You are greeted with a title screen and the first data entry point, "year home was bought?" Enter the year your home was purchased or the last year of assessment, whichever is later. For example, if your home was purchased in 1968 but reassessed in 1975, then enter 1975. You must enter a year between 1968 and 1985, and as a four-digit year. Lines 13 through 17 look for this entry and verify if the year is between 1968 and 1985. I used 1968 as the earliest year allowed, figuring most real estate property has at least been reassessed since that time.

The next data entry point is "assessed value at purchase?" Here again, this means at the time of purchase or latest assessment. Enter this figure as a dollar figure (the '\$' sign is not necessary). Keep in mind if you have added something permanent to the property since buying or assessment to add that to the assessed value. Suppose you added a \$5,000 solar system last year; by all means add \$5,000 to the assessed value. Lines 18 through 21 look for this entry.

The last piece of information you have to enter is the "percent of assessment?" Lines 22 through 26 look for this data. Lines 27 through 31 perform the necessary mathematical calculations to determine present value. Once the assessment figure is entered the screen redisplay the data and the true market value based on 100 percent assessment. You are prompted to hit any key to

continue. Lines 27 through 36 perform this function. You are then asked if you want a screen listing of "house value each year" or "present value?" Lines 37 through 43 look for a keyboard entry and verify it is the correct key input.

Lines 44 through 62 are the years from 1968 to 1985. Based on your earlier input the program continues down the lines until it finds the year match. At that point, the new market value is calculated based on the national inflation rate. Note that Line 62 assumes an inflation rate of 4 percent. When the exact figures are published by the federal government you might want to correct this line. In any case, they won't be far from 4 percent. Lines 63 through 67 display the "present market value" with a pause for user input to run the program again or end. Lines 68 through 71 restart the program or end it, depending on your input. Lines 72 through 76 are the subroutine used to cycle through each year if you selected a yearly listing of the increasing market value.

*House Value* can easily be modified to direct the data to a printer. I didn't have any need for that, so it wasn't done. Feel free to use and modify this program any way you wish. If there is enough interest in a printer version I will work something out and perhaps add it as a later article.



19	.....57
41	.....31
32	.....12
63	.....41
END	.....163

**The listing: HOUSEVAL**

```

1 ' PROG NAME 'HOUSEVAL.BAS'
2 '
3 ' HARRY W. HALLSTROM
4 ' MARSH ROAD
5 ' NORTHFIELD, CT. 06778
6 '
7 ' VERSION 1.6
8 '
9 CLS: EY=0
10 PCLEAR1: CLEAR1000
11 CL$=STRING$(254,32)
12 FOR I=1029TO1050: READX: POKEI,
X:NEXT: FOR I=1065TO1078: READX: PO
KEI, X:NEXT: FOR I=1094TO1113: READ
X: POKEI, X:NEXT
13 PRINT@96, STRING$(32, &H3D);
14 PRINT@165, "YEAR HOME WAS BOUG
HT": PRINT@198, "YYYY" "": LINEINP
UT YR$
15 IF YR$ < "1968" OR YR$ > "1985" TH
EN SOUND200, 3: PRINT: PRINTTAB(1)
ENTER A YEAR BETWEEN 1968-1985":
FORT=1TO300: NEXT: PRINT@128, CL$:
GOTO14
16 IF LEN(YR$) < 4 OR LEN(YR$) > 4 T
HENPRINT@128, CL$: GOTO14
17 YR=VAL(YR$)
18 PRINT@160, CL$
19 PRINT@163, "ASSESSED VALUE AT
PURCHASE
20 PRINT@203, "$": LINEINPUT AV$
21 IF LEN(AV$) = 0 THENPRINT@128,
CL$: GOTO19
22 PRINT@160, CL$
23 PRINT@169, "% OF ASSESSMENT
24 PRINT@210, "%": PRINT@207, ": LIN
EINPUT AA$
25 AA=VAL(AA$): IF AA < 0 OR AA > 10

```

```

0 THENPRINT@128, CL$: GOTO23
26 PRINT@160, CL$
27 AV=VAL(AV$): AA=VAL(AA$)
28 IF AA=100 THEN 31
29 AA=100-AA
30 NV=AV*(AA/100): MV=AV+NV: GOTO3
2
31 MV=AV
32 PRINT@165, "HOUSE PURCHASED: "
; YR$
33 PRINT@229, "ASSESSED VALUE": P
RINTUSING" $###, ###": AV
34 PRINT@290, "MARKET VALUE": YR:
PRINTUSING" $###, ###": MV
35 PRINT@394, "HIT ANY KEY
36 EXEC44539
37 PRINT@160, CL$
38 PRINT@166, "LIST VALUE EACH YE
AR
39 PRINT@207, "OR
40 PRINT@231, "LIST PRESENT VALUE
41 PRINT@327, "ENTER e OR p "
: LINEINPUT A$
42 IF A$ = "E" THEN EY=1 ELSE IF A
$ = "P" THEN 43 ELSE 41
43 PRINT@128, CL$
44 IF YR=1968 THEN SV=MV*.042: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=4.2%
45 IF YR=1969 THEN SV=MV*.054: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=5.4%
46 IF YR=1970 THEN SV=MV*.055: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=5.5%
47 IF YR=1971 THEN SV=MV*.034: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=3.4%
48 IF YR=1972 THEN SV=MV*.034: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=3.4%
49 IF YR=1973 THEN SV=MV*.088: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=8.8%
50 IF YR=1974 THEN SV=MV*.122: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=12.2%
51 IF YR=1975 THEN SV=MV*.07: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSUB
72 'INFLATION=7%
52 IF YR=1976 THEN SV=MV*.048: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=4.8%
53 IF YR=1977 THEN SV=MV*.068: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU

```

```

B72 'INFLATION=6.8%
54 IF YR=1978 THEN SV=MV*.09: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSUB
72 'INFLATION=9%
55 IF YR=1979 THEN SV=MV*.133: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=13.3%
56 IF YR=1980 THEN SV=MV*.124: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=12.4%
57 IF YR=1981 THEN SV=MV*.089: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=8.9%
58 IF YR=1982 THEN SV=MV*.039: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=3.9%
59 IF YR=1983 THEN SV=MV*.038: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSU
B72 'INFLATION=3.8%
60 IF YR=1984 THEN SV=MV*.04: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSUB
72 'INFLATION=4.0%
61 PRINT@128, CL$
62 IF YR=1985 THEN SV=MV*.04: MV
=MV+SV: YR=YR+1: IF EY=1 THEN GOSUB
72 'INFLATION=4.0%
63 PRINT@128, CL$
64 PRINT@166, "PRESENT VALUE @": Y
R
65 PRINT@235, USING"$#, ###, ###": M
V
66 PRINT@362, "HIT ANY KEY
67 EXEC44539
68 PRINT@160, CL$
69 PRINT@325, " RUN AGAIN y/n
70 A$=INKEY$: IF A$="" THEN 70
71 IF A$="Y" THEN 9 ELSE IF A$="N"
THENCLS: PRINTTAB(9) "PROGRAM ENDE
D": END ELSE 70
72 IF YR=1986 THEN GOTO 63: PRINT
@128, CL$
73 PRINT@196, "VALUE @": YR: "=": P
RINTUSING" $###, ###": MV
74 PRINT@325, "HIT ANY KEY TO ADV
ANCE
75 EXEC44539
76 RETURN
77 DATA 72, 79, 85, 83, 69, 96, 86, 65,
76, 85, 69, 96, 67, 65, 76, 67, 85, 76, 65
, 84, 79, 82
78 DATA 66, 89, 96, 72, 110, 72, 65, 76
, 76, 83, 84, 82, 79, 77
79 DATA 86, 69, 82, 96, 113, 110, 118,
96, 96, 96, 96, 96, 91, 67, 93, 96, 113, 1
21, 120, 116

```

Hint...

## Disk Directory Printout

If you have a long disk directory and want to see all of it, or if you simply wish to have a hard copy printout of your directory, one simple command allows you to do this easily.

Just POKE 111,254: DIR and the entire disk directory will appear on your printer, even if it is too long to be fully displayed on the screen.

Hint...

## Saving in ASCII

When you save programs, CoCo can perform this function in two ways: by using binary codes or actual letters and numbers (called ASCII and pronounced "as-key").

Although it takes longer, ASCII is sometimes a more accurate way to save a program, especially when you may be transferring programs between systems — say from a disk-based to a cassette-based system.

To save in ASCII, simply add a comma and an 'A' to the end of your SAVE instruction, like this: CSAVE "PROGRAM", A and the ASCII save is done by CoCo.

# 16 BIT!

The COLOR COMPUTER is an extremely powerful 8 bit computer which, because of its value for money, suitability for both games and serious computing, ease of expansion and the upward compatibility of software between successive new models has earned a dedicated following of knowledgeable computer enthusiasts. The latest challenge to the CoCo is the new generation of 16-bit computers arriving in the market place with their promise of faster program execution and more addressable memory. The 68008 CPU makes this technology available to CoCo owners in a cost effective way, by connecting CoCo to a 16-bit CPU through the expansion port.

The choice of which microprocessor to use was not difficult as the Motorola MC68000 CPU is favoured by many companies producing 16-bit microcomputers. The basic design criteria was simplicity, flexibility and cost. The unit also had to be hardware compatible with the COLOR COMPUTER, to enable easy upgrade of your computer to the new generation. Data transfer had to be in direct memory access to the CoCo for speed, and both CPU's should work separately to enable co-processing.

The actual CPU is a 68008, the essential features of which are that it has an 8-bit data path, and an address bus of 20 lines giving 1 Megabyte of non segmented linear addressing capability. The software is fully compatible, in source and code, with the 68000 CPU. The hardware on the PCB is arranged to provide the maximum flexibility for two modes of processing; the CO-PROCESSING mode, and the

68000 mode. The PCB is capable of expansion from 256K to 512K.

Standard unit includes:-

RS MINI-DOS  
OS9 MINI-DOS totally interactive with COCO-OS9  
256K RAM DISK full 256K continuous Ramdisk  
8, 16, 32 bit Processing  
2K or 8K Variable storage  
Centronics Parallel Port  
Built-in Power Supply  
Plugs directly into Rom Pack Port or Multipack  
Price \$650.00

Options:-

KAMELION- Interface Operating System approx. \$80  
LFASTV68- Ass. Lang. library of utilities POA.  
PHL CSC- Cross Assembler' approx. \$90  
The BEST Cross Assembler- CRASMB16.32 POA.  
Spare Parallel Port

Options (FUTURE):-

68000 COCO Compatible BASIC- 150K+ free space  
OS9 full 68K (Disk)  
512K Upgrade Planned  
RS232 Port

Available from

BLAXLAND COMPUTER SERVICES  
76a Murphy Street, BLAXLAND 2774  
Telephone: (047) 39-3903

---

# RAINBOW ON DISK \$ 15

Inspired by a "schematic scoundrel," this program is an aid for drawing flow chart diagrams

# CoCoflow: CoCocad Expanded

By Dennis Page

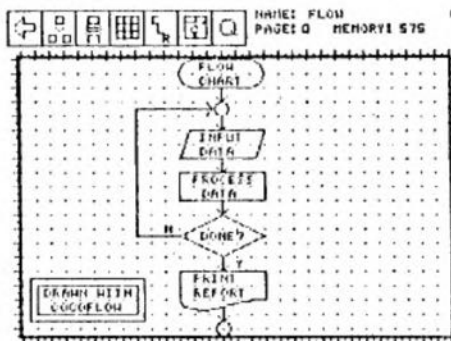
The "mini-CAD" (Computer-Aided Design) program, called *CoCocad*, written by Peter Kerckhoff (RAINBOW, November 1985, Page 61) proved to be very interesting. It aids in drawing schematics. The program is icon menu-driven using a mouse, joystick, X-Pad or touch tablet. The keyboard is seldom used.

I am surrounded by expensive three-dimensional color CAD systems at work and because of that, was skeptical of the *CoCocad* utility. However, after typing in the program to learn how graphics pointing and positioning was handled, new uses for *CoCocad* came to mind. How about modifying *CoCocad* to draw flow charts? A typical computer buff could use the same system to draw schematics for computer circuits and draw corresponding software flow charts as well.

To do this, minor modifications to *CoCocad* are needed. The modified program is called *CoCoflow*, and instead of drawing schematic representations of electronic components, it draws flow chart symbols.

On first look, the only difference between the two programs is that the *CoCocad* diode icon has been replaced by a familiar flow diagram symbol, the diamond-shaped decision block. Figure 1 shows a sample *CoCoflow* screen. The differences become apparent when the flow diagram icon is selected. The pull-down menu shows the first flow chart symbol selection: a terminal symbol that begins or ends the flow chart. As the mouse button is pressed, more symbols show up.

Figure 1: Sample *CoCoflow* Screen



To use a symbol, move the cursor to USE in the menu and press the mouse button. The pull-down menu disappears and the selected symbol may be moved to the desired location. These symbols include right, left, up and down arrows to show flow directions. The arrows may be made longer with the line function, just as *CoCocad* does it.

*CoCocad* modifications are easily accomplished with the following procedure. Copy *CoCocad* to create the new *CoCoflow* program by using the command:

```
COPY "COCOCAD.BAS" TO "COCOFLOW
.BAS"
```

Then load *CoCoflow* by using the command:

```
LOAD "COCOFLOW"
```

Modify *CoCoflow* by typing in Listing 1. The line numbers are arranged to replace the corresponding *CoCocad* lines, so be sure your copy of *CoCocad*

is numbered identically to the listing in the *CoCocad* article. Otherwise, you will have to match up the correct lines for replacement. Line numbers shown alone are line deletions — *CoCoflow* does not need these lines. Be sure to get the line numbers correct, otherwise the wrong lines may be replaced. If you do make a mistake and cannot find the error, just reload *CoCoflow* and start over. When finished, save *CoCoflow* by using the command:

```
SAVE "COCOFLOW"
```

## Description of Modifications

Line 120 changes the dimension of variable arrays CM and CO to reflect the larger flow chart symbols for GET and PUT operations. Line 440 changes the name of the component pull-down menu to symbols and changes the graphics location of the flow chart symbol within the pull-down menu. Line 460 changes the number of selectable symbols and the location of the symbol within the pull-down menu. Lines 470 and 870 also reflect the changes in the number of symbols. Lines 880 through 970 are the DRAW strings for the new symbols. Lines 980 through 1110 are deleted because there are fewer symbols. Line 1760 replaces the diode symbol in the icon menu to the diamond-shaped flow chart symbol.

Now try it! You'll see that *CoCoflow* operates just like *CoCocad*, except that you can now draw flow charts.

*CoCocad* is an excellent example of applying pointing devices, pull-down

menus and icon menus to simplify program operation. Reading Peter's commented program listing reveals much. Rewards await the student who uses *CoCocad* as a learning tool.

As Peter had also suggested in his article, if you use *CoCoflow* to draw a flow chart you hope to have published, please put a little note somewhere denoting that *CoCoflow* was used to

draw the chart. Any questions regarding these modifications may be directed to me at 14108 Doty Avenue #36, Hawthorne, CA 90250. Please include an

**Editor's Note:** The following are modifications to the *CoCocad* program that appeared in "CoCocad: The Schematic Scoundrel" by Peter Kerckhoff.

Load *CoCocad*, type in the following lines and save as *CoCoflow*. *CoCoflow* operates just as *CoCocad* except flow chart symbols appear instead of schematic symbols. For your

The listing: COCOFLOW

```
12Ø DIM C$(3),A(8),AD(8),C1(1),C
2(1),C3(1),L1(6),L2(6),L3(6),L4(
6),CM(32),CO(32),MD(255),MO(255)
:B$="V31L1ØØ04B":NF$="NONE"
```

```
44Ø T$="SYMBOLS":GOSUB85Ø:T$="N
EXT USE":TX=55:TY=115:GOSUB1ØØ
:T$="PREV":TX=55:TY=1Ø7:GOSUB1ØØ
:N=3:GOSUB5Ø:N=1:DRAW"BM56,65":G
OSUB87Ø
```

```
46Ø IF X>8Ø THEN GET(56,65)-(56+
XW,65+YW),CM,G:PUT(5Ø,5Ø)-(11Ø,1
2Ø),MO,PSET:GOTO48Ø ELSE DRAW"BM
56,65C5":GOSUB87Ø:DRAW"BM56,65CØ
":IF Y<1Ø9 THEN 47Ø ELSE N=N+1:IF
N<11 THEN GOSUB87Ø:GOTO45Ø ELSE
N=1:GOSUB87Ø:GOTO45Ø
```

```
47Ø N=N-1:IF N>Ø THEN GOSUB87Ø:G
OTO45Ø ELSE N=1Ø:GOSUB87Ø:GOTO45
Ø
```

```
87Ø ON N GOTO 88Ø,89Ø,9ØØ,91Ø,92
Ø,93Ø,94Ø,95Ø,96Ø,97Ø
```

```
88Ø DRAW"BD7EUEUERERERER34FRFRFD
FD2GDGDGLGLGL34HLHLHUHUHU":XW=48
:YW=16:RETURN:'TERM
```

```
89Ø DRAW"BD16BRU2EU2EU2EU2EUR
42 D2GD2GD2GD2GD2GDL42":XW=48:YW
=16:RETURN:'I/O
```

```
9ØØ DRAW"R48D16L48U16":XW=48:YW=
16:RETURN:'PROCESS
```

```
91Ø DRAW"BD12RERERERERERERERERER
ERERE FRFRFRFRFRFRFRFRFRFRFRFR
GLGLGLGLGLGLGLGLGLGLGL HLHLHLHL
HLHLHLHLHLHLHL":XW=48:YW=25:RET
URN:'DECISION
```

```
92Ø DRAW"R48D16L8GL5GL3GL3GL3GL4
GL6HL4HL2HUHU16":XW=48:YW=22:RET
URN:'DOCUMENT
```

```
93Ø DRAW"BD3EURER2FRDFD2GDLGL2HL
UHU":XW=9:YW=9:RETURN:'CONN
```

```
94Ø DRAW"BD8BR4NU7NH3E3":XW=8:YW
=8:RETURN:'D-ARROW
```

```
95Ø DRAW"BD4BRE3ND7F3":XW=9:YW=8
:RETURN:'U-ARROW
```

```
96Ø DRAW"BD4R7NH3G3":XW=8:YW=8:R
ETURN:'R-ARROW
```

```
97Ø DRAW"BD4BRNR7NE3F3":XW=9:YW=
8:RETURN:'L-ARROW
```

98Ø (delete)

99Ø (delete)

1ØØØ (delete)

1Ø1Ø (delete)

1Ø2Ø (delete)

1Ø3Ø (delete)

1Ø4Ø (delete)

1Ø5Ø (delete)

1Ø6Ø (delete)

1Ø7Ø (delete)

1Ø8Ø (delete)

1Ø9Ø (delete)

11ØØ (delete)

111Ø (delete)

```
176Ø DRAW"BM4,4R6D12R6 BM24,4R12
D12L12U12 BM43,1ØE6F6G6H6 BM62,4
R2BR3R2BR3R2D2BD3D2BD3D2L2BL3L2B
L3L2U2BU3U2BU3U2 BM81,4R12L6D12
BM1Ø2,4D4NR6GFNR6GFNR6GFND4R6ND4
EHEHEHU4L6 BM119,8ND4R6U4F6G6U4L
6":RETURN
```



# WRESTLING ASSEMBLY

by Ian Clarke

NOTE: This article is for newcomers to assembly language - past masters may move right along!

My fellow readers have no doubt seen numerous articles for the assembly language beginner. This is NOT such an article. It is an account of my first brush with assembly language, and my attempts to solve problems without spending money.

It all began with my desire to comply with Graham Morphet's request that programs submitted for publication be accompanied by the appropriate blurb, saved in an ASCII file on the submitted tape. This would normally be done by using a word/text processing program. I don't have one, nor do I intend purchasing one. Without a printer, it would not be worth the money. So, how do I help Graham without dipping into my pocket?

The answer came in a blinding flash! (To be truthful, it came after wading through back copies of magazines. But doesn't the opening sentence sound classy?)

In "Australian Rainbow", March 1985, there was a simple text processing program, and all it would cost me is some typing. This is my meanness again - I don't subscribe to the taped programs which are available.

Did I say that some typing is all that it would cost me? Hollow laughter! The confounded thing is written in assembly language, and I don't have EDTASM+ (or, for that matter, any other assembler).

What is an assembler, do I hear in the background? It is a program which will take the assembly source listings you see in this venerable magazine, and turn them into machine language which can be understood by COCO. Back to the matter in hand... how do I make use of this assembler listing?

I waded through my pile of back issues. Sure enough, there are any number of programs which enable the user to poke the correct values into the memory, and instructions on how to do it. So, I use one. I save the program with a CSAVEN and the correct start / end / exec addresses as shown, and then type "EXEC".

Shock, horror! I am faced with a screen with a three-space orange block, and nothing else. The break key doesn't work. I give the game away in disgust, and settle for submitting my latest program for publication with hand-written notes.

Then I break with my customary thrift, and buy a disk drive from Brian Dougan. Tandy has been out of stock for months, and Brian is a local who comes highly recommended. Smartest thing I've been bullied into for years! (The Briz-Biz mob are to blame for the bullying.) I commence transferring my files from tape to disk, and come across the text processor. Can I make it work?

The first step to take in finding out why it won't

work, is check the values that I've poked into memory are correct. The Dougan-drive has a built-in monitor which enables the user to read bin (machine language) files - something which I believe Tandy do not include. I check my program entry, and find it correct. I stress this, to show that it is not having the Dougan extras which solve the problem. They do, however, help me eliminate the most likely cause i.e. bad data entry. What now?

The second step should be to buy an assembler package, yes? No - I've just re-entered poverty after purchasing a disk drive, remember? The second step is to learn about assembly language. Off to Tandy, and purchase "TRS-80 COLOR COMPUTER ASSEMBLY LANGUAGE PROGRAMMING". I manage to ward off the salesman who insists that I must also buy EDTASM+. I get home, and open the book. Right from the preface, EDTASM+ is pushed. Understandable, as the book is designed to teach assembly language using an assembler. Nevertheless, I commence reading, and find that a smidgeon of understanding seeps through. Time to go back to the text processing program, and have another think.

What is the program supposed to do? The author's comments are comprehensive (I should learn from him, as mine never are). He says the program starts with a menu. So, where is it? Aha! The last half column of his program lists, on the right hand side, a series of instructions called "FCC", each followed by an item such as "6 - CSAVE". The assembly language book says that FCC means "Form Constant Character", and is a pseudo-op which tells the assembler program to allocate the ASCII code for each character in the string which follows. This answers the question which has puzzled me right from the start - why has each FCC instruction got a large block of unused memory after it? It has to be for the ASCII numbers of the strings!

Back I go to POKEing code into memory, after loading the non-working program. Save it again, EXEC it, AND THERE IS THE MENU! AND THE PROGRAM WORKS! Beauty! Graham can now have his ASCII files with any future programs I submit for publication.

- What is the purpose of all this rambling? Well,
1. If you don't want to spend the money, it doesn't necessarily mean you can't solve the problem.
  2. If the solution isn't immediately obvious, keep nibbling at the problem. Occasionally, you'll win.
  3. You learn a lot more about your computer by doing things the hard way, (which frequently means the cheap way).
  4. Assembly language is more interesting than I thought. I must buy an assembler...

## BARDEN'S BUFFER

## Pi to 10,000 Digits

By William Barden, Jr.

Imagine this scene: A friend who owns an Apple IIe or IBM PC walks into your house and makes his usual remark about your "toy CoCo." "What's it doing now," he says with a sneer, "warming up?" Casually flicking a piece of chad off the keyboard, you reply, "No, actually it's calculating pi to 10,000 digits . . ."

This was the scene I envisioned when I first planned this article. Many hours later, I'm a little weary, but still enthusiastic about this month's topic, even though 10,000 digits may be a little optimistic. Imagine using your CoCo's number crunching ability to calculate pi to thousands of decimal places, something that was never really done until the 1950s! A little background might explain the problem.

## Pi through History

Pi, of course, is the ratio of the distance around a circle to the diameter of the circle. As many of you know, pi is approximately 3.1416. A circle with a diameter of 10 inches, for example, has a circumference of about 31.416 inches. It's important to note that pi is an *irrational* number — it's not "even" and goes on to infinity as a never ending fraction. The fraction appears to be made up of random digits. Translating the digits into text characters, for example, would produce every book ever written somewhere within the first billions of billions of digits.

(Speaking of "bilyuns and bilyuns" . . . an interesting note here. Carl Sagan, in his new fictional work, *Contact*, uses the premise that somewhere after the first umpteen digits, pi is *not* random, but has an encoded figure in computer graphics of a circle, presumably put there by the Creator as proof that He exists!)

Believe it or not, the Babylonians had calculated pi to be  $3\frac{1}{8}$  by about 2000 B.C. Even more incredible, the Chinese had arrived at a value of pi of 3.14159 by 264 A.D., an accuracy of 0.0003%!

Early calculations of pi used a method that inscribed thousands of poly-

gons, as shown in Figure 1. The approximation of a circle by polygons gave pi to within 35 decimal places by the 1600s. At that point, however, differential calculus was discovered, and it was found that pi could be expressed as an *infinite series* of terms. The best known of these series for pi was discovered by Leibniz in 1674:

$$\pi = 4(1 - 1/3 + 1/5 - 1/7 + \dots)$$

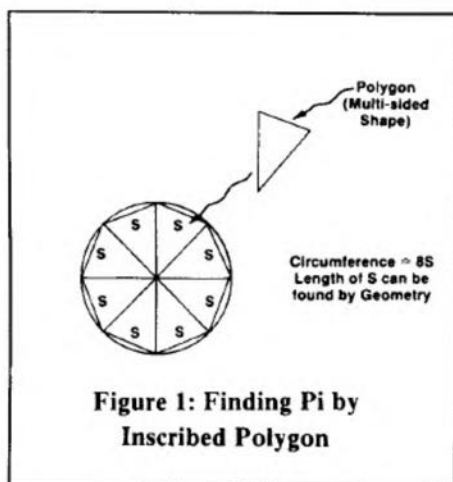


Figure 1: Finding Pi by Inscribed Polygon

Try it with a BASIC program. It works, but requires hundreds of terms to get even several digits of accuracy. From the moment the infinite series method was discovered, a race for calculating pi to the most places ensued, just as it had with polygon approximations. Machin discovered a useful series in 1706, one that *converged* very quickly — only a few terms were required to yield pi to a dozen places or so:

$$\pi = 16(1/5 - 1/(3 \cdot 5^3) + 1/(5 \cdot 5^5 2 - \dots) - 4(1/239 + 1/(3 \cdot 239^3) - (1/(5 \cdot 239^5) + \dots))$$

The Machin series and others have been used in recent times to calculate pi to thousands of places on large mainframe computers, just as they are typically used to search for special prime numbers (Mersenne primes of the form  $2^N - 1$ ). There's really not a great deal

of use for the next hundred thousand places of pi or the next Mersenne prime, but it's fun to do because, like Mt. Everest, pi is *there*. And besides, maybe there *is* that graphical representation of a circle after the billionth digit or so.

(An excellent and fascinating treatment of the history of pi from ancient times to the computer age is contained in the book *A History of Pi* by Peter Beckmann [St. Martin's Press, 1971].)

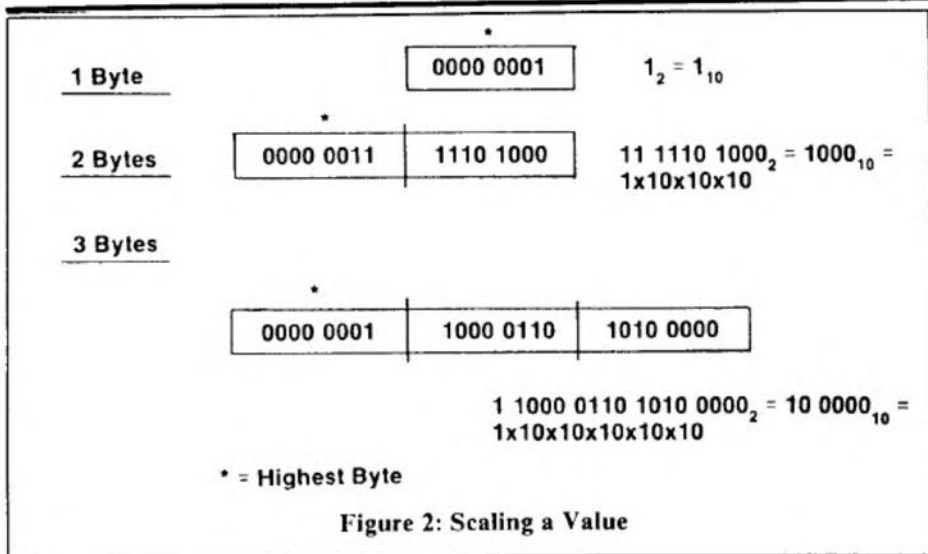
## Because It's There

In this column we'll show you how to compute pi to many places — up to 10,000 decimal digits if you're so inclined (and have a few years)! It won't be an easy task, but it can be a fun project, and it's a perfect squelch to that friend or neighbor who owns an Apple IIe, IBM PC or Cray X-MP. (This column took about three times the nominal time to research and write, but I still had fun.)

## Multiple-Precision Variables

One basic question in computing pi is how 10,000 digits can be represented in a computer system. There are several alternatives. You could store one digit in each byte, or maybe two digits in four bits, two per byte. Each four bits would be *binary coded decimal*, or bcd. However, bcd is a little messy to work with for divides, although it's great for adds and subtracts. You could also use tried and true binary. A little review of binary probably won't hurt at this point.

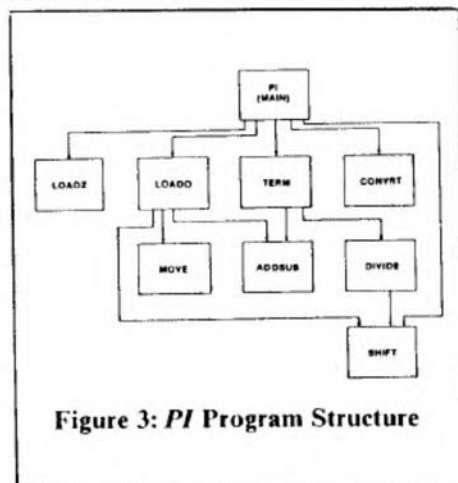
A byte can hold eight bits — values of 00000000 through 11111111 or zero through 255. Two bytes can hold zero through 65,535. Four bytes can hold zero through 4,294,967,296. As a matter of fact, you can make the binary operand as large as you wish. A good rule of thumb is that each 3.5 bits can hold one decimal digit. So 10,000 decimal digits require about 35,000 bits, or about 4,375 bytes. A better calculation of this is 10,000 divided by log base 10 of 2 = 10,000/.301 or 33,219+ bits, 4,152+ bytes.



```

SUM <- 0
TEMP1 <- 1
LAST <- XXXX
ODDEVN <- 1
DIVSOR <- 1
while DIVSOR < LAST do
  begin
    if DIVSOR=1 then TEMP1 <- TEMP1/5 else
      TEMP1 <- TEMP1/25
    TEMP2 <- TEMP1
    TEMP2 <- TEMP2/DIVSOR
    if ODDEVN=1 then SUM <- SUM + TEMP2 else
      SUM <- SUM - TEMP2
    DIVSOR <- DIVSOR + 2
    flip ODDEVN to 1 if 0 or 0 if 1
  end
SUM <- SUM*4
TEMP1 <- 1
LAST <- XXXX
ODDEVN <- 0
DIVSOR <- 1
while DIVSOR < LAST do
  begin
    if DIVSOR=1 then TEMP1 <- TEMP1/239 else
      TEMP1 <- TEMP1/(239*239)
    TEMP2 <- TEMP1
    TEMP2 <- TEMP2/DIVSOR
    if ODDEVN=1 then SUM <- SUM + TEMP2 else
      SUM <- SUM - TEMP2
    DIVSOR <- DIVSOR + 2
    flip ODDEVN to 1 if 0 or 0 if 1
  end
SUM <- SUM*4

```



of 10 followed by 10,000 zeros (for the maximum case), so the term becomes 10,000,000,000,000,000/239. The result of the divide and other operations is always an integer number, and we can stick in the decimal point when we print the answer. I'll show you how it works shortly. The results of all adds, subtracts, multiplies and divides are kept in the MPVs in integer form, without fractions.

### The Algorithm

In a difficult problem such as this, it's best to have a good algorithm before you go charging off into the actual code. There are many ways this calculation can be implemented, but the plan I came up with follows. First, the variables:

- 1) Use three MPVs, called TEMP1, TEMP2 and SUM.
- 2) Use a 16-bit variable called DIVSOR to hold the 1, 3, 5, 7, etc., . . . to be divided into the term.
- 3) Use a 16-bit variable called LAST to hold the last divisor to test for the end of each of the two sets of divisions.
- 4) Use an eight-bit variable called ODDEVN to control whether the term is added to or subtracted from the subtotal.

Now the actual plan, divided into two parts, is computation of the first set of terms and calculation of the second set of terms:

In case you're not familiar with the notation used here, it's a form of "PDL" or Program Design Language. Variables are indicated by uppercase. The arrows indicate that a variable is set to the terms on the right-hand side of the equation. The *begin* and *end* words mark the beginning and ending of a block of operations. The *while* function means that the following block is to be done as long as the condition is true, in this case DIVSOR less than LAST. You can see that the sequence is divided into two parts, one for the 5/25 divides, and one for the 239/(239\*239) divides.

If you don't see how this works, try the algorithm with paper and pencil. Note that variable TEMP1 holds 1/5 the first time through, 1/5^3 the next time, 1/5^5 the next time, and so forth (or 1/239, etc., for the second part). The current DIVSOR is divided into the current power of '5' (or power of 239 term). The result is then added to or subtracted from the SUM.

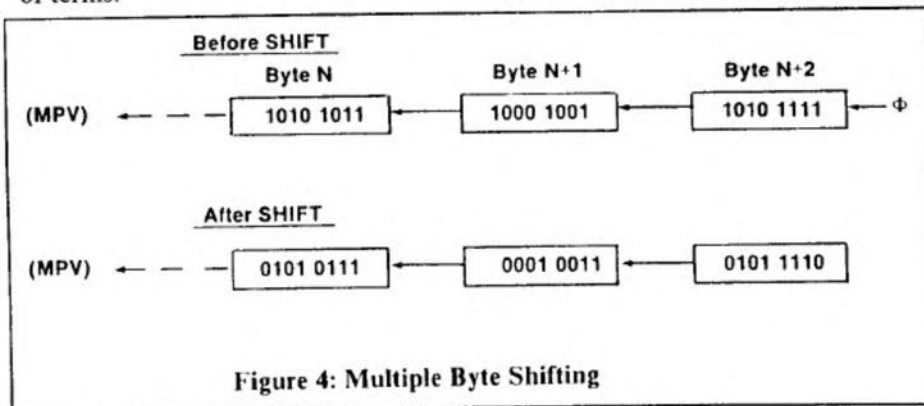
The "LAST" values are not obvious. The values represent the last divisor (preceding the power) to be used. This divisor equals the power of the current term. Each term of the first set is smaller than the preceding term by about 1/25th. Each term of the second set is smaller than the preceding term by

We'll use up to 4,154 bytes, therefore, to hold up to 10,000 decimal digits. For the sake of convenience in the following discussion, we'll call these 4,154-byte variables, MPVs or Multiple-Precision Variables. We'll need a number of these MPVs, some to hold temporary results and some to hold the final results, just as we use several variables in a BASIC program.

### Scaling and Division

The next major problem is the actual calculation — how can we divide 1 by 239, for example? In BASIC it is done by floating-point variables. We can't use floating-point here because we need the *exact* numbers to be represented. Typical BASIC floating-point variables only allow 16 decimal digits of precision — we need thousands of digits of precision!

To accomplish addition, subtraction, multiplication and division of the terms in the calculation, we'll use only integer arithmetic. A question you might be asking, though, is how can we get fractional results with integers? The answer is an old technique, dating back to the '50s or beyond — scaled numbers. Instead of using 1 in the term 1/239, we'll *scale up* the numerator by a factor



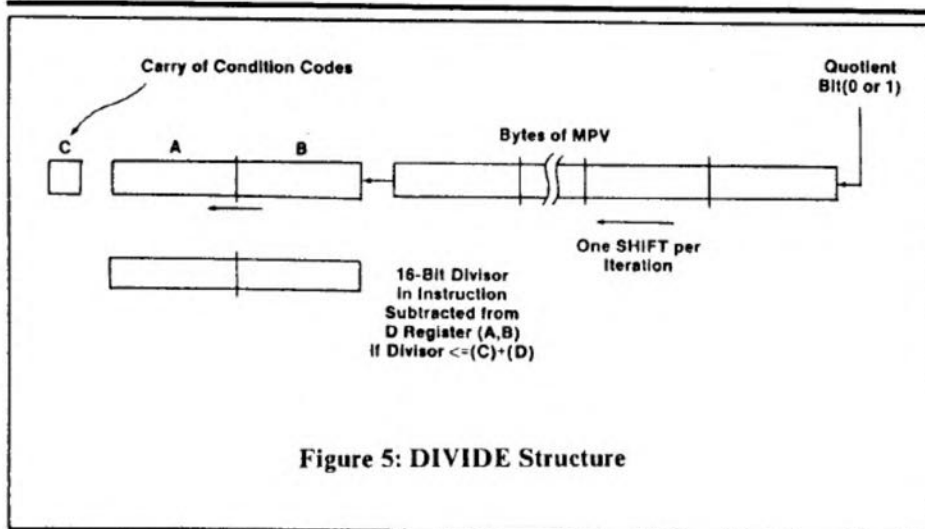


Figure 5: DIVIDE Structure

LOADO first loads a zero into all bytes of the MPV. A scaled one is then loaded into the MPV. This involves loading '1' into the low-order byte of the MPV and then multiplying by 10 as described above.

MOVE moves all the bytes of one MPV into a second MPV, a simple copy operation.

ADDSUB adds the contents of one MPV (the "source" MPV) to the contents of a second MPV (the "destination" MPV), or subtracts the two MPVs. In doing this, any "carry" or "borrow" must be carried over to the next byte, just as in decimal addition or subtraction.

about  $1/(239*239)$ . Divide the number of decimal digits in the MPV by .69897 and add 10 percent of that for the LAST value for the first set. For example, if you're working with 10-byte MPVs, the 80 bits can hold about 22+ decimal digits, so use  $22+/.69897$  or  $33 + 3 = 37$  as the LAST value. (LAST values must be odd.) Divide the number of decimal digits by 2.37 for the LAST value for the second set. In this example,  $22+/2.37$  gives you 9+, or 11 to be safe.

The only thing that's been left out of the algorithm is the scaling. In fact, '1' is not put into the MPV — a power of 10 is, such as 10,000,000,000. This poses a problem. How can you figure out the value to store into the MPV for the power of 10? For short length MPVs, it's easy. If the MPV is one byte, for example, you'd use \$64, equivalent to 100. For two bytes, you'd use \$2,710 for 10,000. When you're working with 10 followed by 10,000 zeros, though, it becomes impossible to calculate the actual value to be put into the MPV, or at least terribly difficult. About the only efficient way to do it is to let the computer do it for you.

One way to store a power of 10 is to put 10 into the bottom of the MPV and then keep multiplying by 10. Because I wanted to be able to use any length MPV from several bytes to 4,154 bytes, I chose this method:

```
MPV <- 1
while highest byte of MPV = zero do
  begin
    MPV <- MPV*10
  end
```

The result of this operation is shown for several lengths of MPVs in Figure 2.

**Planning the Modules**

The structure of the program is shown in Figure 3. Like a lot of programs, this structure didn't just jump from head to paper, but was modified

during the design and coding process. There are 10 modules in the program.

We'll start with the easiest ones first (perhaps you've noticed this flaw in my character by now) SHIFT shifts an MPV one bit to the left. This doesn't sound like much, but remember that all the bytes of the MPV, from a few bytes to 4,154 bytes, must be shifted with the high-order bit going into the low-order bit of the next higher byte, as shown in Figure 4.

**Easy Modules**

LOADZ loads a zero into all bytes of the MPV.

**The Heart of the Program**

DIVIDE is the heart of the program. This subroutine must be as fast as possible to cut down on the program overhead. The structure of DIVIDE is shown in Figure 5. A 16-bit divisor (DIVSOR or 5 or 239) is divided into the MPV. The MPV is shifted into Register D one bit at a time. For each shift, a test is made to see whether the divide will "go." If so, a subtract of the divisor is done and a quotient bit is set to '1'. If not, the quotient bit is set to '0'. Directly after the dividend is shifted into D, the next quotient bit is filled into the low end of the MPV. At the end of

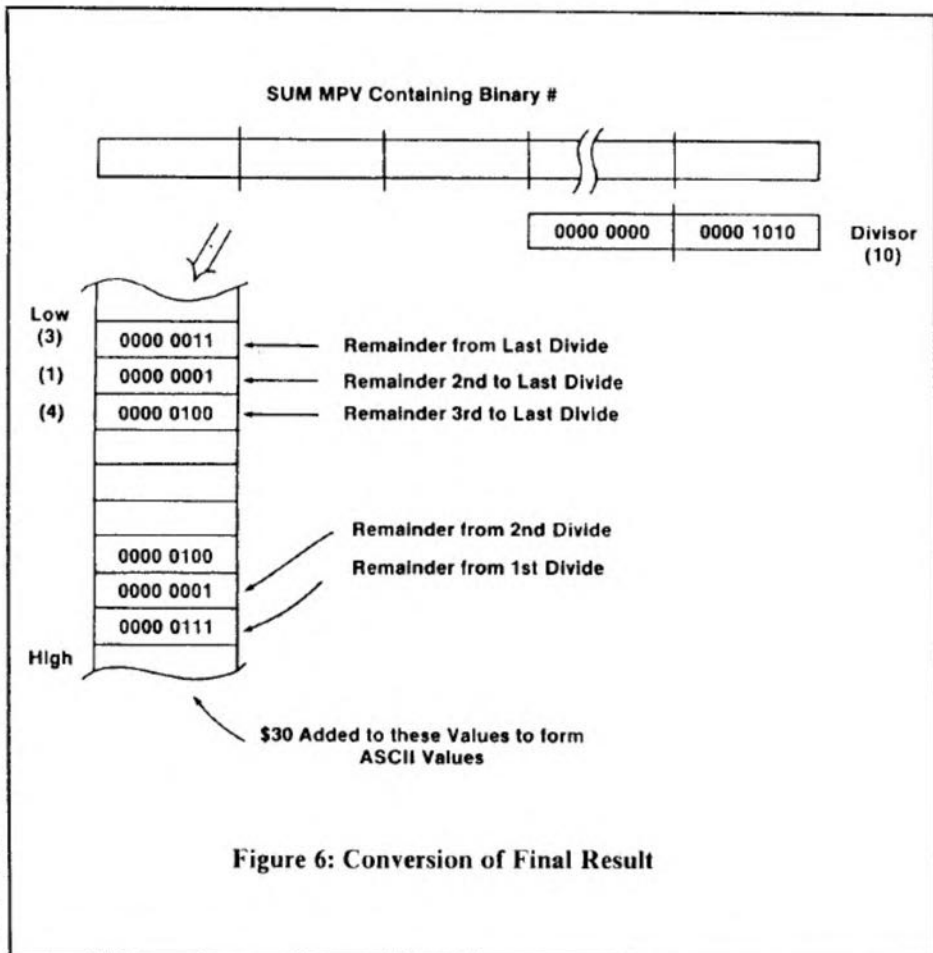


Figure 6: Conversion of Final Result

Listing 1: PINEW

```

00100 *****
00110 * PI TO 10,000 DIGITS ON THE COCO1 *
00120 *****
4000 00130 ORG $4000
00140
4000 16 0774 00150 LBRA PI
00160 *****
00170 *BUFFERS AND WORKING STORAGE *
00180 *****
00190 * CHANGE BYTES, TERM1, AND TERM2 FOR *
00200 * DIFFERENT RESOLUTIONS OF PI *
0032 00210 BYTES EQU 50 4154 BYTES - 10000 DIGITS
00B5 00220 TERM1 EQU 181 LAST DIVISOR, GRP 1
0035 00230 TERM2 EQU 53 LAST DIVISOR, GRP 2
0190 00240 BITS EQU 8*BYTES -# BITS IN MPV
0E90 00250 SUM EQU $E00 DISK GRAPHICS SCREEN
0E32 00260 ENDSUM EQU SUM+BYTES LAST BYTE OF SUM
4003 00270 RMB 1800 BUFFER FOR 10000 BYTES
470B 00280 TEMP1 EQU * START OF TEMP1 MPV
470B 00290 RMB BYTES
00300 00300 NDTMP1 EQU * LAST BYTE OF TEMP1+1
473D 00310 TEMP2 EQU * START OF TEMP2 MPV
473D 00320 RMB BYTES
476F 476F 00330 NDTMP2 EQU * LAST BYTE OF TEMP2+1
4771 00340 SCALE RMB 2 #DEC DIGITS SCALE FACTOR
4772 00350 ODDEVN RMB 1 $A9-ADCA; $A2-SBCA FOR ADDSUB
4774 00360 LAST RMB 2 HOLDS LAST DIVISOR TO BE USED
4774 00370 DIVSOR RMB 2 WORKING DIVISOR-1,3,5..LAST
4776 00380 SIGNI RMB 1 0 IF SIGNIFICANCE
00390 *****
00400 * PI DRIVER *
00410 *****
4777 CE 0E32 00420 PI LDU #ENDSUM POINT TO LS BYTE OF SUM+1
477A BD 60 00430 BSR LOADZ ZERO SUM
477C CE 473D 00440 LDU #NDTMP1 LS BYTE OF TEMP1+1
477E 108E 476F 00450 LDY #NDTMP2 LS BYTE OF TEMP2+1
4783 BD 64 00460 BSR LOADO LOAD 1 TO TEMP1 (SCALE)
4785 CC 00B5 00470 LDD #TERM1 TERMINATOR
4788 FD 4772 00480 STD LAST LAST DIVISOR
478B 86 A9 00490 LDA #A9 SET ADD FOR FIRST TERM
478D B7 4771 00500 STA ODDEVN ADD/SUB FLAG
4790 CC 0005 00510 LDD #5 FOR FIRST DIVIDE
4793 BE 0019 00520 LDX #25 FOR SUBSEQUENT DIVIDES
4796 BD 482C 00530 JSR TERM CALCULATE FIRST GROUP VALUE
00540 * NOW HAVE FIRST GROUP VALUE - MULTIPLY BY 4
4799 CE 0E32 00550 LDU #ENDSUM LS BYTE OF SUM+1
479C 17 0163 00560 LBSR SHIFT SUM<-*2
479F CE 0E32 00570 LDU #ENDSUM LS BYTE OF SUM+1
47A2 17 015D 00580 LBSR SHIFT SUM<-SUM*4
47A5 CE 473D 00590 LDU #NDTMP1 LS BYTE OF TEMP1+1
47A8 108E 476F 00600 LDY #NDTMP2 LS BYTE OF TEMP2+1
47AC 8D 3B 00610 BSR LOADO LOAD 1 TO TEMP1 (SCALE)
47AE CC 0035 00620 LDD #TERM2 TERMINATOR
47B1 FD 4772 00630 STD LAST LAST DIVISOR
47B4 86 A2 00640 LDA #A2 SET SUBTRACT FOR FIRST TERM
47B6 B7 4771 00650 STA ODDEVN ADD/SUB FLAG
47B9 CC 00EF 00660 LDD #239 FOR FIRST DIVIDE
47BC 8E DF21 00670 LDX #57121 FOR SUBSEQUENT DIVIDES
47BF 8D 6B 00680 BSR TERM CALCULATE SECOND GROUP
00690 * NOW HAVE PI/4 - MULTIPLY BY 4 AGAIN
47C1 CE 0E32 00700 LDU #ENDSUM LS BYTE OF SUM+1
47C4 17 013B 00710 LBSR SHIFT SUM<-SUM*2
47C7 CE 0E32 00720 LDU #ENDSUM LS BYTE OF SUM+1
47CA 17 0135 00730 LBSR SHIFT SUM<-SUM*4
47CD 86 FE 00740 LDA #-2 PRINTER CODE
47CF 97 6F 00750 STA $6F ROUTE OUTPUT TO LP
47D1 CE 0E32 00760 LDU #ENDSUM LS BYTE OF SUM+1
47D4 108E 476F 00770 LDY #NDTMP2 LS BYTE OF TEMP2+1
47D8 17 00B3 00780 LBSR CONVERT CONVERT AND PRINT
47DB 39 00790 RTS RETURN TO BASIC
00800 *****
00810 * LOAD 0 SUBROUTINE *
00820 * ENTRY: BYTES=SIZE OF MPV IN BYTES *
00830 * (U)-END OF MPV+1 *
00840 *****
47DC 8E 0032 00850 LOADZ LDX #BYTES GET # BYTES
47DF CC 0000 00860 LDD #0 LOAD 0
47E2 ED C3 00870 LOA010 STD --U STORE 0
47E4 30 1E 00880 LEAX -2,X DECREMENT COUNT
47E6 26 FA 00890 BNE LOA010 GO IF NOT DONE
47E8 39 00900 RTS RETURN
00910 *****
00920 * LOAD 1 AND SCALE SUBROUTINE *
00930 * ENTRY: (U)-END OF MPV+1 *

```

the divide, the quotient bits have filled up the entire MPV and the remainder of the divide is in D.

Other Modules

TERM is the subroutine that actually carries out the calculation of each term, adding or subtracting the term to SUM. TERM is called twice, once for each set of values.

CONVRT is a convert to ASCII subroutine. After all of the calculations have been done, the result is in SUM as a scaled-up value. This huge binary number must now be converted to decimal by successively dividing by 10 and saving the remainders, as shown in Figure 6. The remainders in reverse order represent the equivalent decimal number. Before they are printed, the decimal value of zero through nine is converted to ASCII by adding \$30. The remainders are stored in a buffer, one remainder per byte. This buffer is more than twice as big as an MPV, so two MPVs and additional memory is used for the conversion process.

PI is the "main driver" portion of the program. It calls the subroutines to implement the calculations and generally oversees the operations.

The Actual Assembly Language

I don't mind telling you that the PI program grew far beyond what I had imagined it would consist of. However, it's still small enough for you to follow fairly easily. Its structure follows the physical arrangement in Figure 3 and is listed in Listing 1.

There are three MPVs used in the program, designated TEMP1, TEMP2 and SUM. To change the size of the MPVs, set the parameter Bytes to the number of bytes required. The number of Bytes must be even due to the methods of storage and access in the program. I used values from four up to hundreds. Maximum size is on the order of 4,154 bytes. Remember that changing the number of bytes also requires that you change the TERM1 and TERM2 "LAST" values. Too large a value for LAST and you'll be wasting time; too small a value and you will lose digits in the answer.

The SUM MPV is ORGed at the graphics display area (set this to \$600 for cassette systems). More about that later, but as you might guess, locating the SUM MPV in the graphics screen area allows you to see PI being computed.

The program is ORGed at \$4000. You can change this to below the \$4000 area if you have a 16K system, but will not be able to use larger MPVs. (The program is about 400 bytes exclusive of the

```

00940 * (Y)-END OF TEMP MPV+1 *
00950 * EXIT: 1 * SCALE FACTOR LOADED *
00960 * SCALE SET TO SCALE FACTOR *
00970 *****
47E9 34 60 00980 LOADO PSHS Y,U SAVE POINTERS
47EB 8D EF 00990 BSR LOADZ ZERO MPV
47ED 35 60 01000 PULS Y,U RESTORE POINTERS
47EF 86 01 01010 LDA #1 1
47F1 A7 5F 01020 STA -1,U STORE FOR SHIFT AND ADD
47F3 7F 476F 01030 CLR SCALE RESET SCALE FACTOR
47F6 34 60 01040 LON010 PSHS Y,U SAVE POINTERS
47F8 17 0107 01050 LBSR SHIFT X2
47FB 35 60 01060 PULS Y,U GET POINTERS
47FD 34 60 01070 PSHS Y,U SAVE POINTERS
47FF 1E 23 01080 EXG Y,U SWAP Y,U
4801 17 00B0 01090 LBSR MOVE SAVE X2
4804 35 60 01100 PULS Y,U GET POINTERS
4806 34 60 01110 PSHS Y,U SAVE POINTERS
4808 17 00F7 01120 LBSR SHIFT X4
480B 35 60 01130 PULS Y,U RESTORE POINTERS
480D 34 60 01140 PSHS Y,U SAVE POINTERS
480F 17 00FF 01150 LBSR SHIFT X8
.2 35 60 01160 PULS Y,U RESTORE POINTERS
4814 34 60 01170 PSHS Y,U SAVE POINTERS
4816 86 A9 01180 LDA #9A9 ADD CODE
4818 BD 48EF 01190 JSR ADDSUB X8 + X2 = X10
481B 35 60 01200 PULS Y,U GET POINTERS
481D FC 476F 01210 LDD SCALE SCALE IS COUNT OF DIGITS
4820 C3 0001 01220 ADDD #1 BUMP BY 1
4823 FD 476F 01230 STD SCALE STORE
4826 A6 CB CE 01240 LDA -BYTES,U POINT TO FIRST BYTE
4829 27 CB 01250 BEQ LON010 GO IF NOT NORMALIZED
482B 39 01260 RTS RETURN
01270 *****
01280 * CALCULATE TERM SUBROUTINE *
01290 * ENTRY: (LAST)-LAST DIVISOR VALUE *
01300 * (D)-FIRST DIVISOR - 5/239 *
01310 * (X)-SUBSEQUENT DIVISORS *
01320 * (ODDEVN)-ADD OR SUB OP CODE *
01330 * EXIT: (SUM) HOLDS FINAL VALUE *
01340 *****
482C FD 484D 01350 TERM STD TPI020+1 STORE 5 OR 239
482F BF 4848 01360 STX TOP1+1 STORE 25 OR 239*239
4832 CC 0001 01370 LDD #1 SET DIVISOR TO 1
4835 FD 4774 01380 STD DIVSOR STORE DIVISOR
4838 FC 4774 01390 TPI010 LDD DIVSOR GET DIVISOR
483B 10B3 4772 01400 CMPD LAST TEST FOR LAST
483F 22 4C 01410 BHI TPI000 GO IF DONE
4841 1083 0001 01420 CMPD #1 TEST FOR DIVISOR=1
4845 27 05 01430 BEQ TPI020 GO IF SO
4847 CC 0000 01440 TOP1 LDD #0 SQUARED TERM
484A 20 03 01450 BRA TPI030 CONTINUE
484C CC 0000 01460 TPI020 LDD #0 SQ RT TERM
484F CE 473D 01470 TPI030 LDU #NDTMP1 LS BYTE OF TEMP1+1
4852 BD 6C 01480 BSR DIVIDE TEMP1/5 OR TEMP1/25
4854 CE 476F 01490 LDU #NDTMP2 LS BYTE OF TEMP2+1
4857 108E 473D 01500 LDY #NDTMP1 LS BYTE OF TEMP1+1
485B 8D 57 01510 BSR MOVE TEMP2<=TEMP1
485D FC 4774 01520 LDD DIVSOR GET CURRENT DIVISOR
4860 CE 476F 01530 LDU #NDTMP2 LS BYTE OF TEMP2
4863 8D 5B 01540 BSR DIVIDE TEMP2<=TEMP2/DIVISOR
4865 CE 0E32 01550 LDU #ENDSUM LS BYTE OF SUM+1
4868 108E 476F 01560 LDY #NDTMP2 LS BYTE OF TEMP2+1
486C B6 4771 01570 LDA ODDEVN GET ADD/SUB FLAG
486F 17 007D 01580 LBSR ADDSUB ADD/SUB TO TEMP2 TO SUM
4872 FC 4774 01590 LDD DIVSOR GET CURRENT DIVISOR
4875 C3 0002 01600 ADDD #2 DIVISOR<=DIVISOR+2
4878 FD 4774 01610 STD DIVSOR SAVE FOR NEXT DIVIDE
487B B6 4771 01620 LDA ODDEVN GET ADD/SUB FLAG
487E 81 A9 01630 CMPA #9A9 ADD?
4880 27 04 01640 BEQ TPI040 GO IF ADD
4882 86 A9 01650 LDA #9A9 SET TO ADD
4884 20 02 01660 BRA TPI050 CONTINUE
4886 86 A2 01670 TPI040 LDA #9A2 SET TO SUB
4888 B7 4771 01680 TPI050 STA ODDEVN STORE FOR NEXT ADD/SUB
488B 20 AB 01690 BRA TPI010 LOOP
488D 39 01700 TPI000 RTS RETURN
01710 *****
01720 * CONVERT MPV TO DECIMAL AND PRINT *
01730 * ENTRY: (U)-END OF SOURCE MPV+1 *
01740 * (Y)-END OF DEST+1 2X MPV + *
01750 * EXIT: SOURCE DIGITS PRINTED IN DEC *
01760 *****
488E 34 20 01770 CONVRT PSHS Y SAVE END OF DEST AREA
4890 CC 000A 01780 CON010 LDD #10 DIVISOR FOR CONVERT
4893 34 60 01790 PSHS U,Y SAVE PARAMETERS
4895 8D 29 01800 BSR DIVIDE X/10

```

MPV areas.) The version shown was assembled with the Radio Shack Disk Assembler (26-3254). Use the /SR option and assemble to disk as *PINEW/BIN*.

A BASIC driver is shown in Listing 2. This driver simply loads the *PINEW/BIN* module, sets the graphics mode, then executes the *PI* program.

#### Listing 2: PIBASIC

```

100 CLEAR 200, &H3FFF
110 LOADM "PINEW/BIN"
120 DEF USR0=&H4000
130 SCREEN 1,1
140 PMODE 1
150 PCLS
160 A=USR0(0)
170 GOTO 170

```

#### Program Notes

Many of the subroutines use address pointers passed in Register U registers or U and Y. These pointers usually point to the end of an MPV plus '1'. The reason for this is that *auto-decrementing* is used in many subroutines. A pointer to the MPV points to the low-order byte, as operations such as multiple-precision adds and subtracts and shifts start there. An instruction such as STD, --U (LOADZ) decrements Register U *before* the instruction is executed. After the decrement, the pointer points to the last 16-bit value. An autodecrement of ,U works similarly, except the pointer points to the last byte after the auto-decrement. Using auto-decrement saves an LEAU, -1,U or similar instruction.

The name of the game in much of this code is efficiency, especially in subroutines such as SHIFT and ADDSUB, which are used constantly. In DIVIDE, for example, good programming practice calls for using a divisor on the stack, or at worst, a memory variable. However, the divisor in DIVIDE is stored in an immediate instruction field just for the sake of squeezing out some extra speed by eliminating the instructions required in the multiply loop.

The LOADO subroutine uses a multiply of times 10 after initially storing a '1'. We could have used a generic multiply subroutine here, just as the DIVIDE is a generic divide. However, we used a well-known trick multiply — a type of shift and add. Multiplying by 10 is equivalent to shifting one bit left, saving the result as X2, shifting twice again to get eight times the original value, then adding the saved X2 term. The shift is done by the SHIFT subroutine, which we already have, and the multiple-precision add by ADDSUB.

The ADDSUB subroutine stores the

```

4897 35 60 01810 PULS U,Y RESTORE
4899 E7 A2 01820 STB ,-Y SAVE REMAINDER
489B B6 4776 01830 LDA SIGNI GET SIGNIF FLAG
489E 27 F0 01840 BEQ CON010 GO IF Q NOT 0
01850 * NOW PRINT FROM (Y)
48A0 A6 A0 01860 CON020 LDA ,Y+ GET DIGIT 0-9
48A2 8B 30 01870 ADDA #030 CONVERT TO ASCII
48A4 34 20 01880 PSHS Y SAVE POINTER
48A6 AD 9F A002 01890 JSR [A002] PRINT
48AA 35 20 01900 PULS Y RESTORE POINTER
48AC 10AC E4 01910 CMPLY ,S TEST FOR END
48AF 23 EF 01920 BLS CON020 GO IF NOT END
48B1 35 20 01930 PULS Y RESET STACK
48B3 39 01940 RTS RETURN
01950 *****
01960 * MOVE ONE MPV TO ANOTHER *
01970 * ENTRY: BYTES=SIZE OF MPV IN BYTES *
01980 * (U)=END OF DEST MPV+1 *
01990 * (Y)=END OF SOURCE MPV+1 *
02000 * EXIT: DEST MPV NOW=SOURCE MPV *
02010 *****
48B4 8E 0032 02020 MOVE LDX #BYTES GET # BYTES
48B7 EC A3 02030 MOV010 LDD ,--Y GET SOURCE WORD
48B9 ED C3 02040 STD --U STORE IN DEST WORD
48BB 30 1E 02050 LEAX -2,X DECREMENT COUNT
48BD 26 F8 02060 BNE MOV010 GO IF NOT DONE
48BF 39 02070 RTS RETURN
02080 *****
02090 * DIVIDE MPV BY 16 BIT DIVISOR *
02100 * ENTRY: BYTES=SIZE OF MPV IN BYTES *
02110 * BITS=SIZE OF MPV IN BITS *
02120 * (U)=END OF MPV+1 *
02130 * (D)=DIVISOR *
02140 * EXIT: (MPV)=QUOTIENT *
02150 * (D)=REMAINDER *
02160 *****
48C0 FD 48DE 02170 DIVIDE STD OP1+2 SETUP DIVISOR
48C3 FD 48E3 02180 STD OP2+1 SETUP DIVISOR
48C6 108E 0190 02190 LDY #BITS LOAD # BITS IN MPV
48CA 86 01 02200 LDA #1 SET SIGNI TO NONE
48CC B7 4776 02210 STA SIGNI SIGNIFICANCE FLAG
48CF C0 0000 02220 LDD #0 INITIALIZE MS BYTE
48D2 34 40 02230 DIV010 PSHS U SAVE MPV ADDRESS
48D4 8D 2C 02240 BSR SHIFT SHIFT MPV
48D6 35 40 02250 PULS U RESTORE ADDRESS
48D8 59 02260 ROLB LAST BIT TO D
48D9 49 02270 ROLA HIGH-ORDER BYTE
48DA 25 06 02280 BCS OP2 GO IF HIGH-ORDER BIT
48DC 1083 0000 02290 OP1 CMPD #0 DUMMY-TEST DIVISOR
48E0 25 08 02300 BLO DIV020 GO IF NO GO
48E2 83 0000 02310 OP2 SUBD #0 DUMMY-SUBTRACT DIVISOR
48E5 6C 5F 02320 INC -1,U SET Q=1
48E7 7F 4776 02330 CLR SIGNI SET SIGNIFICANCE
48EA 31 3F 02340 DIV020 LEAY -1,Y DECREMENT # BITS
48EC 26 E4 02350 BNE DIV010 GO IF MORE
48EE 39 02360 RTS RETURN
02370 *****
02380 * MULTIPLE-PRECISION ADD OR SUBTRACT *
02390 * ENTRY: BYTES=SIZE OF MPV IN BYTES *
02400 * (U)=END OF DEST MPV+1 *
02410 * (Y)=END OF SOURCE MPV+1 *
02420 * (A)=$A9 IF ADD, $A2 IF SUB *
02430 * EXIT: DEST MPV-DEST MPV-SOURCE MPV *
02440 *****
48EF B7 48F9 02450 ADDSUB STA OP3 SET MODE
48F2 1C FE 02460 ANDCC #0FE RESET C
48F4 8E 0032 02470 LDX #BYTES GET # BYTES
48F7 A6 C2 02480 ADD010 LDA ,-U GET DEST BYTE
48F9 A9 A2 02490 OP3 ADCA ,-Y ADD OR SUB SOURCE
48FB A7 C4 02500 STA ,U DECREMENT AND STORE
48FD 30 1F 02510 LEAX -1,X DECREMENT COUNT
48FF 26 F6 02520 BNE ADD010 GO IF NOT DONE
4901 39 02530 RTS RETURN
02540 *****
02550 * SHIFT MPV LEFT ONE BIT *
02560 * ENTRY: BYTES=SIZE OF MPV IN BYTES *
02570 * (U)=END OF MPV+1 *
02580 * EXIT: (MPV)<=MPV SHIFTED LEFT *
02590 * C=MSB *
02600 *****
4902 1C FE 02610 SHIFT ANDCC #0FE RESET C
4904 8E 0032 02620 LDX #BYTES GET # BYTES
4907 69 C2 02630 SHIF010 ROL ,-U SHIFT LEFT ONE BIT
4909 30 1F 02640 LEAX -1,X DECREMENT COUNT
490B 26 FA 02650 BNE SHIF010 GO IF NOT 0
490D 39 02660 RTS RETURN
02670 END

```

flag for add or subtract into the OP3 instruction. The flag is also the actual opcode used for ADCA or SBCA. The two instructions are the same format otherwise.

The CONVERT subroutine uses a common method of converting from binary to decimal. Divide any binary number by 10. Save the remainder and divide the quotient by 10 again. Continue this process until the quotient reduces to zero. The remainders, in reverse order, are the equivalent decimal number. In this case we don't need to know how many digits are in the number to be converted, because that's handled automatically — the process stops when the quotient (residue, really) disappears.

We could have added a decimal point after the first decimal digit in printing out the answer — it must be '3'. However, we left it off in this version, having been beaten into apathy by the program. It might also have been nice to format the printout by arranging the data into groups of 10 digits. (All of this will be done in the next version.)

### Seeing the Calculations

As I mentioned, *PI* actually lets you see the SUM term being calculated. When *PI* is executed, you'll see the screen clear. No further screen activity occurs while LOAD0 loads a scaled '1' into the first MPV. After this fill, however, the screen is partially filled with data. The amount of data depends upon the number of bytes you have selected for the MPVs and the screen mode used in the BASIC driver. The screen data represents the first value in SUM — the first term of the first set. As the program number crunches away, you'll see the data change, each change occurring to portions of the data further and further to the right and down, representing less and less significant digits. If you're running a version of the program that prints many digits, with significant time delays between term updates, keep your eye on a spot on the screen until the next change to zero-in on the digit positions being modified.

After the first set of data has been calculated, you'll see the next set of terms changing the data area. This set converges more rapidly. It starts from the high-order digits as well, but moves much faster through the screen area.

After the second set of data has been processed, the program converts SUM (the screen) to ASCII. In doing so, it successively divides SUM by 10, making SUM smaller and smaller. When this happens, you'll see a blank portion in the data area growing and snaking

through the data area as more and more leading zeros are produced in SUM. When SUM is reduced down to zero, the ASCII digits are printed from the temporary ASCII buffer.

**How Fast is PI and How does the CoCo Compare to Other Systems?**

PI is a fairly efficient program, but it's obvious that our CoCo is no Cray X-MP! It takes about three hours to

generate and print 500 decimal digits and about 23 hours for 1,000 digits (see Figure 7). Mainframe computers typically generate thousands of digits of pi in six to eight hours. For smaller MPVs, the speed drops dramatically and the screen is fun to watch. (As I write this, I'm running PI for 2,000 digits — estimated completion time is 240 hours.)

Could the PI program be sped up?

Undoubtedly, both by more efficient code and by more efficient algorithms. This is a first attempt at the problem of generating pi and, frankly, it turned out to be an interesting project. It's one of those programming problems that simply can't be done in anything other than assembly language, where speed is all important. A perfect project for one of those silly hackers who can't think of anything but computers.

**Figure 7: PI to 1,000 Decimal Digits**

```

31415926535897932384626433832795028841971693993751058209749445923078164062862089
9862803482534211706798214808651328230664709384460955058223172535940812848111745
02841027019385211055596446229489549303819644288109756659334461284756482337867831
65271201909145648566923460348610454326648213393607260249141273724587006606315588
17488152092096282925409171536436789259036001133053054882046652138414695194151160
94330572703657595919530921861173819326117931051185480744623799627495673518857527
2489122793818301194912983367336244065664308602139494639522473719070217986094370
27705392171762931767523846748184676694051320005681271452635608277857713427577896
0917363717872146844090122495343014654958537105079227968925892354201995611212902
1960864034418159813629774771309960518707211349999983729780499510597317328160963
18595024459455346908302642522308253344685035261931188171010003137838752886587533
20838142061717766914730359825349042875546873115956286388235378759375195778185778
05321712268066130019278766111959092164201989380952572010654858632788659361533818
27968230301952035187124
  
```

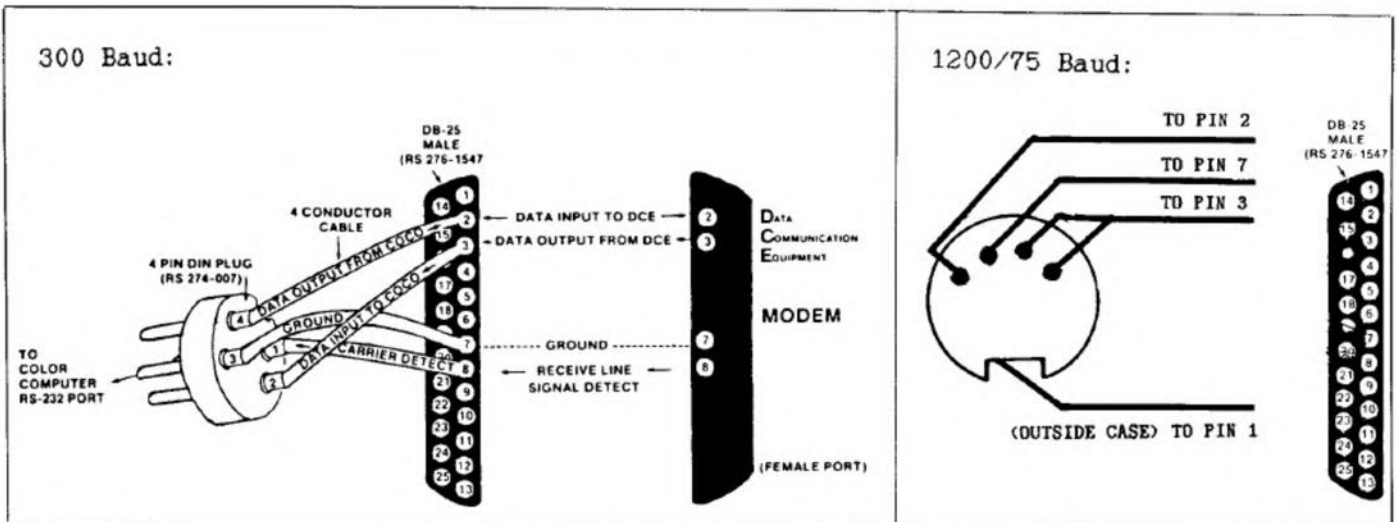
— Last Few Digits Invalid

# RS-232 INTERFACING

by Graham Morphett

With excellent and low cost equipment like the Avtek Mini Modem (Paris Radio, approx \$210) around these days, the time is ripe to get into communicating with your computer.

The following diagrams illustrate how to connect your CoCo's RS-232 port to the Avtek Mini modem for both 300 baud and 1200/75 baud communication.





# PATTERNS OF MANY COLOURS

by John Redmond

Last month, we started to devise a new graphics language, XGAL, and we managed to solve most of the painful problems associated with getting the computer to operate in the right graphics mode. The trouble was, though, that we didn't get around to actually doing anything on the screen. Now is the time to remedy all that - we can start drawing things.

Before anything can be put on a graphics screen, we have to have a word that can plot points. It is called, quite appropriately, PLOT and it expects on the stack the x and y coordinates of the desired location. Simple enough? No, it isn't. It involves a great deal of hard work AND it involves a couple of important policy decisions. Consider these questions:

1. where is point (0,0)?
2. what happens if we try to plot that is off the screen?

The first question relates to the mapping of the points. Most commonly, the origin (0,0) is at the top left-hand corner of the screen. This is because of quite trivial programming convenience and goes against what we learnt in maths at school. The origin should be at the bottom left-hand corner and that's where we will put it. A small difference, you might think, but it really isn't. When we start to use the Turtle graphics routines and want to set the directions in degrees, it is very confusing to think upside-down all the time. Let's do it properly from the start.

It is essential to have a way of handling the plotting of off-screen points. Remember that, when we plot, we are poking byte values into memory. If a point is outside the legitimate bounds of screen memory, you might easily find yourself bashing critical parts of RAM, like where Forth is sitting. It happens, believe me! So, what do we do? One way is to trim the coordinates to fit the screen. This is what ECB does with the circle statement - if the circle is too big, it simply draws a border. I've tried this and found that the cluttered borders are unsightly. What then? Well, we could break out of the program and issue an error message, as Basic loves to do. Dreadful! The best way is to simply not plot the offending point and continue on with the processing of the program.

That is just what the word OFFSCREEN? does. If the coordinate is out of range, the two stack values are thrown away (2DROP) and the top entry on the RETURN stack (NOT the DATA stack) is also dropped (using R> DROP) before EXIT is invoked. Now, here's the clever part. OFFSCREEN? is used inside PLOT. Therefore, if we do a normal EXIT from OFFSCREEN?, we would return to PLOT and execute the next word there, which is MAP. But we don't want to do that; so, by first dropping the top entry of the return stack, we manage to EXIT from the word that called OFFSCREEN?, which is PLOT. In other words, we have EXITed through two levels of nesting. Any other language would have to return a flag which would tell PLOT that there has been an error.

Mind-bending stuff, this. There is just SO much to Forth. It is power, not a straight-jacket like Pascal or C. End of digression, end of homily.

If OFFSCREEN? is not offended by the x or y coordinate, it simply subtracts the y value from 191 (OBF HEX) and returns two values. MAP then takes over. It calls on other words which have a lot of hard work to do. First of all, YOFFSET has to look at HEIGHT C@. Now this value is a hybrid. If the value is 0, it is taken as a false flag and no division is done. If, however, it is non-zero, it represents the vertical size of a pixel (look back at the values in GRAPHICSARRAY). In such an event, it is necessary to divide the y coordinate by this value. Finally, YOFFSET multiplies this corrected value by the number of bytes in one screen line (B/LINE C@) and adds the result to the base address of video memory.

Then XOFFSET takes over. It calculates a further address offset to be added to what YOFFSET produced. The >> word carries out a right shift on the x value to get the offset. Then MASK scratches its head and decides upon a byte mask. In some cases, where WIDTH C@ gives a value of 4, there is some bit juggling before the correct mask can be chosen. After all this is over, MAP is able to return the desired byte mask and video address. But PLOT is not yet finished.

(PLOT) has a balancing act to do. Not only must it insert the desired pixel, in the right colour, at the right place on the screen: it must also retain the other pixels already coded for in the byte. To retain the pixels, a reverse byte mask is needed, so the original mask is copied to the top of the stack with OVER and reversed by exclusive-or-ing with OFF HEX. This negative mask is then applied to the byte already in memory by the sequence OVER C@ AND. The original (positive) mask is then applied to the colour template in (INK), ORed with the negatively-masked value and replaced in memory. With that, (PLOT) and PLOT are finished.

I think you will agree that PLOTting is not a trivial matter. The very complexity of all this is, I think, the reason why commercial programs have not exploited the graphics capabilities of the CoCo. One final comment before we move on: notice the factoring of the words within MAP and PLOT, i.e., YOFFSET, MASK, (PLOT), etc. Could you convert these to assembly code? It is VERY easy, but why bother? Well, it turns out that nearly all the time taken to draw something on the screen is taken up by PLOT. If, therefore, we can speed up the words that PLOT uses by converting them to inline machine code, all of our graphics will be speeded up. This must wait for a later article. Suffice it to say that we can do things at least four to five times faster.

Now to draw something. Let's start with a circle which, believe it or not, is easier to draw than a straight line. There are some clever ways to draw circles but, for the moment, we will use the simplest: given an x offset, we will calculate the y offset using Pythagoras's theorem. Therefore, we

need a square root routine. I talked about this word in an earlier article and I fear that I was misunderstood. This SQRT word is not the ultimate in square root routines. I devised it to carry out fairly quick processing of single-length integers for just the present job - graphics on a screen of moderate resolution. A more sophisticated word would be slower and inappropriate. Horses for courses.

SQRT is a lot faster than the Basic ROM, but it involves division, and division is by far the slowest thing that you can do with a 6809. So let's make the most of the square roots we calculate and exploit the symmetry of a circle. That is what all the +X+Y, +X-Y, etc. are about. Each square root is then used eight times! It's just a very small blow for efficiency. Efficiency is a very important matter for Forth programmers and we will be talking quite a lot about it in future articles.

Notice how CIRCLE decides how many coordinate pairs to calculate. It uses a loop and takes the value stored in (RADIUS) and approximately divides it by the square root of 2 (remember your surds?). By geometry, that is as far out from the x coordinate of the centre that we need to go.

In conjunction with the words CENTRE and RADIUS, CIRCLE can go a fair way towards understanding plain English. The fact that the location and radius are stored, too, can make for more natural programming. We change them only when we want to. The following will draw concentric circles of different sizes and colours:

```
128 96 CENTRE
10 RADIUS YELLOW INK DRAW CIRCLE
15 RADIUS BLUE INK DRAW CIRCLE
20 RADIUS RED INK DRAW CIRCLE, etc.
```

Now for the straight line. In Cartesian graphics, we have to give it a beginning and an end, using START and END. That's easy; what we have to do now is fill in between these ends using PLOT. But how many times do we use PLOT? If a line is from (0,0) to (10,10), the answer is easy: eleven (= 10 + 1) points total. But what about the line from (0,0) to (10,20)? Now we need 21 points! If we plot only eleven, we get gaps, because the y values are changing twice as fast as the x values.

This will always happen if the gradient is greater than 1, i.e., if DY (delta Y) is greater than DX (delta X). (School geometry is relevant after

all!) Now look at the definition of LINE. It asks STEEP to determine whether the absolute value of M is greater than 1. If it is, LINE calls upon YLINE to use Y as the independent variable: otherwise X is the independent variable and XLINE is used. In this way, all lines, in all directions, are drawn properly without gaps.

XLINE and YLINE are very busy words and they do a lot of work. Before doing a spot of stack juggling to get the right loop indices, they have to find a way of avoiding repeated time-consuming divisions inside the loop that draws the line. They call upon GRAD to scale the gradient by multiplying by 128 (80 HEX), save it in M and initialize an accumulator (DELTA) as 0. Each time through the loop, the latest value in DELTA is fetched by +DELTA, divided by 128 (using 7 right shifts) and added to the initial value on the stack. On the side, too, the value in M is added to DELTA ready for the next time through the loop. A bit complicated, I grant, but a lot faster than using DY/DX as a rational fraction and having a slow \*/ inside the loop.

Well, we're there! Now we can draw lines and circles all over the screen in any graphics mode. Compile the program and type in from the keyboard:

```
MEDIUM SEMI GRAPHICS
NEW DISPLAY 20 CIRCLES 10 LINES
```

CIRCLES and LINES are two simple demo words, given at the bottom of the listing. Look at the postfix notation in the above commands. Would you really have preferred to type:

```
GRAPHICS(SEMI(MEDIUM))
DISPLAY(NEW) CIRCLES(20) LINES(10) ?
```

These demo words use a couple of other definitions - CHOOSE is very similar to a definition in 'Starting Forth' and produces a random number from zero to one less than the number it is given. WAIT is there, too, at the end of CIRCLES and LINES. It will sit and wait for you to touch a key before continuing.

Well now, all the unpleasant work is over in the development of XGAL. The Turtle and String graphics are still to come, but they are really VERY easy. Don't wait for me - start on it yourself. If you don't wet your fingers with Forth code, you'll never get to love the language.

```
\ XGAL - EXTENDED GRAPHICS
\ APPLICATIONS LANGUAGE,
\ VERSION 1.05,
\ <C> JOHN REDMOND, 1986.
\ PART 2:
\ WORDS FOR PLACING A POINT
\ ON THE SCREEN.
: ADJUST ( N--N' )
WIDTH C@ 3 >
IF 2/ THEN ;
: MASK ( X--MASK )
P/BYTE @ AND ADJUST
MASKSETS COUNT + + C@ ;
: YOFFSET ( Y--ADDRESS )
HEIGHT C@ ?DUP
IF / THEN
B/LINE C@ *
VBASE + ;
: XOFFSET ( X--OFFSET )
WIDTH C@ >> ;
: MAP ( X,Y--MASK,ADDR )
YOFFSET OVER XOFFSET +
SWAP MASK SWAP ;
: ONSCREEN? ( X,Y--X,Y' )
DUP 0< OVER OBF > OR
IF 2DROP R> DROP EXIT THEN
SWAP DUP 0< OVER OFF > OR
IF 2DROP R> DROP EXIT THEN
OBF ROT - ;
: (PLOT) ( MASK,ADDR )
OVER OFF XOR
OVER C@ AND
ROT (INK) @ AND OR
SWAP C! ;
: PLOT ( XCOORD,YCOORD-- )
ONSCREEN? MAP (PLOT) ;
\ WORDS FOR CIRCLE DISPLAY.
VARIABLE (CENTRE) 2 ALLOT
VARIABLE (RADIUS)
VARIABLE RSQUARE
: SQRT ( N--N' )
DUP 5 >> 1+
BEGIN 2DUP / OVER -
DUP ABS 1 >
WHILE 2/ + REPEAT
ROT 2DROP ;
: CENTRE ( X,Y )
(CENTRE) 2! ;
: RADIUS ( N )
DUP (RADIUS) !
DUP * RSQUARE ! ;
\ ( DX,DY -- X,Y ) FOR EACH OF:
: +X+Y (CENTRE) 2@ ROT +
SWAP ROT + SWAP PLOT ;
: +X-Y (CENTRE) 2@ ROT -
SWAP ROT + SWAP PLOT ;
: -X+Y (CENTRE) 2@ ROT +
SWAP ROT - SWAP PLOT ;
: -X-Y (CENTRE) 2@ ROT -
SWAP ROT - SWAP PLOT ;
: COORDS ( XVAL--YVAL )
```

```

RSQUARE @
OVER DUP * - SQRT ;

DECIMAL
: CIRCLE
(RADIUS) @ 71 100 */ 1+ 0
DO I COORDS
2DUP +X+Y
2DUP SWAP +X+Y
2DUP +X-Y
2DUP SWAP +X-Y
2DUP -X+Y
2DUP SWAP -X+Y
2DUP -X-Y
SWAP -X-Y
LOOP ;

```

\ WORDS FOR LINE DISPLAY.

```

HEX
VARIABLE (START) 2 ALLOT
VARIABLE (END) 2 ALLOT
VARIABLE DX
VARIABLE DY
VARIABLE M
VARIABLE DELTA
VARIABLE SIGN

: START (START) 2! ;
: END (END) 2! ;
: DRAW ; \ JUST A FILLER

: DELTAS
(START) 2@ (END) 2@

```

```

ROT - DY !
SWAP - DX ! ;

: STEEP DX @ ABS DY @ ABS < ;

: GRAD 80 ROT ROT */
DUP ABS M ! 0< SIGN !
0 DELTA ! ;

: +DELTA DELTA @
M @ DELTA +!
?DUP 0= IF EXIT THEN
7 >>
SIGN @ IF NEGATE THEN + ;

: LIMITS (END) 2@ (START) 2@ ;

: XLINE DY @ DX @ GRAD
LIMITS DX @ 0>
IF 2SWAP THEN DROP 1+ ROT
DO I OVER +DELTA PLOT LOOP
DROP ;

: YLINE DX @ DY @ GRAD
LIMITS DY @ 0 >
IF 2SWAP THEN
SWAP DROP 1+ SWAP
DO DUP +DELTA I PLOT
LOOP DROP ;

: LINE
DELTAS STEEP
IF YLINE ELSE XLINE THEN ;

```

\ EXAMPLES OF USE:

```

\ ORANGE INK BUFF PAPER
\ 23 60 CENTRE 14 RADIUS
\ NEW DISPLAY
\ DRAW CIRCLE
\ 0 0 START 39 22 END
\ OLD DISPLAY
\ DRAW LINE

VARIABLE RND HERE RND !
: RANDOM RND @ 31421 * 6927 +
DUP RND ! ;
: CHOOSE ( U1--U2)
2* RANDOM M* DABS SWAP DROP ;

: WAIT INKEY DROP ;

DECIMAL
: CIRCLES
0 DO
7 CHOOSE COLOURED INK
256 CHOOSE 192 CHOOSE CENTRE
45 CHOOSE RADIUS
DRAW CIRCLE
LOOP WAIT ;

: LINES
0 DO
7 CHOOSE COLOURED INK
255 CHOOSE 192 CHOOSE START
255 CHOOSE 192 CHOOSE END
DRAW LINE
LOOP WAIT ;

```

# ADVENTURE GAMES



What's on  
The Best of CoCoOz #5

ADV 32K/16K ..... S. RAYNER  
(Restore a nuclear power plant)

ORBQUEST ..... TONY PARFITT  
(Save your people from evil!)

LABYRINT ..... JAMES RAYMOND  
(Try to get out of this one!)

ADVENTURE + ..... SEAN HOWE  
(Find wealth!)

QUEST ..... C&K SPRINGETT  
(Seek out the 3 crystals.)

PRISON ..... TIM ALTON  
(You gotta get out, see..)

OPALTON ..... IAN CLARKE  
(A mining adventure.)

WIZSOC ..... DARRELL BERRY  
(Get free before time runs out!)

TREASURE ..... C DAVISON ET AL  
(Find the Warlock's possessions)

PRICE	Tape	\$10.00
	Disk	\$21.95

NOTE: Most of these games  
are 32K or 64K.

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

# VERY BASICALLY OS9

by Jack Fricker

When we left off last month you had two backup copies of your distribution disk. This month we are going to fix a couple of bugs in the RS232 and PRINTER modules.

We will use the accompanying listing. Just type in the listing as shown, this is for version 1.01 only. If you have version 1.0 then you need to type in the listing for 1.00, for version 2.0 you don't need to modify the drivers because they have already been fixed.

After typing in the listing you will have to verify the modified modules. To do this just type the following.

```
save /d0/newrs232 rs232
save /d0/newprinter printer
verify u </d0/newrs232 >/d0/rs232
verify u </d0/newprinter >/d0/printer
```

What verify does is get the module from the input pathname (</d0/newrs232). Then it calculates a new CRC value and writes the new module with the new CRC to the output pathname (>/d0/rs232). The CRC(cyclic redundancy check) is a value that is calculated on the size of the module. OS-9 requires that a module have a header with a correct CRC before it will allow you to execute or use that module.

The less than (<) and greater than (>) symbols are used to redirect the input and output file names. Verify must be used with the input redirected and if using the update option (u) the output must be redirected as well. If you do not use them verify will hang up the system and you will need to reboot.

Your module directory (the files in memory now) will probably look very like this.

```
Module Directory at 10:14:58
Os9   Os9p2  Init
Boot  CCDisk  D0
D1    D2     D3
Term  Ioman  Rbf
Scf   SysGo  Clock
Shell P      Pipe
Piper Pipeman Printer
Rs232 T1     T2
Aciapak Mdir
```

The above Mdir is for version 1.1. Unless you are running a terminal or a direct connect modem you do not need t1, t2 or Aciapak, if you only have 2 drives you also don't need D2 and D3.

To create a new system disk use the following as a guide.

```
Save Temp1 os9 os9p2 init boot ccdisk
Save Temp2 d0 d1 term ioman rbf scf
Save temp3 sysgo clock shell p pipe piper pipeman
```

Do NOT save either rs232 or printer. If you have double sided 40 track drives you will probably need a program called sdisk which is available

```
Build Bootlist
/d0/temp1
/d0/temp2
/d0/temp3
/d0/rs232
/d0/printer
```

Now use format to create a blank disk in drive 1. Then use "os9gen /d1 </d0/Bootlist" to create a new disk that can be booted from Disk Colour Basic.

If you do not have 40 track double sided disks you may need to create boot disks for specific languages.

For example if you are using BACIC09 it is highly unlikely that you will have any use for the following files and directories in your basic09 system disk. You wouldn't need the defs directory, but you will need that on your assembly language disk. You won't need the sys files except for motd & errmsg. Errmsg is the file that PRINTERR looks for when an error is generated and prints an expanded message on your screen. Errmsg must reside on drive 0.

If you have a printer you may prefer to list errmsg to the printer and keep it near the terminal. The list is also in the rear of the red commands manual( you did remember to read it at least twice).

Next you use DSAVE to create a file to copy the entire disk.

```
DSAVE #20k -s20k /d0 >/d0/copylist
```

This line introduces 2 new options that we haven't used up to date the #20k and -s20k. The #20k allows dsave to use 20k of memory. This option is available to nearly all OS9 programs.

The -s20k will pass the 20k option onto the copy cammands when they are created. This saves you using the editor to do it.

Use the editor to remove references in the copylist to programs that you do not wish copied over to the new disks. Then use the entire procedure to create a new disk for Basic09 , C , assembler, pascal or anything else.

Before you actually copy everything over, you should edit the startup file on the disk so that you do not have to type tmode every time to enable lower case. To do this type the following.

```
Tmode .1 -upc
```

Version 1.01	=99	=AC	=58
	=C0	=FC	L PRINTER
DEBUG	=94	=FA	. .+BB
L PRINTER	L RS232	Q	=F2
. .+65	. .+B2		L RS232
=04	=04	Version 1.00	. .+72
=82	=82		=04
=01	=01	DEBUG	=82
=A2	=A2	L PRINTER	=01
=00	=00	. .+65	=A2
=CD	=CD	=04	=00
=00	=00	=82	=CD
=63	=63	=01	=00
=00	=00	=A2	=63
=2D	=2D	=00	=00
=00	=00	=CD	=20
=13	=13	=00	=00
=00	=00	=63	=13
=05	=05	=00	=00
L PRINTER	L RS232	=2D	=05
. .+9C	. .+E1	=00	L RS232
=12	=12	=13	. .+B2
=C6	=C6	=00	=12
=00	=00	=05	=C6
=59	=59	L PRINTER	=00
=58	=58	. .+A5	=59
L PRINTER	L RS232	=12	=58
. .+AA	. .+EF	=C6	L RS232
=F2	=42	=00	. .+BA
L PRINTER	L RS232	=59	=F2
. .+11F	. .+11B		Q

continued  
from Page 37

```

460 PRINT" 4. TYPE EACH LETTER
NEEDED FOR WORD.":PRINT" 5.
PRESS ENTER.":PRINT" 6. TO END
GAME TYPE '&'.":PRINT@450,"TO R
ESUME PLAY, PRESS ENTER."
470 GOSUB380:IFASC(X$)<>13THEN47
OELSEIFZ=0THEN160ELSE200
480 FORI=0TO1500:NEXT:T$="T5L8D0
2BB-BGF+GL4DP8":U$=T$+"T5L8D02BB
-BGF+GL4EP8":PLAYU$:RETURN
490 C=USR2(0):PRINT@484,"TO RESU
ME PRESS ANY KEY.";
500 GOSUB380:IFX$=""THEN500ELSEC
LS:GOTO260
510 CLEAR100,256*PEEK(116)+255:E
ND
520 CLS:OPEN"I",#1,"B":INPUT#1,A
$,E$,Z:CLOSE#1:LOADM"C"
530 POKES,255:C=USR2(0):GOTO210
540 CLS:GOSUB570
550 GOSUB380:IFX$="T"THEN310ELSE
IFX$<>"D"THEN550
560 CLS:OPEN"O",#1,"B":WRITE#1,A
$,E$,Z:CLOSE#1:SAVEM"C",&H3B00,&
H3DFF,0:GOTO510
570 PRINT@294,"TAPE OR DISK? (T
/D)":RETURN

```

## ANNUAL SALE May only!!

Othello	\$8.00
Say the Word	\$20.00
CoCoOz, 1985 backwards	\$6.00ea
MiCoOz	\$5.00ea

## SAVE

Australian Rainbow back issues	\$1.00ea
MiCo & GoCo back issues	\$0.50ea
Disks box	\$25.00ea
Blank Tapes	\$1.00ea
Byte (only 5 left!)	\$2.50ea
MiCo Help	\$4.00ea
CoCo Help	\$5.00ea
The Probe	\$35.00
disk or tape	
Video Amp with sound	\$30.00ea

ONLY available from  
GOLDSOFT  
P.O. Box 1742  
Southport. Qld. 4215.

# Firing Up BASIC09

By Richard A. White

Last month we prepared a disk with selected portions of OS-9, BASIC09 and RUNB. The intent was to be able to back up this disk to provide working disks for each major programming effort, and one or two disks that can be a collection of small projects. Now I will assume you have done this and, disk in hand, are ready to get on with it. And cheers to you who got on with it on your own. Put the disk in Drive 0 and boot up.

For reasons that will be apparent later, I suggest you enter both the date and the time at the prompts — did you put SETIME in your start-up file? Next you will see the BASIC09 copyright notice, BASIC09 on the left margin and READY below it. You are in BASIC09's System Mode. There are three modes, System, Edit and Debug. Commands available from System let you attend to housekeeping chores like loading, saving and other file handling activities as well as issue commands to OS-9.

Commands can be sent to OS-9 by typing the dollar sign (\$) and following it with your Shell command. For example, SMFREE will report the amount of free memory. By prefacing your Shell commands with the dollar sign you can do anything you might normally do from OS-9 provided there is enough memory available. The rub is that there isn't much free memory when BASIC09 is resident.

You have some control of memory usage from the System Mode. Enter MEM at the prompt and BASIC09 reports the amount of memory available in its buffer. This is the original buffer size you requested, less the amount taken by any resident procedures. Now, say you want to list a file but there is not enough free memory to load and execute LIST. You typed \$LIST MY-FILE, the disk drive ran, the Error #207 message appeared, and the BASIC09 and READY prompts reappeared. If you set your BASIC09 buffer size at 14K in your start-up file, you can now reduce the buffer from BASIC09's System Mode by typing MEM 12000. This reduces your

buffer size approximately 2,000 bytes, which should give LIST some elbow room. MEM 14000 will return those 2K bytes if you need them later.

Limited memory forces us to keep our program procedure modules small. This is good since it forces the creation of more maintainable programs. Only a piece of a program is active at any one time. BASIC09 lets us develop these pieces separately and keep them on the disk separately. They are loaded as needed and can be removed from memory when not needed to make room for others. This is even better than PASCAL where all procedures generally are kept in the program before the code that calls them. This also makes it easier to load and edit some module from another program that does nearly what we want rather than starting from scratch.

To reinforce the memory lesson, if you don't specify buffer memory when calling BASIC09, only a 4K buffer is allocated. Both your program and its data must fit into the buffer. But, you can request added memory when you first call BASIC09, or you can use MEM to enlarge the buffer from System Mode of BASIC09.

From System Mode, the DIR command displays the name, size and variable storage requirements of each procedure in the buffer or work space.

What the manual does not say is that pressing the ENTER key alone does the same thing. Since we have not yet started writing a procedure, the work space should be empty and DIR or ENTER should return the following on the screen.

```
Basic09
Ready
B:
  Name      Proc-Size  Data-Size
13055 free
Ready
B:
```

I had asked for 14K of buffer so you can see that BASIC09 grabs nearly 1K for its own operations. I have been working on a notepad program. The main module is called "notes." This module and others in the program run two utility modules called "printat" and "isupper," which are loaded with "notes." Now when I press ENTER, I get this display.

```
Ready
B:
  Name      Proc-Size  Data-Size
  notes     1134      5038
  printat   82         22
  *isupper  317        54
```

11537 free

```
Ready
B:
```

The free memory has been reduced by the amount of memory used for the procedures. The data sizes are reported, but data space has not been allocated at this point. However, there is obviously plenty of space left for the data. The asterisk points to the last active procedure. In this case, "isupper" was the last procedure loaded, so it was active last. If I request only a 4K byte buffer, there will be space for the procedures, but not for the data and BASIC09 will display the following.

```
MEM 4000
Ready
B:
  Name      Proc-Size  Data-Size
  notes     1134      5038?
  printat   82         22
  *isupper  317        54
```

1297 free

```
Ready
B:
```

BASIC09 knows that there is not enough memory to run "notes" and flags the fact with a question mark following the data size.

With BASIC09, you will get to know and love the friendly asterisk. Think of it as meaning all or all the way, depending on the context in which it is used. It is particularly useful when saving, killing and packing programs with a number of modules in the work space. If I type SAVE NOTES the procedure "notes" will be saved to a file on disk named "notes." If I type SAVE\* NOTES, the procedures "notes," "printat" and "isupper" will all be saved to one file on disk named "notes." The next time I load notes, all three procedures will be loaded.

Procedures are written and edited in the Edit Mode. Type EDIT, or E, and the procedure name and you will be in Edit. Lowercase works as well, and I usually stay in lowercase.

Color BASIC gives you the capability to start printing anywhere on the screen with PRINT@. BASIC09 lacks PRINT@, but has a way to position the cursor at a particular column and row. It's a tad awkward, so I put the code to do this in a short utility named "printat." For tutorial purposes it is an excellent first program, for it is both simple and will be continually usable as you program in BASIC09. It is called with this line in a BASIC09 procedure.

```
RUN printat(col,row)
```

To write "printat," we enter the Edit Mode and proceed to type it in. Here is what your screen will show. The E: is the Edit Mode prompt.

```
Ready
E: printat
PROCEDURE printat
*
E:
```

Edit Mode is a line-oriented text editor. Compared to a screen-oriented editor like *TSEDIT*, *Telewriter* or *DynaStar*, line editors leave much to be desired. Still, BASIC09's editor has certain advantages that make it the preferable editor for entering BASIC09 programs. First, it checks the syntax of each line as it's entered. Next, when you leave the Edit Mode, a check is made for other types of programming errors, for example, incomplete control structures such as FOR without a NEXT. Finally, it is able to deal with line numbers or do without them. There are 19 editor commands. I will discuss only a few.

A line of text is preceded by a space. This is perhaps the hardest thing to remember since it is different from most other word processors you may be more familiar with. The editor tries to inter-

pret a non-space character immediately after the E: prompt as a command character. An asterisk immediately following a command character means all or go all the way. If the plus sign (+) means move forward one line, +10 means move forward 10 lines and +\* means go to the end of the program. An ENTER alone moves you forward one line. The minus sign (-) moves you backward in the program, -10 means go back 10 lines and -\* means go back to the beginning.

There are commands: to change, c; delete, d; list, l; renumber, r; and s (search for a string). These apply to the current line or the next occurrence of a string in a change or search command, except when followed by the asterisk

```
PROCEDURE printat
0000  PARAM col,row:INTEGER
000B  PRINT CHR$(2); CHR$(col+32); CHR$(row+32);
0021  END
```

meaning all. Of these commands, only LIST can be followed by a number meaning the number of lines forward to be listed.

The change command is a single-line, text substitution editor. You follow the 'c' with a delimiter character such as a slash, comma or period. You choose the delimiter so it is different from any character in the original text string or the substituting text string. Next comes the original text string, another delimiter character (it must be the same character as the first delimiter) and the text string to be substituted. A final delimiter is optional. You cannot change a line number with the 'c' command. That is what 'r' is for.

The most important command is 'q', which allows you to quit editing and return to System Mode. But be warned: quitting is not always graceful. Here is where errors that are non-syntax in origin are reported, sometimes at great length. Don't worry if some get lost off the screen. From System Mode you can list your procedure to the printer and all the errors are printed at the end of the listing. Just another service from friendly BASIC09.

The operation of all these commands is well-covered in the BASIC09 manual. Read it and practice.

Following is the entire "printat" procedure.

The numbers in the left column are the number of bytes from the start of the procedure file to the start of each line in Hex. If you enter your program in lowercase, BASIC09 will change all BASIC09 keywords to capitals and leave variables and procedure names in low-

ercase. BASIC09 does some other text formatting to improve readability, including automatic indenting and removal of unnecessary spaces and parentheses.

A PARAM statement is a special type of dimension statement that defines variables to which values will be passed by the calling procedure. In this case, integers for the column and row position of the cursor will be sent. The ability to pass various types of data to and from a procedure makes modular programming possible.

In BASIC09, if a simple string or numeric variable is used without being declared in some way, the variable will be automatically dimensioned with a

default size. A string will be set to 32 characters, while a number will be real (floating point decimal). These defaults will seldom be the ideal. Integer and byte variables take less memory, run faster and can be used in most cases. A string may be as short as one character to get a 'y' or 'n' response, or can be thousands of bytes long to form a buffer into which characters are poked. Always declare your variables with DIM or PARAM.

OS-9 makes available a number of control codes to manage the alpha and graphics screens. These codes are summarized on Page 131 of your red OS-9 commands manual. BASIC09 uses them by printing them to the standard output. OS-9 intercepts them and routes them for action. The code "02" initiates the position alpha cursor operation, which is followed by two values, one for the column across the screen plus 32 and the other for the row down plus 32. The screen starts at Column 0, Row 0. An ending semicolon (;) holds the cursor at the selected location and END sends control back to the calling procedure.

The procedure "isupper" is designed to convert lowercase letters in a string to uppercase. One use is to convert menu and prompt response characters so you need only test for uppercase characters.

```
PROCEDURE isupper
0000  DIM count,line_length:INTEGER
000B  PARAM answer:STRING[25]
0017  DIM ascii:INTEGER
001E  DIM char:STRING[1]
002A  DIM work_string:STRING[25]
0036
0037  count=-1
003E  line_length=-LEN(answer)
0047  work_string=""
```

```

994E
994F WHILE count<line_length+1 DO
995F   ascii:=ASC(MID$(answer,count,1))
996E   IF ascii<96 THEN
997A     char:=CHR$(ascii)
9983     work_string:=work_string+char
998F     count:=count+1
999A   ELSE
999E     char:=CHR$(ascii-32)
99AA     work_string:=work_string+char
99B6     count:=count+1
99C1   ENDIF
99C3 ENDWHILE
99C7
99CB answer:=work_string
99D9 END

```

All "isupper" variables are DIMensioned at the start of the procedure. The contents of the string variable "answer" will be supplied by the calling procedure so it is dimensioned using the keyword PARAM. All other variables are local to "isupper" and are dimensioned using the keyword DIM. A number of variables of the same type may be included in a single DIM statement as long as they are separated by commas. Variables "count" and "line\_length" are in the same statement and "ascii" could have been added as well. String variables "char" and "work\_string" require different dimensioning statements since their lengths are different.

Variables declared with DIM are local to the procedure where they are declared. This means you can use the same variable name in another procedure to mean something entirely different. Compare this to Color BASIC where variables are global and have a single meaning anywhere in the program.

You can write an assignment statement four ways. The forms "LET count=1", "LET count:=1", "count=1" and "count:=1" will all work. The last, "count:=1," models PASCAL syntax and is preferred.

After the variables are declared they must be initialized. Variable declaration sets aside memory space for the variable, but does not change what is in that memory space, which could be anything. This also is different from Color BASIC where all numeric variables are initialized to zero and all strings are set to null when a program is first run. So, the statement "work\_string:="" is vital if "isupper" is to function.

FOR...TO...NEXT is the only loop control statement in Color BASIC. BASIC09 provides four plus a special form of IF/THEN (EXITIF...THE N...ELSE...ENDEXIT) to escape from a loop. I choose to use the WHILE...DO...ENDWHILE in "isupper" though I could have used any of the looping control structures. WHILE...DO makes a test at the very beginning and does the code in the loop only if the test proves "true." If "isupper" is sent a null string, and

line\_length=0, the program jumps over the WHILE...DO and returns "answer" unchanged to the calling procedure. This avoids an error in the statement ascii:=ASC(MID\$(answer ,count,1)).

The contents of "ascii" will be the ASCII value of a character from the "answer" string. If this value is less than 96, the character must be a non-lowercase character and can be added directly to "work\_string." If the character is lowercase, 32 is subtracted from its ASCII value yielding the ASCII value of its uppercase equivalent. The character having this value is found and added to "work\_string."

Each control structure has a unique END word. This is because these structures may span many lines of code and BASIC09 has no other way of knowing what belongs with the control structure and what does not. Color BASIC is no different. Each FOR must have a NEXT, which may be many lines down. IF...THEN...ELSE must be on a single line so the next line number gives Color BASIC the structure termination information it needs. In "isupper" ENDIF terminates the IF...THEN...ELSE structure. The following ENDWHILE does the same for WHILE...DO, sending the program to the test in the WHILE...DO. If the test is false, control goes to the line following ENDWHILE.

This line, answer:=work\_string, assigns an all uppercase string to "answer," which returns it to the calling procedure. The END is optional, but is good programming practice making the procedure more readable.

You can run "printat" and "isupper" from BASIC09's System Mode, but you will not fully see what they do. A better way is to write a short procedure that uses each and demonstrates what they do, as the following "demo" procedure does.

```

PROCEDURE demo
9999 DIM temp$:STRING[1]
999C DIM answer:STRING[25]
9918 REPEAT
991A
991B PRINT CHR$(12)
9929 PRINT
9922 PRINT "Enter string."
9933 PRINT
9935 GET #9,answer
993E RUN isupper(answer)
9948 RUN printat(2,19)
9953 PRINT answer
9958 RUN printat(2,12)
9963 PRINT "Enter another? y/n"
9979 RUN printat(2,14)
9984 GET #9,temp$
998D
998E RUN isupper(temp$)
9998 UNTIL temp$="N"
99A4 END

```

The procedure "demo" uses only two variables that are both dimensioned with DIM statements. The variable "temp" is one character long and is used to get single key responses from the keyboard. String variable "answer" can take up to 25 characters from the keyboard.

A REPEAT...UNTIL loop makes its test at the end of the loop, so its code will always be executed at least once. It is an excellent control structure where you want to repeat until the user indicates "quit" with a particular keystroke.

Printing CHR\$(12) clears the screen. This is the same as CLS in Color BASIC.

"GET #0,answer" gets characters from the keyboard, path #0, and puts them into the variable "answer" until its 25-character limit is reached or until it sees a carriage return (ENTER).

The procedure "isupper" is run with the string "answer" supplied as a parameter. It converts the string to all uppercase and returns the string in the variable "answer," which is printed to prove the fact to you. Notice the use of "printat" to position the cursor on the screen. Actual numbers are used as parameters, but these could have been integer-type variables.

Finally, the user is asked if he wishes to enter another string. Since a single character string response is needed, the one character string variable "temp\$" is used in the GET statement, eliminating the need for the user to type both the character and an ENTER. The statement "RUN isupper(temp\$)" does any needed case conversion. If temp\$="N", the test after UNTIL is true and the procedure ends. Note that when testing for equality, temp\$="N", only the equal sign is used, the colon-equal symbol (:=) is reserved for assignment statements only.

We have covered a little about a lot of things in this article. We have seen part of the operation of the System and Edit modes. I have also given you two useful procedures and a program to demonstrate them to get you some hands-on programming. I have passed right by some things I don't use as much, if at all. We will pick some of these up in later columns.

Our last bit of business is how to depart BASIC09 from the System Mode. Type bye and press the ENTER key. ☺



# 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

\$25.00

With instructions and sound

\$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.

With amplifier

\$49.95

\$59.95

### 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.)

Australian CoCo Magazine. (Aug '84 to now.)

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)

Feb '85 through  
through Jan '85  
each

\$4.50

\$2.50

\$3.45

each

\$1.00

each

\$2.00

each

\$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!

Each Tape

\$9.50

Subscription, 6 months

\$42.00

12 months

\$75.00

Each DISK

\$10.95

Subscription on disk, 12 months

\$102.50

Back issues of CoCoOz are always available

**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

\$15.00

Subscription, 12 months

\$144.00

**NEW for 1986 ONLY** Each DISK

\$15.00

Subscription on disk, 12 months

\$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

\$9.50

Subscription, 12 months

\$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

\$39.95

**Bric a Brac:**

Blank tapes . . . 12 for \$18.00 or \$1.70 each.

Cassette cases . . . 15 for \$5.00.

Disks . . . (they work!) \$3.50 each or \$28.95 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, Minerva 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:**

**What's on:**

**Best of CoCoOz #1. EDUCATION**

ROADQUIZ ..... ROB WEBB  
 HANGMAN ..... ALEPH DELTA  
 AUSTGEOG ..... P. THOMAS  
 SPELL ..... IAN LOBLEY  
 FRACTUT ..... ROBBIE DALZELL  
 ICOSA ..... BOB WALTERS  
 TAXMAN ..... TONY PARFITT

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

TREASURE ..... DAVISON & GANS  
 MASTERMIND ..... GRAHAM JORDAN  
 ANESTHESIA ..... MIKE MARTYN  
 OREGON TRAIL ..... DEAN HODGSON  
 ADVENTURE ..... STUART RAYNER

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

LE-PAS ..... Wrongsoft  
 COCOMIND ..... STEVE COLEMAN  
 OILSLICK ..... JEREMY GANS  
 CCMETEOR ..... BOB THOMSON  
 BATTACK ..... JEREMY GANS  
 PROBDICE ..... BOB DELBOURGO  
 CHECKERS ..... J & J GANS

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

PAGER ..... ?  
 HI ..... ALEX. HARTMANN  
 SPOOL64K ..... WARREN WARNE  
 CREATITL ..... BRIAN FERGUSON  
 FASTEXT ..... OZ-WIZ  
 DATAGEN ..... ROBIN BROWN  
 SPEEDCTR ..... PAUL HUMPHREYS  
 PRNTRSORT ..... PAUL HUMPHREYS  
 BIGREMS ..... BOB T  
 DIR ..... PAUL HUMPHREYS

**Best of CoCoOz #4. BUSINESS**

HI ..... ALEX. HARTMANN  
 (Disk Directory manager)  
 BANKSTAT ..... BARRY HATTAM  
 (Statement annal & store)  
 INSURE ..... ROY VANDERSTEEN  
 (Analyse home contents)  
 SPOOL64K ..... WARREN WARNE  
 (Printer spooler req 64K)  
 2BC ..... WARREN WARNE  
 (Hold 2 sep progs in mem)  
 DATABASE ..... PAUL HUMPHREYS  
 (THE tape database)  
 RESTACC ..... DUNG LY  
 (Tape restruant accounts)  
 PRSPDSHT ..... GRAHAM MORPHETT  
 (Disk print out SPDSHEET)

MARKET ..... ALEPH DELTA  
 TOWNQUIZ ..... ROB WEBB  
 ALFABETA ..... RON WEBB  
 TANK ADDITION ..... DEAN HODGSON  
 TABLES ..... BARRIE GERRAND  
 KIDSTUFF ..... JOHANNA VAGG  
 FLAGQUIZ ..... ROB WEBB

SHOOTING GALLERY ..... TOM DYKEMA  
 GARDEN ..... DAVE BLUHDORN  
 YAHTZEE ..... KEVIN GOWAN  
 BATTLESHIP ..... CHRIS SIMPSON  
 ANDROMIDA ..... MAX BETTRIDGE

PYTHON ..... ?  
 POKERMCH ..... GRAHAM & MATTHEWS  
 SPEEDMATH ..... DEAN HODGSON  
 LNDATTCK ..... ALDO DEBERNARDIS  
 INVADERS ..... DEAN HODGSON  
 RALLY ..... TONY PARFITT  
 FOURDRAW ..... JOHANNA VAGG

COPYDIR ..... THOMAS SZULCHA  
 LABELLER ..... J.D. RAY  
 SCRPRT ..... TOM DYKEMA  
 MONITOR+ ..... BRIAN FERGUSON  
 BEAUTY ..... BOB T  
 PCOPY ..... B. DOUGAN  
 RAMTEST ..... TOM DYKLEMA  
 DISKFILE ..... B. DOUGAN  
 LABEL ..... F. BISSELING

PERSMAN ..... PAUL HUMPHREYS  
 (Personal finance management)  
 CC5 ..... GRAHAM MORPHETT  
 (Sales Invcicing-tape sys)  
 COCOFILE ..... BRIAN DOUGAN  
 (Tape data base)  
 DPMS ..... PAUL HUMPHREYS  
 (Disk Program Management Sys)  
 4OKGREY ..... RAY GAUVREAU  
 (40K Basic for grey 64K CoCo)  
 TAXATION ..... ?  
 (Calc tax payable)  
 SPDSHEET ..... GRAHAM MORPHETT  
 (Disk 22 colour spreadsheet)  
 ACS3 ..... GREG WILSON  
 (Multi disk data base)

Tape \$10.00  
 Disk \$21.95

Tape \$10.00  
 Disk \$21.95

Tape \$10.00  
 Disk \$21.95

Tape \$10.00  
 Disk \$16.00

Tape \$10.00  
 Disk \$21.95

**Specials**  
 Any two tapes \$17.00  
 Any 3 tapes \$20.00  
 Two Disks or more each \$16.00

**Next:**

Best of CoCoOz #5. Adventure Games  
 Best of CoCoOz #6. Preschool Education Due: June '86

**SUBSCRIPTION FORM**

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

**AUSTRALIAN RAINBOW**

6 months \$24.75  
 12 months \$39.95

**AUSTRALIAN CoCo/Softgold**

6 months \$19.00  
 12 months \$31.00

**RAINBOW ON**

6 months  TAPE \$ 81.00  DISK \$ 81.00  
 12 months \$144.00 \$172.00  
 deb monthly \$ 15.00 \$ 15.00

**CoCoOz ON**

6 months  TAPE \$ 42.00  DISK \$ 58.00  
 12 months \$ 75.00 \$102.50  
 deb monthly \$ 9.50 \$ 10.95

**MicoOz**

6 months  TAPE \$ 42.00  
 12 months \$ 75.00

**Additional Requirements:**

.....  
 .....

Sub No:

Name:

Address:

P.C.

Phone No.:

Credit Card No.:

Bankcard  Visa  Mastercard

Authorised AMT \$ .....

**OFFICE USE ONLY:**

Received:-

## MK 1 SERIAL/PARALLEL PRINTER INTERFACE

CONNECT CO-CO I or II to a PARALLEL PRINTER  
Revised MK1 PRINTER INTERFACE

**Features:**

- \* EXTRA SERIAL PORT for MODEM, no more plugging /unplugging cables.
- \* Compatible with Standard Centronics Parallel Printers eg: EPSON, GEMINI, BMC, CP80, TANDY ETC.
- \* Plugs into CO-CO or CO-CO II Serial Port and includes all cables and connectors.
- \* SIX Switch selectable Baud rates 300, 600, 1200, 2400, 4800, 9600
- \* Power Pack is required for Printers not supplying power at pin 18 on the Parallel Connector eg: EPSON, BMC, CP80.
- \* Increases Printing Speed by up to 30% on TANDY DMP100/200 Printers

NOW ONLY — \$89.95 (including postage)  
Add \$9 for Power Pack if required

AVAILABLE FROM: G. & G. FIALA  
P.O. BOX 46  
THORNLEIGH, NSW. 2120  
phone: (02)-84-3172

## TANDY 1000 UPGRADES



**NEW!**

Everything for the 1000!

- COMBI/1000 - single slot memory to 640K, RS232, DMA
- Internal 10 & 20 Meg Hard Drives
- Floppy Drives
- Barcode Reader
- Supercharger!
- COMPLETE SYSTEMS

Ring for brochure/price list

**ASP**  
MICROCOMPUTERS

TELEPHONE (03) 500 0628  
P.O. BOX 259  
CAULFIELD EAST 3145

## Hardware/Software Specialists

For All Your CoCo Needs

**AUTO ANSWER \$399.00**

### INFO CENTRE

THE FIRST BULLETIN BOARD SYSTEM  
for Tandy's computers

(02) 344 9511 — 300 BPS (24 Hours)

(02) 344 9600 — 1200/75 BPS  
(After Hours Only)

### SPECIAL!

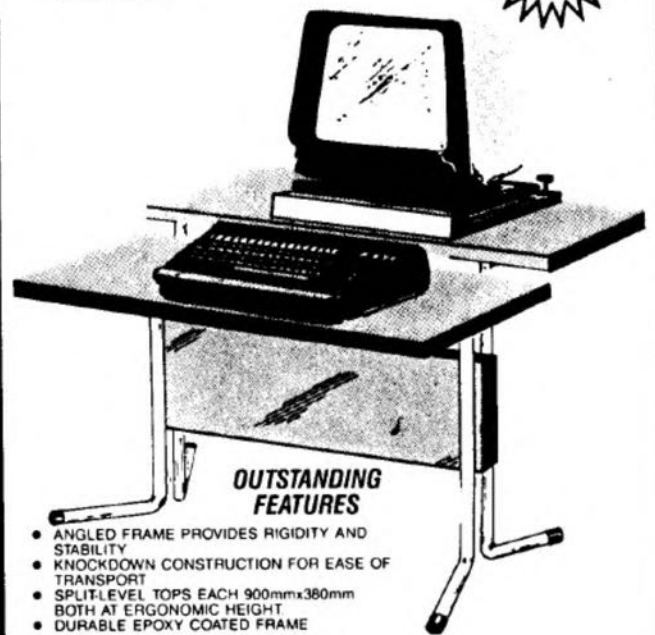
Avtek Mini Modem + Cable + CoCo  
Tex Program — the total Viatel System —  
\$279.00

We also have the largest range of Software for  
OS-9 and Flex operating systems.

## PARIS RADIO ELECTRONICS

161 Bunnerong Rd., Kingsford, N.S.W. 2032.  
(02) 344 9111

## HOME COMPUTER TABLE



### OUTSTANDING FEATURES

- ANGLED FRAME PROVIDES RIGIDITY AND STABILITY
- KNOCKDOWN CONSTRUCTION FOR EASE OF TRANSPORT
- SPLIT-LEVEL TOPS EACH 900mm x 380mm BOTH AT ERGONOMIC HEIGHT
- DURABLE EPOXY COATED FRAME
- ONLY \$125.00
- DELIVERY WITHIN BRISBANE METROPOLITAN AREA PLUS \$7.00
- DELIVERY ELSEWHERE IN QUEENSLAND PLUS \$14, IN AUST. PLUS \$20.
- MONEY BACK IF NOT SATISFIED
- TRADE ENQUIRIES WELCOMED

s&m contract furniture pty ltd



893 STANLEY ST. CNR  
HAMPTON ST. WOOLLOONGABBA  
QUEENSLAND 4102. PH: 391 8188

# User Group Contacts

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

ACT:	CANBERRA MTH	JOHN BURGERS 062 58 3924	KUDJEE	BRIAR STORE 063-72-1958	WHITEROCK	GLENN HODGES 070 54 6583	CANADA - COCO:	Richard Hobson 416 293 2346
CANBERRA STR	LES THURBORN 062 88 9226		MARBUCCA HDS	WENDY PETERSON 065 68 6723	SA:	JOHN HAIMES 08 278 3560	Ontario	
SYDNEY:			HEVCASTLE	GRAEME CLARKE 068 69 2095	ADELAIDE	STEVEN EISENBERG 08 250 6214		
BANKSTOWN	CARL STERN 02 646 3619		NOVRA	LYN DAVSON 049 49 8144	NORTH	BETTY LITTLE 08 261 4083		
BANKSTOWN WEST	ARTH PITTARD 02 72 2881		ORANGE	ROY LOPEZ 044 48 7031	GREENACRES	KEN RICHARDS 08 384 4503		
BLACKTOWN	KEITH GALLAGHER 02-627-4627		PARKES	JIM JAMES 066 62 8625	MORPETHVALE	ROB DALZELL 08 386 1647		
CARLINGFORD	ROSKO MCKAY 02 624 3353		PERRITH	DAVID SMALL 068 62 2682	PORT NARLUNGA	GLENN DAVIS 08 296 7477		
CHAITSWOOD	BILL O'DONNELL 02 419 6081		PORT MACQUARIE	ALEX SCHORFIELD 047 31 5303	SEACOMBE HTS	BILL BOARDMAN 086 82 2385		
or	MARK ROTHWELL 02 817 4627		SPRINGWOOD	RON LALOR 065 83 8223	PORT LINCOLN	KEVIN GOWAN 086 32 1368		
CLOTTON	HERMAN FREDRICKSON 02 6636379		TANWORTH	DAVID SEAMONS 047 51 2107	PORT PIRIE	MALCOLM PATRICK 086 45 7637		
HILLS DIST	DERIS CONROY 02 671 4065		TARMOOR	ROBERT VEBB 067 65 7256	WHYALLA	BOB DELBOURGO 002 25 3896		
HORSBY	ATHALIE SMART 02 846 8830		UPPER HUNTER	GARY SYLVESTER 046 81 9318	HOBART	VIM DE PUIT 002 29 4950		
KENTHURST	TOM STUART 02 654 1610		URALLA	TERRY GRAVOLLIN 065 45 1698	KINGSTON	ANDREW VILLIE 004 35 1839		
LEIGHARDT	STEVEN CHICOS 02 560 6207		VAGGA VAGGA	FRANK MUDFORD 067 78 4391	VIRVARD			
or	GORGE ECHEGARAY 02 560 9664		WYONG	CES JENKINSON 069 25 2263	VA:	IAN MACLEOD 09 448 2136		
LIVERPOOL	LEONIE DUGGAN 02-607-3791		NT:	JOHN WALLACE 043 90 0312	PERTH	TERRY BURBETT 090.21.5212		
MACQUARIE FIELDS			DARWIN	BENNTON PRIOR 089.81.7766	KALGOORLIE			
ROSEVILLE	KEN UZZELL 02 467 1619		QLD:		MELBOURNE:			
SUTHERLAND	IAN ANABELL 02 528 3391		BRISBANE:		DADDENONG	DAVID HOBROCKS 03 793 5157		
SYDNEY EAST	JACKY COCKINGS 02 344 9111		BIRKDALE	COLIN NORTH 07 824 2128	DODCASTER	JUSTIN LIPTON 03 857 5149		
ALBURY	ROB DURCAN 060 43 1031		BRASSALL	BOB UNSWORTH 07 201 8659	FRANKSTON	BOB HAYTER 03.783.9748		
ARRIVALE	DANIEL CLARSON 067 72 8031		EAST	ROB THOMPSON 07 848 5512	MARE VAREEN	LEIGH EAMES 03 704 6680		
BLAYKIND	BRUCE SULLIVAN 047 39 3903		IPSWICH	MILTON ROVE 07 281 4059	NT EASTERN	KEVIN KAZAZES 03 437 1472		
BROKEN HILL	TERRY NOOMAN 080 88 2382		PINE RIVERS	BARRY CLARKE 07 204 2806	MELTON	MARIO GERADA 03 743 1323		
CAMDER	KEVIN WINTERS 046.66.8068		SOUTH WEST	GRAHAM BUTCHER 07 376 3400	RINGWOOD	IYOR DAVIES 03 758 4496		
COFFS HARBOUR	BOB KERRY 066 51 2205		SANDGATE	MARK MICHELL 07 269 5090	SUBSBURY	JACK SMIT 03.744.1355		
COOMA	ROSS PRATT 0648 23 065		SCARBOROUGH	PETER MAY 07 203 6723	BAIRSDALE	COLIN LEHMAN 051 57 1545		
COORABONG	GEORGE SAVAGE 049 77 1054		BIGGENDER	ALAN MENHAM 071 27 1271	BALLARAT	MARK BEVELANDER 053 32 6733		
COOTLAMUNRA	CHERYL WILLIS 069 42 2264		BLACKWATER	ANNIE MELJER 079.82.6931	CHURCHHILL	GEORGE SPOWART 051 22 1389		
DEMLIQUIN	VAYNE PATTERSON 068 81 3014		BOVEN	TERRY COTTON C/O 077 86 2220	EMERLID	LEIGH EAMES 059 68 3392		
DUBBO	GRAEME CLARKE 068 89 2095		BUNDABERG	RON SIMPKIN C/O TANDY	GOUDBURN VALLEY	TONY HILLIS 058 59 2251		
FORBES	JOHANNA VAGG 068 52 2943		CAIRNS	GLENN HODGES 070 54 6583	HASTINGS	MICHAEL MORCK 059 79 2879		
GOSFORD	GARY BAILEY 065 54 5029		DALBY	ANDREW B. SIMPSON 074.62.3228	MAFFRA	MAX HUCKERBY 051 45 4315		
GRAFTON	PETER SEIFERT 043 32 7874		GOLDSTONE	CAROL CATHCART 079 78 3594	MORVELL	GEORGE FRANCIS 051 34 5175		
GUYRA	PETER LINDSAY 066 42 2503		GOLD COAST	GRAHAM MORPHEIT 075 51 0015	SALE	BRYAN McBUGH 051 44 4792		
JUNEE	MICHAEL J. HARTMAN 067 79 7547		HERVEY BAY	LESLAY HORWOOD 071 22 4989	SHEPPARTON	ROSS FARRAR 058 25 1007		
KEMPSEY	PAUL MALONEY 069 24 1860		MACKAY	LEN MALONEY 079511333x782	SMYTHESDALE	TONY PATTERSON 053 42 8815		
LITHGOW	RICK FULLER 065 62 7222		MARYBOROUGH	NORM VINN 071 41 6638	SWAN HILL	BARRIE GERRARD 050.32.2838		
LITHGORE	ROB HILLARD 066 24 3089		MORONG	MT ISA PAUL BOUCKLEY-SIMONS 077 23 6280	TRARALGON	MORRIS GRADY 051 66 1331		
MAITLAND	DAVID BERGER 063 52 2282		ROCKHAMPTON	PETER ANGEL 071 68 1628	VONTHAGGI	LOIS O'KEARA 056 72 1593		
MORBE	LYN DAVSON 049 49 8144		TARA	STEVEN YOUNGBERRY	YARRAVONGA	KEN SPONG 057 44 1488		
	ALF BATE 067 52 2465		TOOWOOMBA	GRAHAM BURGESS 076 30 4254				
			TOWNSVILLE	JOHN O'CALLAGHAN 077 73 2064				

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

\* Line Master  
 \* 10 Faces  
 \* Space Problems

In  
 Australian Coco  
 This Month:

\* Pix Save  
 \* Underworld  
 \* MiCo

POSTAGE  
 PAID  
 AUSTRALIA