Registered by Australian Post — Publication No QBG 4009

PNG K4.95 NZ $5.20

$4.50

For your TANDY Color Computer

AUSTRALIAN RAINBOW

February, 1986

No.56

FIRESTORM!
DIGITISING THE WORLD
ZOOM BOOM
FORTH
MORE BBS
OS-9
VIDEO VIPERS

And much more!

NOW AVAILABLE ON THE **QUICKSHIP PROGRAM** from your TANDY STORE
R90-9715

CoCo Max®

COLORWARE

File Edit Goodies Font Style

untitled

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

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

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

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

AMAZING!!
AMAZING!!
AMAZING!!

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

$99.95

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

SYSTEM REQUIRED:
- Any 64K COCO
- A standard Joystick, Mouse or Koala Pad

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

CFM **COMPUTERWARE FOR MICROS.**

bankcard welcome here

VISA

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

RAINBOW Info

How To Read Rainbow

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

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

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

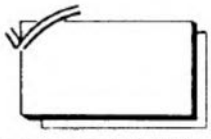
What's A CoCo

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

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

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

The Rainbow Check Plus



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

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

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

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

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

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

Using Machine Language

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

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

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

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

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

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

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

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

The Crew

Founder Greg Wilson
Publishers Graham & Annette Morphett
Managing Editor Graham Morphett
Accounts Annette Morphett
Assistand 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 procedding month.
Printed by:
Australian Rainbow Magazine
P.O. Box 1742
Southport. Qld. 4215.
Registered Publication QBG 4009.

AUSTRALIAN RAINBOW

INDEX

Reviews ..... P 5
Education:
PATTERN BLOCKS
.... by Richard Ramella P 6
CAREER CHOICE
..... by Steve Blyn P 8
How To Think
.. by Michael Plog,Ph.D P 9
ADVENTURE
..... by Fred Scerbo P 11
MATH CLASS
.... by M & J. Lamonica P 15
FIRESTORM
..... by Mike Kilby P 18
ZOOM PLANT
..... by Bill Bernico P 20
VIDEO VIPERS
..... by Robert Rice P 21
DIGITIZING THE WORLD
..... by William Barden P 23
SLEEP TIGHT Part 2
..... by Dennis Weide P 28
Beginner's Hardware
..... by Tony DiStefano P 37
SCREEN ALTERNATIVES
..... by Bill Bernico P 39
CoBBS
..... by Richard Duncan P 41
GRAPHS
..... by Bob Ddelbourgo P 51
LIFE WITH FORTH
..... by John Redmond P 53
CoCo CONNECTION
..... by Geoff Fiala P 55
OS9- Getting Started
..... by Bruce Warner P 57
New OS9
..... by Bob Rosen P 61
Corrections ..... P 62
Subscripton Page ..... P 63

lower case = article only
UPPER CASE = PROGRAM + ARTICLE

The holidays are over and we are now back into the swing of things. For Australian Rainbow Magazine, this means a return to many of the features we've all missed over the holiday period.

One change which is to be effective immediately, is the move of the Education Page from this magazine to Australian CoCo.

We are happy to make the necessary change to your subscription, if you have been buying Australian Rainbow for the Education Page. Not that Rainbow will be totally devoid of the Education input, because we will continue to have the Education input from America in these pages.

The change has been coming for some time, but was catalysed by Tandy's successful entry into the N.S.W. Education contract with the T1000.

Australian CoCo / softgold seemed to be the place for Australian educational inforation - especially now that we have to cater for the T1000 too.

We are looking forward to a very exiting year. Tandy's sales of CoCo over Christmas exceeded all but the wildest expectations. This means that there is a very large number of new users who will be seeking information.

The Computer Hut in Bowen is geared to bring you a broader range of software than ever, and specialist items which they don't handle, seem to be in ready supply from several alternative sources around the country.

The very latest software is of excellent quality, and hardly bears any resemblence to material of even a year ago. (This especially applies to the software our readers are sending - the quality of which is superb!)

And CoCo watchers this year will probably see a new CoCo on Tandy's shelves.

This CoCo has been very elusive, - Tandy watchers have been given several long blind alleys in which to wander about in!

The trouble is, that there have been 3 prototypes floating about.

Alas, it looks like those who wanted a 68000 machine might have to wait - but I'm not going to say more - I was wrong in my December prediction - I want to check my facts before I make another incorrect statement!

Ok, you oldies can now turn the page - I want to talk to all our new readers.

This magazine has a history which goes back 4 years, when a guy by the name of Greg Wilson started reprinting the American Rainbow here in a style which was unique. It won him admiration and friends around Australia and around the world.

He showed us a way to use our computers to learn - learn about computing - but often also, about ourselves.

The method is simple really - it's called sharing.

As you learn, you share what you have learnt.

I know there are lots of people around who know more than you, but there are going to be increasingly more people who know less than you, if you keep on with this thing.

And we have found that in computing, the person who is just ahead of someone else, is often the best teacher. The person who knows everything knows too much, and finds it hard to talk on a beginner's level.

How do we share? There are four ways.

The first and most important is to become a member of the local user group. I know you probably don't want to be a part of another club, but you do need contacts - people you can call when you get stuck - the more the merrier.

The second way to share is by phoning and writing to us. We don't take articles or programs over the phone, but there's a lot of conceptionalizing that does go on! All the Australian programs and articles in Australian CoCo and Australian Rainbow are supplied by people who what to share what they have learnt.

The third way to share is to buy a modem and use your computer to talk to one of the many BBS springing up all over Australia. A list of phone numbers can be found in January's Australian CoCo which was a reference issue - it had a lot of useful tid bits and lists in it.

The fourth way to share is to develop your relationship with Tandy. Get to know your local manager personally and tell him/her not only about your trials and tribulations, but also about your successes. When you get along the road a while, and you think the time appropriate, talk to Tandy's Head Office people too, folk like Ken Allen, the computer buyer and Ben Shariff, troubleshooter - these people value your comments.

I guess it all comes back to one well known cliche - "you only get back what you are prepared to put in."

In other words, in computing - if you want to learn, get involved - the more involved you are, the more you'll learn.

I should also add, the more you learn, the less you'll regret the time you spend learning!

Graham

REVIEWS

Software Review

Survive The Dangerous Drive In *Color Car Action*

Several months ago I bought an action program, *Bump and Jump*, for my Intellivision. It is a fast, fun and addicting game. Novasoft has created an excellent rendition of the *Bump and Jump* game, *Color Car Action*, for the CoCo. *Color Car Action* is easy to play. The object is to accumulate points by driving on a very dangerous road. Points are gained by staying alive and bumping cars off the road. Each type of car has a point value ranging from 200 to 500 points.

The road is divided into patterns of 20 to 60 screens. If you manage to complete a pattern without smashing any cars, 50,000 bonus points are awarded. The road is composed of over 500 screens. After completing 10 screens, there is a very short pause because the program generates 10 new screens. I did not find this pause annoying; it helped me keep track of my progress. The terrain graphics cover the four seasons, spring, summer, fall and winter.

Your car has the ability to jump cars and terrain. In order to initiate a jump the car must have a speed greater than 60 mph. The maximum speed of the car is 100 mph. If you are traveling faster than 60 mph and hit the joystick firebutton your car flies into the air. The distance the car flies is determined by your speed at the time the jump is initiated. Car speed is indicated at the top of the screen. When the car is moving fast enough to jump, a special message, JUMP OK, appears next to the speed meter. The car cannot be destroyed in the air, but watch out when you land. A diamond-shaped caution sign appears when you approach a terrain jump.

Each game is started with three cars and you are awarded an extra car for each 30,000 points accumulated. I'm not a very good wrist jockey, consequently, I did not reach 30,000 points and receive a bonus car. The five highest scores are displayed on the title screen. Names are entered using the joystick, and up to five letters are allowed.

Color Car Action comes on disk and is accompanied by two pages of instructions. The instructions are clearly written. I was operating the game in short order. Overall, the game has good graphics and color, and the sound effects are good. Just watch out for the dump trucks!

— Gabriel Weaver

Software Review

Action And Adventure With *Ghana Bwana*

Do you enjoy action games with a bit of an adventurous twist? If yes, give *Ghana Bwana* a try. You will join Professor Chance, better known as "Ghana Bwana," on his hazardous quest for the Great Secret of the legendary Erebus Island. If your luck holds out, you can make it to the treasure site.

This game is Steve Bjork's (*Zaxxon*) latest creation. It requires a 64K CoCo with Color BASIC and one disk drive. A Speech-Sound Pak can be added to make the game a little more interesting. Either the keyboard or joystick can be used for game play. I found the latter to be easier to use.

Ghana Bwana takes up one full disk. However, it is not copy protected and can be backed up with the BACKUP command. If you do a DIR, all you see is a loader program called '*'. Simply RUN"*" to start. Your screen should turn blue, for the most part, after the graphics screen appears. If it's not, press Reset until it is. Pressing the ENTER key at this point takes you to an options screen. Here the number of players, controller type and difficulty levels can be selected. When all set, just hit the joystick button or space bar to play.

There are a total of nine screens. I only got to the fourth. On the first, Professor Chance is in a small boat (outrigger). It is up to you to steer the boat and pick up pieces of a map, which are represented as small square dots. For each piece of paper collected, a small map is drawn in the upper left corner. When enough pieces are found, the map is complete and you go on to the next level. To my knowledge, the object is to pick up the map, score bonus points by collecting things such as keys and bow and arrows, and get to the treasure on the final screen. Players must also avoid the obstacles, such as enemies who fire at you, potholes, rolling rocks, sharks and waterfalls. Each screen gets progressively more difficult. To gain some hints and tips, it's a good idea to read the small manual, which is written as a cartoon.

Ghana Bwana has an appealing look, but game play can be tedious. All the menus are formatted nicely, and there's a scoreboard hall of fame that's updated and saved to disk when the game is over. I was content playing for about half an hour, but began to get disgusted when I kept having to start over. If the time runs out or you get killed at a certain screen, you must start over on that screen and collect the entire map again.

If you have a Y-cable or Multi-Pak and Radio Shack's Speech-Sound Pak, you can add voice and a few more sounds to the game. However, the speech is hard to understand. I get better quality speech with my Voice-Pak and a text-to-speech program. There is not much added sound when using the Pak, but what's there does make it more interesting and sounds neat. (It may be interesting to know that this is the first arcade game Radio Shack is selling that uses the Speech-Sound Pak.)

Ghana Bwana is a nice game, even though it can make one feel frustrated after an hour or so. But don't fret, if you're a good game player you can probably get farther and go faster than I did. Try it out at Radio Shack; it's worth the look.

— Darren Nye

A discovery process of color and shape

Pattern Blocks: Reality Play

By Richard Ramella

Tinker Toys, Legos, Lincoln Logs, Construx, Erector Sets, Capsela and ordinary blocks . . . these are some of the building toys available for children.

Except for sleep, there may be no more peaceful time in family life than when a child is quietly playing with a toy comprised of modular units. Some important learning goes on during these imagination-fed activities. The real world, with many of its physical rules intact, is emulated in miniature. The relationships of shapes are made clear in pragmatic ways. Fractions, form, planning, art and engineering become tangible for the child. Best of all, the kid is just having fun!

Pattern Blocks is a 16K Extended Color BASIC shape game even young children can play. Older youngsters and adults may enjoy using it for more complex art. The game also has possibilities for students learning geometric shapes and fractions.

At the start, 10 shapes labeled 'A' through 'J' are printed at the right of the screen. These are made of rectangles and triangles, each drawn within a square. At the left of the screen, a block cursor flickers. Using the arrow keys moves this cursor among 64 positions, eight across and eight down. Pressing a letter from 'A' to 'J' sets that shape in place. It is set without its square boundary. Moving around the grid, the player can create complex patterns and outlines by using the 10 available shapes.

The shapes are all orange at the start of play. To change their colors, press '1' for blue, '2' for green and '3' for orange. Pressing the 'A' key fills an entire block in the current color, while the 'B' key blanks the position to white.

Keys 'C' through 'J' set their shapes into place by a rule determined with GET/PUT graphics. By using the OR alternative, these shapes are set over white areas as they exist, but their blank parts don't interfere with already set shapes. To test this, run the program and press the 'E' key to make an orange rectangle in the top of the square. Now press the '1' key to change the color set to green and press the 'F' key to set the bottom of the square in green. Move the cursor off the block to see the effect. Other mirror shapes may be combined in this way, however, setting one color over another can produce unpredictable results unless some study is made of opposites and complementaries.

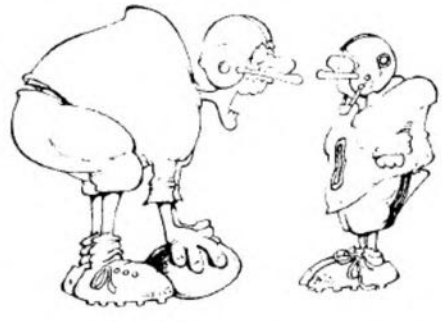
In another experiment, go to a blank area and press 'C', then 'F'. Appropriate combinations of 'C' or 'D' with 'E' or 'F' result in four different arrangements of three-on, one-off.

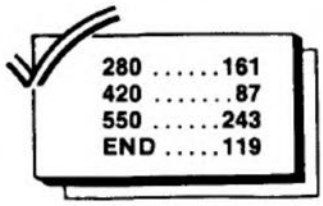
To still the flashing cursor momentarily, press ENTER. Press the space bar to restore the cursor.

The program uses POKE 65495,0 to speed up the graphics. If your machine won't work with this POKE, delete Line 130. In playing the game, always press keyboard number '0' (zero) to end a program run. This uses a POKE to bring the system back to normal speed. If you incorrectly end the program by pressing the BREAK key, you can't load and save cassette material or line print until you type POKE 65494,0 and ENTER, or turn the computer off and on again.

In a classroom setting, a child can work with the concepts of fractions and geometry using *Pattern Blocks*. A teacher may challenge the child to create a house, a whirligig, a parallelogram, a diamond, etc. For another use of the program, delete Line 140, GOTO 180. With this change, the program begins with the prompt: ACROSS 1-8, DOWN 1-8? Answer by typing whole numbers separated by a comma and in the range of 1 to 8. The result is a box drawn on the play area. The challenge may be, for example, to divide a three by three square box into mirror shapes of different colors, or equal areas in different shapes, or three equal shapes of different colors. The complexity of the problem posed depends on the child's ability.

There are various possibilities for this game as it is meant to be a discovery process. I'm confident youngsters will discover how to work *Pattern Blocks* in many different ways. As a toy, it involves the same kind of thinking as any other building toy. In my observations, children have created wildly original combinations of color and shape, intuitively learning and applying real world rules. The results are as different as the children who create them: an 8-year-old boy's robots and space ships, a 12-year-old girl's pristine geometric patterns, and the joyous chaos of a 3-year-old. Like any building toy, *Pattern Blocks'* worth is gauged by its imaginative uses.





The listing: PTRNBLKS

```
100 REM * PATTERN BLOCKS * TRS-8
0 EXTENDED COLOR BASIC 16K
110 REM * BY RICHARD RAMELLA
120 CLS: PMODE 3,1: PCLS 1: COLO
R 3,1: SCREEN 1,1
130 POKE 65495,0
140 GOTO 180
150 INPUT "ACROSS 1-8, DOWN 1-8"
;A,D: A=INT(A): D=INT(D)
160 IF A<1 OR A>8 OR D<1 OR D>8
THEN CLS: GOTO 150
170 SCREEN 1,1: LINE(0,0)-(A*24,
D*24),PSET,B
180 DIM A(2,6),B(2,6),C(2,6),D(2
,6),E(2,6),F(2,6),G(2,6),H(2,6),
I(2,6),J(2,6),K(2,6)
190 U$=CHR$(94): D$=CHR$(10): L$
=CHR$(8): R$=CHR$(9)
200 V=4: GOSUB 520
210 N=0: FOR Y=1 TO 153 STEP 38
220 FOR X=200 TO 230 STEP 30: N=
N+1
230 DRAW "BM"+STR$(X-1)+","+STR$
(Y-1)+";C"+STR$(V)+"R25D25L25U25
"
240 K=X+23: L=Y+23
250 IF N=1 THEN PAINT(X+2,Y+2),V
,V: GET(X,Y)-(K,L),A,G
260 IF N=2 THEN GET(X,Y)-(K,L),B
,G
270 IF N=3 OR N=4 THEN LINE(X+12
,Y)-(X+12,Y+24),PSET
280 IF N=3 THEN PAINT(X+20,Y+20)
,V,V: GET(X,Y)-(K,L),C,G
290 IF N=4 THEN PAINT(X+2,Y+2),V
,V: GET(X,Y)-(K,L),D,G
300 IF N=5 OR N=6 THEN LINE(X,Y+
12)-(X+24,Y+12),PSET
310 IF N=5 THEN PAINT(X+2,Y+2),V
,V: GET(X,Y)-(K,L),E,G
320 IF N=6 THEN PAINT(X+2,Y+22),
V,V: GET(X,Y)-(K,L),F,G
330 IF N=7 OR N=8 THEN LINE(X,Y)
-(X+24,Y+24),PSET
340 IF N=7 THEN PAINT(X+22,Y+2),
V,V: GET(X,Y)-(K,L),G,G
350 IF N=8 THEN PAINT(X+2,Y+22),
V,V: GET(X,Y)-(K,L),H,G
360 IF N=9 OR N=10 THEN LINE(X,Y
+24)-(X+24,Y),PSET
370 IF N=9 THEN PAINT(X+22,Y+22)
,V,V: GET(X,Y)-(K,L),I,G
380 IF N=10 THEN PAINT(X+2,Y+2),
V,V: GET(X,Y)-(K,L),J,G
390 NEXT X,Y
400 Q$=INKEY$: IF Q$=CHR$(13) TH
EN GOSUB 720
410 FOR T=1 TO 2: GET(P,Q)-(P+23
,Q+23),K,G
420 PUT(P,Q)-(P+23,Q+23),K,PRESE
T: NEXT T
430 IF Q$="" THEN 400 ELSE IF Q$
="0" THEN POKE 65494,0: END
440 GG=INSTR("ABCDEFGHIJ",Q$)
450 K=P+23: L=Q+23: IF GG>0 THEN
 ON GG GOSUB 620,630,640,650,660
,670,680,690,700,710: GOTO 400
460 V=INSTR("1234",Q$): IF V>0 T
HEN V=V+1: GOTO 210
470 IF Q$=U$ AND Q>0 THEN Q=Q-24
480 IF Q$=D$ AND Q<168 THEN Q=Q+
24
490 IF Q$=L$ AND P>0 THEN P=P-24
500 IF Q$=R$ AND P<168 THEN P=P+
24
510 GOTO 400
520 DRAW"BM208,35;C2U6E3F3D3L3R3
D3"
530 DRAW"BM239,36;U9R4F1D3G1L3R3
F1D2G1L4"
540 DRAW"BM216,67;H2L3G2D4F2R3E2
"
550 DRAW"BM239,72;U7R3F2D4G2L3"
560 DRAW"BM208,111;R7L7U4R4L4U4R
7"
570 DRAW"BM238,111;U4R4L4U4R7"
580 DRAW"BM216,143;H2L3G2D4F2R3E
3L4"
590 DRAW"BM238,149;U8D4R7U4D8"
600 DRAW"BM211,179;R3L2D8L2R4"
610 DRAW"BM238,185;D1F2R3E2U6L2R
4"
620 PUT(P,Q)-(K,L),A,PSET: RETUR
N
630 PUT(P,Q)-(K,L),B,PSET: RETUR
N
640 PUT(P,Q)-(K,L),C,OR: RETURN
650 PUT(P,Q)-(K,L),D,OR: RETURN
660 PUT(P,Q)-(K,L),E,OR: RETURN
670 PUT(P,Q)-(K,L),F,OR: RETURN
680 PUT(P,Q)-(K,L),G,OR: RETURN
690 PUT(P,Q)-(K,L),H,OR: RETURN
700 PUT(P,Q)-(K,L),I,OR: RETURN
710 PUT(P,Q)-(K,L),J,OR: RETURN
720 EXEC 44539: RETURN
730 REM * END OF LISTING
```

Preparing For The Right Career Choice

By Steve Blyn

Career education is a topic that is rapidly gaining in popularity in many schools. The choice of one's future occupation is becoming increasingly complex.

As new technology replaces older jobs, it also creates new ones. Today we see more and more job titles than we have ever seen before. Advances in robotics are a fine example of this phenomena. Robots are indeed replacing many workers. The automobile industry is especially affected by robots. The new jobs of robot technician and repairperson as well as robot "watchers" are now available for the displaced workers.

It is becoming more difficult to keep track of and inform students of the career choices facing them. New York City is the largest school system in the country. A major move to increase career awareness has been instituted this year. It is called the Regents Action Plan. It calls for a new emphasis on teaching career awareness, shops with modern trade skills, computer literacy and foreign languages.

The hope of this plan is to better prepare the one-million students in New York City for future employment. Their Board of Education is very serious about this plan. Many new foreign language teachers were recruited worldwide during the summer. More than 10 million dollars was spent on computer hardware and software last year by the New York City Board of Education, and a similar amount will be spent this year. Much of this equipment is used to give students some background and training for the jobs that will be available to them after high school graduation.

To further help the students in learning about their own interests, a career awareness survey test was given. This test is called the Harrington-O'Shea Career-Decision Making System. It is an inexpensive and easily administered test. The survey is in two parts. In part one, a large chart helps explain job definitions to the students. Several hundred typical jobs are listed and classified. Part two consists of a 120-question survey that attempts to focus in on student interests.

A self-scoring method is provided to show the student which areas to consider. Careers are broken up into six main categories: crafts, scientific, the arts, social, business and clerical jobs. A further breakdown of jobs within the student's primary interest area is then determined by the test. Specific career choices can, therefore, be suggested to each student.

This test is not only useful in making future career choices; it is primarily used by New York City to help the students initially make a wise choice on a high school. There are many specialized high schools in the city and also special programs available in the regular high schools. This test helps students make an intelligent decision about which high school programs to apply for.

This month's program deals with careers on a beginner's level. The program asks students to match jobs with their descriptions. As written, the material is on a level for younger children.

Youngsters are usually taught about careers through the study of community helpers. They often go on field trips to visit the local firehouse, police station, bank and other neighborhood places. Parents are sometimes invited to speak to the class to explain other types of jobs. As children grow older, their fund of career knowledge should expand through other contacts.

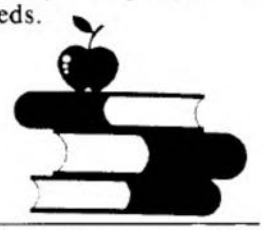
We hope you will change our data to suit your needs. For young children, it is a good idea to make several versions of the program. The extra versions can include different sets of community workers. There is, of course, no need to retype the program to make different versions. Run the original, list it, replace the data and save your new version. Older children can benefit by sets of data that describe a wider, more technical range of career choices.

The data comes in sets of 12 items. This amount was used because 12 job titles format well on the CoCo's screen. We always want our programs to format well so they are attractive and help hold the interest of the students.

Line 60 reads the 12 job descriptions (A$) and the 12 job titles (B$). Line 100 selects a random job description to be matched up by the user with the correct job title. Lines 130-210 print out these job titles. Line 220 gets the student's answer. Line 250 checks to see if the answer is a valid choice. If not, the answer is erased and another try is given. Line 240 is the escape. If 'E' is pressed as a response to any question, the program ends. This is a nicer way of exiting the program than merely pressing the BREAK key. Lines 260-310 underline the student's answer. This serves to further reinforce the student's control of and interest in the program: He sees which number he pressed as well as his answer underlined. Lines 320-330 tell the student whether he is right or wrong. If incorrect, the same question is repeated until answered correctly.

We felt there was no reason for a formal scoring in this program. It should not be looked upon as a test. It should be used until mastered. We hope you use and modify this program to your child's needs.

Listing
on Page 18.



The Learning Process: The Importance Of 'How To Think'

By Michael Plog, Ph.D.

For many years, educators have faced the problem of process versus product for student learning. I assume Plato had a similar problem when he started his school in ancient Greece.

The product of student learning is easy to see and easy to measure. A student either knows or does not know, for example, the date the Normans invaded England, the right answer to a mathematics problem, the location of a river, etc. These things are the product of learning. Most achievement tests — compiled by a teacher or commercially made — deal with the products of learning. Generally, products of learning are facts.

The process a student uses to get the correct answer, however, is an entirely different matter. Here, the question is not so much a single correct response, but what steps the student takes to get the answer. Do you remember math teachers saying they were less worried about the right answer than in how you *got* the answer? Those teachers were more concerned with the processes the student had to go through than simply knowing the right answer. In many ways, the process of learning is more important than the product.

We know it is impossible for students to learn everything that is important in any single field. There is too much factual knowledge to be retained by any one person, even experts in specialized areas. If students have the correct process of inquiry, they will be able to find knowledge when they need it, and apply it to their own life situations and problems. This skill, the ability to find the correct answer, is more important than any single correct answer we can ever teach.

As far as I know, the microcomputer is not used very much for process learning in schools. This is a shame, because these skills are much more important to people than simple factual knowledge. If you know how to find and interpret factual knowledge, you have a much better chance for success in any venture. Most microcomputer work in schools is still drill and practice — learning factual knowledge from a screen instead of a workbook. Very little work with the microcomputer is trying to get students to put together a body of facts to come to a new understanding of the material being covered.

It is somewhat ironic that process learning is more common in non-educational uses of the microcomputer. Adventure games and Simulations are usually marketed as entertainment items. In an Adventure game, the players must learn the rules as the game is played. This is a type of process learning. Adventure games involve a process the player needs to learn in order to solve the game correctly. Unfortunately, few of the processes used in Adventure games are valuable in normal school settings.

Simulations, on the other hand, are ideally suited to process learning. In a Simulation, players are given the opportunity to try different situations, see what happens, and form conclusions. When you stop and think about that, we have the same steps as the "scientific method," a process of thinking used in all fields of learning, not just the traditional science fields.

The scientific method starts with observations, which lead to a set of questions about the topic. The questions lead to a set of hypotheses. A hypothesis is not simply a question, but a question stated in such a way that it can be tested — found to be true or false. The next step is to test the hypotheses, to determine which are accurate and which are false. The final step is to draw conclusions that help explain the world around us.

The scientific method is used by scholars in all fields of learning. Sociologists, for example, consider themselves scientists of human group behavior. They follow the same steps of inquiry as the chemists, but with different considerations. The content of sociology is different from the content of chemistry; the process of inquiry and learning is the same for both fields.

A microcomputer Simulation can provide a great help to this process. Observations no longer have to be based on students actually looking at something. The computer can tell you what will happen in different situations. This can lead to faster development of hypotheses, and some Simulations are set up to actually test hypotheses. Thus, conclusions can be drawn faster.

I recently saw an interesting Simulation disguised as a science test. The student is presented with a problem about the shape of red blood cells in salt water solutions. As the amount of salt in the solution changes, the shape of the blood cell changes. The student is given the opportunity to go through a series of experiments, then asked to select the best conclusion from the information provided. Without ever using the word "hypothesis," students are expected to conduct tests on a series of hypotheses and draw conclusions about shapes of blood cells in solutions containing different amounts of salt.

In this particular example, the microcomputer keeps track of the choices made by the student — at each step of the process. The correct answer is far

less important than the steps taken by the student to get the result. This test is designed to see how students go through the process of science. Chances are that students in most schools will not have conducted this experiment, even if the school has the equipment required. Thus, the test developers are assuming the problem is unknown to the students and the process will have to be followed, instead of the students relying on past knowledge (product).

This Simulation was designed as a test of student ability to follow the process of science. Yet, the same program could be used as instruction for students instead of a test. Students in a science class could use this same program to conduct experiments that are generally not done in most science classes today.

It is my opinion that we need more educational Simulations for instruction in classrooms. Science Simulations are easy to construct because we have a large amount of information on experiments conducted by professionals. That knowledge can be reshaped into Simulation exercises for students. But science is not the only area that can benefit. Social studies Simulations, based on actual past events, can also be used to give the student a sense of the processes involved.

From my limited knowledge, the main thing that separates entertainment Simulations from educational Simulations is the content of the program. Instead of dealing with a mystery story or a futuristic army, educational Simulations deal with some topic relevant to a classroom. Also, many educational Simulations have been written by people who are knowledgeable about programming, but have limited classroom experience.

Do any of you have Simulations you use in classrooms? Have you written Simulations that are appropriate for student use? Your Color Computer is ideally suited to Simulations. If you use a disk drive, you can have a long Simulation lesson. Even with a tape recorder, the Color Computer can divide the Simulation into several parts, each following logically from the other. You are not limited to 64K with Simulations.

Students today need to learn the processes of learning, especially so to counteract the misguided emphasis on factual learning promoted by some people who do not understand deductive thought. In many ways, it is the most basic education we can provide students. (And I do hope we are all students, continuing to learn no matter what our age.) We need to encourage such use of the microcomputer, and especially encourage the development of software for this purpose.

The listing: COMMHELP

```
1Ø REM"COMMUNITY HELPERS"
2Ø REM" STEVE BLYN,COMPUTER ISLA
ND,NY
3Ø XX=RND(-TIMER)
4Ø DIM A$(12),B$(12)
5Ø FOR T=1 TO 12
6Ø READ A$(T),B$(T):NEXT T
7Ø K$=STRING$(7,195)
8Ø G$=STRING$(32,2Ø4)
9Ø H$=STRING$(32,195)
1ØØ X=RND(12)
11Ø CLS5
12Ø PRINT@ Ø,G$;
13Ø PRINT@32,A$(X)
14Ø PRINT@64,H$;
15Ø PRINT@128,STRING$(32,239);
16Ø PRINT@192,"1. PILOT  2. WAIT
ER  3. BARBER"
17Ø PRINT@256,"4. COOK   5.SALES
MAN 6. NURSE"
18Ø PRINT@32Ø,"7.FIREMAN 8.TEACH
ER  9.MECHANIC"
19Ø PRINT@384,"1Ø.JANITOR11.TYPI
ST 12. POLICE"
2ØØ PRINT@96," "
21Ø PRINT@96,"      WHICH IS MY
JOB ? ";
22Ø LINE INPUT N$
23Ø SOUND 2ØØ,2
24Ø IF N$="E" THEN CLS:END
25Ø IF VAL(N$)>Ø AND VAL(N$)<13
THEN 26Ø ELSE 2ØØ
26Ø S=VAL(N$)
27Ø R=S*11-11
28Ø IF S>3 AND S<7 THEN R=R+32
29Ø IF S>6 THEN R=R+63
3ØØ IF S>9 THEN R=R+31
31Ø PRINT@225+R,K$;
32Ø IF S=X THEN PRINT@459,"CORRE
CT !";:PLAY"L15ABCABCABC":FOR T=
 1 TO 15ØØ:NEXT T:RUN 1Ø
33Ø IF S<>X THEN PRINT@448,"PLEA
SE PRESS ENTER TO TRY AGAIN.";
34Ø EN$=INKEY$
35Ø IF EN$=CHR$(13) THEN 11Ø
36Ø GOTO 34Ø
37Ø DATA I FLY AN AIRPLANE.,PILO
T
38Ø DATA I BRING THE FOOD AT RES
TAURANTS.,WAITER
39Ø DATA I CUT AND STYLE PEOPLES
' HAIR.,BARBER
4ØØ DATA I MAKE THE FOOD AT REST
AURANTS.,COOK
41Ø DATA I SELL THINGS AT STORES
.,SALESMAN
42Ø DATA I HELP DOCTORS MAKE PEO
PLE WELL.,NURSE
43Ø DATA I PUT TO PUT OUT FIRES.
,FIREMAN
44Ø DATA I HELP STUDENTS TO LEAR
N THINGS.,TEACHER
45Ø DATA I FIX CARS AND TRUCKS.,
MECHANIC
46Ø DATA I CLEAN AND FIX-UP BIG
HOUSES.,JANITOR
47Ø DATA I WORK IN AN OFFICE.,SE
CRETARY
48Ø DATA I PROTECT PEOPLE AND PL
ACES.,POLICEMAN
```

An Educational Adventure For The CoCo And MC-10

By Fred B. Scerbo

The Wish

Many of you have suggested I create an Adventure game. I have to be honest, though — I really don't like Adventure games! There have been only one or two that I have played to completion because I simply do not have the patience an Adventure game requires. (For that matter, I have never finished a game of "Monopoly" and only rarely have played a chess game to completion, without a computer that is!) Still, I must admit I have been very impressed with some of the Adventure games. However, even though these may be fun to play, I have always hoped this time could be spent in a more educational way. Don't get me wrong. I do believe an Adventure game can have some real educational benefits in and of itself. The verbal and deductive skills required to play one can indeed be worth reinforcing.

Since much of my mail has encouraged me to create more generic educational software, the combination of these two fields can help me grant two wishes at once. Add to that the real desire from some of you to see more MC-10 versatility, and we have three wishes in one: an educational Adventure that works with just 16K Color BASIC or a 20K MC-10.

The Inspiration

Years ago, when I first started writing programs for the Color Computer, I was in the same position of many new CoCo owners. I wanted to get as much classroom use out of the old silver chassis as time and creativity would permit. Since almost no one was creating anything for our CoCo at the time, I purchased many books looking for programs I could translate to Color BASIC.

One such book that I found was *Mostly BASIC: Applications for your TRS-80*, by Howard Berenbon (Howard W. Sams & Co., Inc., Indiana, 1980). Mr. Berenbon's book contained a number of programs designed for the Model III. I was very interested in a listing called *The Dungeon of Htam*, which was listed as an educational Adventure game. It took many hours to translate the program's Level II BASIC to fit the screen limitations of the 32 by 16 CoCo screen. Finally, I got a working version debugged enough to allow my students a chance to try something different on the computer.

What was interesting about *Dungeon of Htam* (which is math spelled backwards) was its use of the typical Adventure commands such as 'N', 'S', 'E' and 'W', its "Math Monsters" named DDA, BUS, LUM and VID (add, sub, mul and div) and a map that helped you find your way around the dungeon — if you could find it. *Htam* had two levels, and certain rooms had trap doors that could be, in the words of the game, costly.

My students liked *Htam*, but it was not really that flexible for my purposes. There were no difficulty levels to control the types of problems created and, unless you found the map, the game was difficult to get around in using 'N', 'S', 'E' and 'W' as directions.

In recent years, *Htam* has not gotten much use unless a student would ask, "Do you still have that dungeon game?" Therefore, I finally decided the time was right to create my own math Adventure game using a format somewhat like that of *Htam*. This time, however, I could start from scratch using some of the programming techniques I have developed in recent years. I would create my own Adventure world and have control over what kind of math problems are presented so that even my lowest level math students could enjoy it. Keep in mind that *Math Miner* is not a rewrite of *Htam*. It is a totally new game created solely for the Color Computer and MC-10, which *Htam* never was. Still, my thanks go out to Mr. Berenbon, whose excellent book provided the inspiration for my early programming efforts.

The Program

One thing you will notice about the listing for *Math Miner* is that the program lines are shorter than usual for a "Wishing Well" program. This is for two reasons. First, MC-10 only allows a line 128 characters long as oppposed to CoCo's 255. Secondly, since MC-10 doesn't have a built-in EDIT command, I wanted to keep the lines as short as possible so any retyping of errors could be kept to a minimum.

Another difference you will notice is the absence of the ELSE command, which is missing from MC-10's IF/THEN command. The loss of ELSE makes programming a little more difficult and memory consuming, but considering that even an Apple IIe doesn't have ELSE, it is a small price to pay to make the program work on both.

My first prototype of *Math Miner* used screen POKEs for the map, but I decided to go back to using PRINT@ commands. Since the screen memory for MC-10 is different than CoCo's, I usually had to use a screen offset value (MC in the listing) for every POKE used. My first translation kept locking up because of a misplaced POKE value. I soon discovered the source of the bug and fixed it, but I again realized how easy it might be for someone typing in the program to lose it all to a typo in the POKEs. Therefore, I did a little rewriting and used the offset on a PEEK command instead, which is not program destructive as a POKE can be. (POKE in the wrong place and BASIC can be changed in ways you may not like!) Either way, the program works just fine, but you will still have to make one line change to translate to MC-10. More on that later.

Running the Program

If you are using a 16K Extended machine, PCLEAR1 before loading the game. This frees up the extra memory needed.

Math Miner takes place in a cavernous mine deep underground. You start out at the lowest level with 500 units of fuel in your backpack. This fuel is used later when you make it to the surface/roof area where a helicopter awaits to aid in your escape. You must accumulate enough fuel to fly the copter away.

Instead of using 'N', 'S', 'E' and 'W' commands, your movements can be made by using the arrow keys. The map always appears in front of you. (There is even a routine built in that prevents the map from scrolling off the screen.) Each of the five levels is an 8 by 8 grid. Your position is indicated by a set of brackets on the screen. Move the arrow keys and the brackets will move on the map.

There are two symbols on the map. "TR" stands for treasure room, where you can answer a math question and earn fuel points. "PW" stands for passageway and is your way of moving from one level to another. You can only use a passageway if a certain number of questions have been correctly answered (anywhere from five to 15).

A spirit creature also roams the empty corridors. If he approaches you, you must answer his question. However, if you have the Shield, you can ignore him by refusing to lower it. Times may occur when you may want to lower the shield just to accumulate correct answers. This is valuable if you end up on a level with too few TRs. The shield is found by correctly answering certain problems.

Upon entering a passageway, it may transport you to the next highest level. On other occasions, it may not be in working order and tell you to return later. Still other times, you might fall through a trap door to a lower level. If you have the Magic Wand, you are protected from falling through, but may find the wand vanishes from your hand. To keep track of inventory, press 'I' for an accounting of what you are carrying.

Once the fifth level has been reached, you must also locate the key to the helicopter. If all these elements are in place and there is enough fuel, you can exit the level and escape to the roof. You are then presented with your game statistics.

If it is necessary to end a game early, pressing the '@' button asks if you wish to quit. 'Y' ends the game while 'N' continues it. This prevents accidentally ending a game by hitting the wrong key.

One of the routines incorporated into this program is my own version of the INPUT command using INKEY$. The screen reacts exactly as if using INPUT while ignoring the letters on the keyboard, which should not be included in a mathematical answer. The backspace arrow erases errors and you must still press ENTER to record your answer. The main advantage of the routine is that it effectively neutralizes the CLEAR key. My students have used educational programs from other sources that use INPUT, and pressing the CLEAR key while in this mode wipes the entire screen clean, removing any work the student has done. This can be very frustrating in an educational program, so I developed this routine to bypass that problem.

Educational Uses

Math Miner lets you choose from addition, subtraction, multiplication and division, or a combination of any of these. On running the program select the numbers 1-4, which give the type of math problems. You may then choose a difficulty level (1-5) that controls how hard the problems will be. Next, choose whether you want the problem types assorted. Pressing 'Y' while on multiplication gives problems of the two categories before it (addition and subtraction), while pressing 'N' gives only multiplication. You be the judge of how difficult the problems should be. This feature lets the program grow with the student.

Also included is a variety of responses in the DATA statements at the beginning of the list. This adds a little spice to the program and helps reinforce reading skills since the student must read all the text to effectively play the game.

MC-10 Version Only

I had to make a number of changes to make this program work on the MC-10's cramped keyboard. MC-10 does not have separate arrow keys like the CoCo. Instead, to use the backspace arrow or cursor controls, the control key must be held down while hitting either 'A', 'S', 'W' or 'Z'. Since these keys are visibly marked with the arrow symbols, I made a change in the program to allow it to react to those letters without having to press the control key. This is also true when using the backspace arrow to correct an answer on a math problem. Press the 'A' key, which is normally the backspace with control, only you don't need to use the control key.

To keep the line changes to a minimum, I used variables A1, A2, A3 and A4 to stand for the ASCII values of the keys pressed. If you type in the listing as shown, the values of the arrow keys are used. However, if you wish to use the 'A', 'S', 'W' and 'Z' keys, Line 15 must be changed so it is no longer a REM with an IF/THEN statement. Therefore, replace Line 15 with:

```
15 MC=15360:A1=87:A2=90:
A3=65:A4=83
```

Use of the REM in the listing prevents these values from being used in the CoCo version. If you forget to change these values with the MC-10, you will find that the screen keeps flashing because the wrong memory location is being checked to see if the map scrolled off the screen. If this happens, make the change in Line 15.

The use of these variables at the beginning of the program ensures that only one change be made to switch machines. I think you will find this much easier than having to alter a half-dozen lines.

I have spent a good deal of time testing this out on the MC-10, but it is possible that I did not encounter every combination it generates. Therefore, if you get an ?SN Error in any line, retype it with spaces inserted between the BASIC commands. MC-10 interprets BASIC a little differently than the CoCo, so a line that works on CoCo without spaces may not always work on the MC-10 without inserting them.

All Models

Be sure to type in the DATA statements exactly as they appear. This makes sure we do not get any word-wrap problems.

Conclusion

Give *Math Miner* a try. Those with youngsters in the family will find that they will spend a good deal of time trying to increase their scores. Most importantly, however, is my hope that this program may inspire one of you to create something original much the way those early programs I experimented with did for me. □

70	103	635	251
170	51	725	69
240	97	800	172
340	233	920	34
430	221	980	89
540	52	END	46

The listing: MATHMINR

```
1 REM**************************
2 REM*   MATH MINER (C)1985   *
3 REM*    BY FRED B.SCERBO    *
4 REM*    60 HARDING AVENUE   *
5 REM* NORTH ADAMS, MA. 01247 *
6 REM**************************
10 CLS0:CLEAR400:DIMR(8,8,5),PS(
8,8):A1=94:A2=10:A3=8:A4=9
15 REM IF MC-10 THEN MC=15360:A1
=87:A2=90:A3=65:A4=83
20 READI:IFI=0THEN35
25 FORY=1TO7:SET(I,Y,1):NEXT
30 GOTO20
35 READI:IFI=0THEN50
40 SET(I,1,1):GOTO35
45 DATA1,4,7,10,14,18,22,25,30,3
3,36,39,42,48,51,57,0,2,3,5,6,11
,12,13,16,17,19,20,31,32,34,35,0
50 SET(11,4,1):SET(12,4,1):SET(1
3,4,1):SET(23,4,1):SET(24,4,1):F
ORI=1TO6:SET(42+I,I+1,1):NEXT
55 FORI=1TO7STEP3:FORY=52TO54:SE
T(Y,I,1):NEXTY,I:FORI=1TO4STEP3:
FORY=58TO61:SET(Y,1,1):NEXTY,I
60 SET(61,2,1):SET(61,3,1):SET(5
9,5,1):SET(60,6,1):SET(61,7,1)
65 PRINT@162," BY FRED B.SCERBO
(C) 1985 ";
70 PRINT@228," SELECT PROBLEM LE
VEL: ";:PRINT@294," 1) ADDITION
     ";:PRINT@326," 2) SUBTRACT
ION    ";
75 PRINT@358," 3) MULTIPLICATION
 ";:PRINT@390," 4) DIVISION
  ";
80 X$=INKEY$:HK=RND(9999):IFX$="
"THEN80
85 LL=VAL(X$):IF LL<1THEN80
90 IF LL>4THEN80
95 PRINT@452," DIFFICULTY LEVEL:
 1-6 ";
100 X$=INKEY$:IFX$=""THEN100
105 DL=VAL(X$):IF DL<1THEN100
110 IF DL>6THEN100
115 DL=DL*75
120 PRINT@452," ASSORTED LEVELS
(Y/N) ";
125 X$=INKEY$:IFX$=""THEN125
130 IFX$="Y"THEN AT=1:GOTO145
135 IFX$="N"THEN AT=0:GOTO145
140 GOTO125
145 CLS0:PRINT@263," PLEASE STAN
D BY ";
150 L=1:H=1:V=1:OH=1:OV=1:UF=500
:PA=0:SP=RND(8)+2
155 SH=0:BP=1::W=320
160 TF=4000+RND(DL)*RND(DL)
165 R$=CHR$(128):B$=R$+R$+R$+R$:
FORI=1TO40:W$=W$+B$:NEXTI
170 FORI=1TO8:FORY=0TO7:PS(Y+1,I
)=32+Y*4+Z:NEXTY:Z=Z+32:NEXTI
175 FORI=1TO5:FORY=1TO8:FORQ=1TO
8:P=RND(5)-1:IF P>2THEN P=0
180 IF P=2THEN P=RND(3)-1
185 R(Y,Q,I)=P
190 IF P=2THEN Z=1
195 NEXTQ,Y:IFZ=0THENR(8,8,I)=2
200 Z=0:NEXTI
205 R(1,1,1)=0
210 FORI=1TO10:READER$(I):NEXT
215 DATA"THE COAST IS CLEAR. YOU
 MAY MOVETO ANOTHER ROOM.","NO O
NE IS HERE. YOU MAY PROCEED TO A
NOTHER CHAMBER IF YOU DARE!"
220 DATA"SO FAR, SO GOOD! NO DAN
GER IS INSIGHT AT THIS POINT!","
YOUR LUCK IS HOLDING OUT SO FAR.
DON'T GET TOO CARELESS!"
225 DATA"YOU SEE NOTHING BUT DAM
P, BARE  WALLS AROUND YOU. KEEP
GOING."
230 DATA"THE STONE FLOOR IN FRON
T OF YOU HAS A COLD EMPTY LOOK.
GO ON."
235 DATA"THIS HALLWAY APPEARS AS
 EMPTY ASALL THE REST. CONTINUE.
"
240 DATA"YOU'RE LUCKY THAT THE T
ORCHES  ARE LIT TO GUIDE YOUR W
AY. GO !","I HEAR SOMETHING! OH
NO, IT'S  A GHOSTLIKE FIGURE!"
245 DATA"THE SPIRIT CREATURE HAS
 JUST   APPEARED BEFORE YOU!"
250 CLS0:FORI=0TO31:PRINT@I,CHR$
(96);:PRINT@I+288,CHR$(96);:NEXT
255 PRINT@11," LEVEL ";L;
260 PRINT@32,"";
265 FORI=1TO8:FORY=1TO8
270 IFR(Y,I,L)=0THENPRINTB$;
275 IFR(Y,I,L)=1THENPRINTR$CHR$(
116)CHR$(114)R$;
280 IFR(Y,I,L)=2THENPRINTR$CHR$(
112)CHR$(119)R$;
285 NEXTY,I
290 GOTO380
295 PRINT@PS(H,V)R$;:PRINT@PS(H,
V)+3,R$;
300 TN=TN+1:IFUF<=0THEN975
305 IFPEEK(1024+MC)<>0THEN250
310 X$=INKEY$:IFX$=""THEN310
315 IFX$="@"THEN955
320 IFX$=CHR$(A1)THEN V=V-1:GOTO
380
325 IFX$=CHR$(A2)THEN V=V+1:GOTO
380
330 IFX$=CHR$(A3)THEN H=H-1:GOTO
380
335 IFX$=CHR$(A4)THEN H=H+1:GOTO
380
340 IFX$="I"THEN350
345 GOTO380
350 PRINT@W,W$;:PRINT@W,"YOU HAV
E"UF"FUEL UNITS.":PRINT"YOU ARE
CARRYING:"
355 IF WA=1THENPRINT"  THE MAGIC
 WAND"
360 IF KY=1THENPRINT"  THE HELIC
OPTER KEY"
365 IF SH=1THENPRINT"  THE ENCHA
NTED SHIELD"
370 IF WA=0 AND KY=0 AND SH=0THE
NPRINT"  ONLY YOUR FUEL PACK."
375 GOTO380
380 IF H<1THEN H=1
385 IF H>8THEN H=8
390 IF V<1THEN V=1
395 IF V>8THEN V=8
400 PRINT@PS(OH,OV),R$;:PRINT@PS
(OH,OV)+3,R$;
405 PRINT@PS(H,V),CHR$(123);:PRI
NT@PS(H,V)+3,CHR$(125);:OH=H:OV=
V
410 IFR(H,V,L)=2THEN495
415 IFR(H,V,L)=1THEN730
420 ER=RND(10):PRINT@W,W$;
425 PRINT@W,ER$(ER):IF ER<9THEN3
80
430 IF SH=0THEN465
435 PRINT"YOU HAVE THE SHIELD! D
O YOU WISHTO LOWER IT (Y/N)?"
440 X$=INKEY$:IFX$="Y"THEN475
445 IFX$="N"THEN455
450 GOTO440
455 PRINT"HE CANNOT HOLD YOU! PR
OCEED ON."
460 GOTO380
465 PRINT"YOU HAVE NO SHIELD SO
YOU MUST  ANSWER HIS QUESTION. <
ENTER>."
470 GOTO480
475 PRINT@384,"YOU MUST ANSWER H
IS QUESTION.   PRESS <ENTER> TO
SEE IT."
480 X$=INKEY$:IFX$<>CHR$(13)THEN
480
485 GOSUB795:IF PA>15THEN755
490 GOTO380
495 CH=RND(3)
500 IF PA<SP THEN510
505 GOTO525
510 PRINT@W,W$;:PRINT@W,"THIS PA
SSAGEWAY WILL NOT WORK  UNTIL Y
OU HAVE SOLVED MORE OF"
```

```
515 PRINT"THE PROBLEMS ON THIS L
EVEL. KEEPMOVING ON YOUR QUEST."
520 GOTO300
525 IF CH=1 AND WA=1THEN535
530 GOTO580
535 PRINT@W,W$;:PRINT@W,"DO YOU
WISH TO USE THE MAGIC    WAND IN
 THIS PASSAGEWAY (Y/N)?"
540 X$=INKEY$:IFX$="Y"THEN555
545 IFX$="N"THEN580
550 GOTO540
555 CH=INT(RND(12)/4):IF CH=0THE
N CH=1
560 IF CH=1THEN570
565 GOTO585
570 PRINT@W,W$;:PRINT@W,"THE WAN
D KEEPS YOU FROM FALLING THROUGH
 A TRAP DOOR, BUT IT THENVANISHE
S FROM YOUR HAND.":WA=0
575 GOTO300
580 IF CH=1THEN L=L-1
585 IF CH=2THEN660
590 IF CH=3THEN L=L+1
595 IF L=0THEN605
600 GOTO615
605 PRINT@W,W$;:PRINT@W,"THE PAS
SAGEWAY LEADS TO NOWHERE.TRY AGA
IN LATER! KEEP MOVING ON.":L=1
610 GOTO300
615 IF L<6THEN660
620 IF KY=0THEN630
625 GOTO640
630 PRINT@W,W$;:PRINT@W,"YOU DO
NOT HAVE THE HELICOPTER  KEY SO
YOU MAY NOT GO TO THE    ROOF YE
T. ":L=5
635 GOTO300
640 IF UF)TF THEN1000
645 PRINT@W,W$;:PRINT@W,"YOU DO
NOT HAVE ENOUGH FUEL YET SO KEEP
 LOOKING FOR MORE CHANCESTO EARN
 FUEL. YOU STILL NEED   ";
650 PRINTTF-UF"UNITS TO ESCAPE."
:L=5
655 GOTO300
660 IF L<1THEN L=1
665 IF L)5THEN L=5
670 IF CH=2THEN680
675 GOTO690
680 PRINT@W,W$;:PRINT@W,"THIS PA
SSAGEWAY ISN'T WORKING ATTHIS TI
ME. TRY AGAIN LATER."
685 GOTO300
690 IF CH=1THEN700
695 GOTO710
700 PRINT@W,W$;:PRINT@W,"A TRAP
DOOR OPENS AND YOU TUMBLEBACK TO
 LEVEL ";L:FORI=1TO1000:NEXTI:SP
=RND(8)+2:PA=0
705 GOTO250
710 IF CH=3THEN720
715 GOTO300
720 PRINT@W,W$;:PRINT@W,"THIS PA
SSAGEWAY TRANSPORTS YOU  UP TO L
EVEL ";L:FORI=1TO1000:NEXT:PA=0:
SP=RND(8)+2
725 GOTO250
730 PRINT@W,W$;:PRINT@W,"YOU ARE
 IN ONE OF MANY SECRET  TREASUR
E ROOMS. YOU CAN READ A  QUESTIO
N CARVED ON THE WALL."
735 PRINT"PRESS (ENTER) TO READ
THE WALL."
740 IFINKEY$<>CHR$(13)THEN740
745 BP=RND(4)+1:GOSUB795:BP=1
750 IF YA<>RR THEN300
755 IF SH=1THEN765
760 GS=RND(10):IF GS=10THENPRINT
"YOU FIND A SHIELD ON THE FLOOR.
":SH=1
765 IF L=5 AND KY=0THEN775
770 GOTO780
775 PRINT"A KEY APPEARS IN YOUR
HAND.":KY=1
780 IF WA=1THEN300
785 GS=RND(10):IF GS=>9THENPRINT
"YOU FIND A WAND ON THE FLOOR.":
WA=1
790 GOTO300
795 PRINT@W,W$;:PRINT@W,"YOU MAY
 NOT LEAVE THIS CHAMBER  UNTIL Y
OU ANSWER THIS QUESTION. WHAT IS
 ";
800 IF AT=1THEN810
805 ON LL GOTO815,825,840,855
810 KK=RND(LL):ON KK GOTO815,825
,840,855
815 FL=RND(DL):SL=RND(DL):PRINTF
L"+"SL;:RR=FL+SL
820 GOTO865
825 FL=RND(DL):SL=RND(DL):IF SL>
FL THEN825
830 PRINTFL"-"SL;:RR=FL-SL
835 GOTO865
840 WL=INT(DL/5):FL=RND(WL):SL=R
ND(WL)
845 PRINTFL"X"SL;:RR=FL*SL
850 GOTO865
855 WL=INT(DL/4):FL=RND(WL)+1:SL
=RND(WL)+1:FL=SL*FL
860 PRINTFL"/"SL;:RR=FL/SL
865 PRINT" ?":PRINT@489,CHR$(175
);
870 Y$=""
875 X$=INKEY$:IFX$=""THEN875
880 IFX$=CHR$(13)THEN920
885 IFX$=CHR$(A3)THEN895
890 GOTO905
895 PRINT@489," ":PRINT@489,CHR$
(175);
900 GOTO870
905 IFASC(X$)<48 OR ASC(X$)>57TH
EN875
910 Y$=Y$+X$:PRINT@489,Y$CHR$(17
5);
915 GOTO875
920 YA=VAL(Y$)
925 IF YA=RR THEN935
930 GOTO945
935 PRINT@PS(H,V)+1,R$;:PRINT@PS
(H,V)+2,R$;:PRINT@W,W$;:PRINT@W,
"CORRECT! THE ANSWER IS"RR:PL=RN
D(DL)*BP+1:UF=UF+PL
940 PRINT"YOU GAINED"PL"MORE FUE
L UNITS.":PRINT"YOU NOW HAVE"UF"
FUEL UNITS.":PA=PA+1:R(H,V,L)=0:
CR=CR+1:RETURN
945 PRINT@W,W$;:PRINT@W,"SORRY!
THE ANSWER IS"RR:PL=RND(DL*2)+1:
UF=UF-PL:PRINT"YOU HAVE LOST"PL"
FUEL UNITS."
950 PRINT"YOU ONLY HAVE"UF"LEFT!
":WR=WR+1:RETURN
955 PRINT@W,W$;:PRINT@W,"DO YOU
WANT TO QUIT (Y/N) ?"
960 X$=INKEY$:IFX$="Y"THEN1015
965 IFX$="N"THEN250
970 GOTO960
975 PRINT@W,W$;:PRINT@W,"SORRY Y
OU LOST ALL YOUR FUEL    AND ARE
 NOW TRAPPED IN THESE"
980 PRINT"CASTLE WALLS FOREVER.
TOO BAD!  PRESS ENTER FOR YOUR S
TATISTICS.";
985 X$=INKEY$:IFX$=""THEN1010
990 IFX$=CHR$(13)THEN1015
995 GOTO985
1000 PRINT@W,W$;:PRINT@W,"YOU MA
KE IT TO THE ROOF.YOU HAVETHE KE
Y & MORE THAN ENOUGH FUEL TO ESC
APE. CONGRATULATIONS!"
1005 PRINT"PRESS ENTER FOR YOUR
STATISTICS.";
1010 X$=INKEY$:IFX$<>CHR$(13)THE
N1010
1015 CLS:PRINT@101,"YOU USED"TN"
MOVES AND":PRINT@165,"ANSWERED"C
R"CORRECTLY"
1020 PRINT@229,"WHILE DOING"WR"W
RONG.":NQ=CR+WR:IF NQ=0THENNQ=1
1025 MS=INT(CR/NQ*100):PRINT@293
,"YOUR SCORE IS"MS"%."
1030 PRINT@357,"ANOTHER TRY (Y/N
) ?";
1035 X$=INKEY$:IFX$="Y"THEN RUN
1040 IFX$="N"THENCLS:END
1045 GOTO1035
```

CoCo MATH CLASS

by Mary and James Lamonica

My wife, a math teacher, and I have found that most of the arithmetic programs available are lacking in certain areas. We tried to create a program that would be versatile and interactive with traditional teaching methods. In *Add/Sub5*, we think we have achieved our goal.

Add/Sub5 has three levels of difficulty based on the number of digits. Addition, subtraction or a combination of both may be generated. You may also include both positive and negative numbers in the problems if desired.

All of the student's responses are done with the INKEY$ statement to simplify and speed up operation. The back-arrow key may be used to erase if the student makes a typing error. Two colored bands move across the screen after the problem is printed. This is designed to make the student think and not just enter the first answer that pops into his or her head.

When a student answers incorrectly, the problem and the incorrect response are stored in an array and may be printed out at the end.

In creating this program, we made use of subroutines to do repetitive operations. We also used the LEN function for determining the correct PRINT@ location. This was necessary because of the importance of position in our arithmetic system. To use this, we also had to make use of the STR$ function. Lines 8225 to 8255 and lines 8290 to 8310 illustrate the use of combinations of the LEN and STR$ functions.

The program was written with a 16K standard BASIC Color Computer. It will run with 16K Extended, but since it needs almost 9K of RAM, it is necessary to type PMODE0:PCLEAR1 before you CLOAD the program.

Line	Description
1000 -1500	Generates main menu
2000 -2060	Generates submenus
2100 -2230	Converts to negative numbers
3000 -3100	Checks for the correct response, keeps score and prompts for another problem or a return to the submenu
4000 -4440	Graphics subroutines
6000 -7545	Generates single digit problems
8000 -9435	Generates double digit problems
10000-11435	Generates triple digit problems
12000-12060	Stores incorrect problems and responses
13000-13100	Printout routine
14000-14020	Prompts for printout when array maximum of 50 incorrect is reached

1500	246	8905	186
3060	117	10010	215
6010	40	10287	66
6537	223	10872	27
7130	173	12000	74
8070	215	13034	96
8300	20	END	22

The listing: ADD SUB5

```
5 DIM WA$(50)
1000 CLS(3):SOUND128,2:SOUND128,
4
1050 PRINT@64,"THIS IS A PROGRAM
 OF ADDITION & SUBTRACTION PROBL
EMS WRITTEN BY JAMES & MARY JEAN
 LAMONICA, 1983";
1100 PRINT"ENTER YOUR NAME BELOW
 AND THEN  CHOOSE ONE OF THE FOL
LOWING BY  PRESSING THE NUMBER"
1150 PRINTTAB(5)"(1) SINGLE DIGI
T"
1200 PRINTTAB(5)"(2) DOUBLE DIGI
T"
1250 PRINTTAB(5)"(3) TRIPLE DIGI
T"
1255 PRINTTAB(5)"(4) PRINT OUT I
NCORRECT              RESPONSES"
1260 PRINT@448,"ENTER YOUR NAME
HERE ";
1270 INPUT SN$
1350 A$=INKEY$
1375 A=VAL(A$)
1385 IF A<1 OR A>4 THEN 1350
1400 ON A GOSUB 6000,8000,10000,
13000
1425 SC=0:P=0
1450 GOTO 1000
1500 GOTO 32767
2000 PRINT@96,"CHOOSE ONE OF THE
 FOLLOWING"
2010 PRINTTAB(10)"(1) ADDITION"
2020 PRINTTAB(10)"(2) SUBTRACTIO
N"
2030 PRINTTAB(10)"(3) MIXED"
2040 PRINTTAB(10)"(4) MAIN MENU"
2060 RETURN
2090 CLS(3)
2100 PRINT@64,"DO YOU WANT TO IN
CLUDE NEGATIVE NUMBERS IN THE PR
OBLEMS ?       (Y=YES, N=NO)"
2110 IN$=INKEY$
2120 IF IN$="Y" OR IN$="N" THEN
2130
2125 GOTO 2110
2130 RETURN
2200 RS=RND(3)
2210 IF RS=1 OR RS=3 THEN X=-1*X
2220 IF RS=2 OR RS=3 THEN Y=-1*Y
2230 RETURN
3000 IF Z1=Z THEN SC=SC+1
3020 IF Z1=Z THEN PRINT@352,"YOU
 ARE CORRECT! ";SN$ ELSE PRINT@3
52,"INCORRECT! THE ANSWER IS "Z
3025 IF Z1=Z THEN GOSUB 4200 ELS
E GOSUB 4400
3030 P=P+1
3040 PRINT"SCORE="SC" OUT OF"P
3050 PRINT"NEXT PROBLEM (N)"
3060 PRINT"SUB-MENU (M)"
3070 IF Z1<>Z THEN GOSUB 12000
3080 Z1$="":Z2$=""
3100 RETURN
```

```
4000 FOR H=0TO63
4010 SET(H,0,4)
4020 SET(H,1,4)
4030 SET(H,2,5)
4040 SET(H,3,5)
4050 SET(H,18,4)
4060 SET(H,19,4)
4070 SET(H,20,5)
4080 SET(H,21,5)
4090 NEXT H
4100 RETURN
4200 SOUND 89,6
4210 SOUND 125,6
4220 SOUND 147,6
4230 SOUND 176,12
4240 RETURN
4400 SOUND 58,6
4410 SOUND 58,6
4420 SOUND 58,6
4430 SOUND 5,12
4440 RETURN
6000 CLS(3):SOUND128,2:SOUND128,
4:PRINT@37,"SINGLE DIGIT PROBLEM
S.";
6010 GOSUB 2000
6230 SC=0:P=0
6250 A1$=INKEY$
6260 A1=VAL(A1$)
6270 IF A1=4 THEN RETURN
6280 IF A1<1 OR A1>3 THEN 6250
6290 GOSUB 2090
6300 ON A1 GOSUB 6400,7000,7500
6380 GOTO 6000
6390 RETURN
6400 CLS(3):SOUND128,2:SOUND128,
4
6410 PRINT@101,"SINGLE DIGIT ADD
ITION.";
6420 X=RND(9):Y=RND(9)
6430 IF IN$="Y" THEN GOSUB 2200
6440 Z=X+Y
6445 XX$=" +"
6460 PRINT@168,TAB(16)X
6480 PRINTTAB(14)"+ "Y
6500 PRINTTAB(14)"-----":PRINT
6505 GOSUB 4000
6510 Z2$=INKEY$
6515 IF Z2$=CHR$(8) THEN Z1$=""
6520 IF Z2$=CHR$(8) THEN 6510
6525 Z1$=Z1$+Z2$
6530 Z1=VAL(Z1$)
6534 IF Z<0 THEN PA=274-LEN(STR$
(Z))
6535 IF Z=>0 THEN PA=275-LEN(STR
$(Z))
6536 PRINT@PA,Z1$
6537 IF Z<0 AND LEN(Z1$)=LEN(STR
$(Z)) THEN 6550
6538 IF Z>=0 AND LEN(Z1$)=LEN(ST
R$(Z))-1 THEN 6550
6545 GOTO 6510
6550 GOSUB 3000
6650 M$=INKEY$
6660 IF A1=3 AND M$="N" THEN RET
URN
6665 IF M$="N" THEN GOTO 6400
6670 IF M$="M" THEN RETURN
6680 GOTO 6650
6700 RETURN
7000 CLS(3):SOUND128,2:SOUND128,
4:PRINT@100,"SINGLE DIGIT SUBTRA
CTION";
7040 Y=RND(9)
7045 X=RND(9)
7050 IF X<Y AND IN$="N" THEN GOT
O 7045
7055 IF IN$="Y" THEN GOSUB 2200
7060 Z=X-Y
7070 XX$=" -"
7080 PRINT@168,TAB(16)X
7100 PRINTTAB(14)"- "Y
7120 PRINTTAB(14)"-----":PRINT
7122 GOSUB 4000
7125 Z2$=INKEY$
7130 IF Z2$=CHR$(8) THEN Z1$=""
7135 IF Z2$=CHR$(8) THEN 7125
7140 Z1$=Z1$+Z2$:Z1=VAL(Z1$)
7142 IF Z<0 THEN PA=274-LEN(STR$
(Z)) ELSE PA=275-LEN(STR$(Z))
7144 PRINT@PA,Z1$
7145 IF Z<0 AND LEN(Z1$)=LEN(STR
$(Z)) THEN 7150
7146 IF Z>=0 AND LEN(Z1$)=LEN(ST
R$(Z))-1 THEN 7150
7147 GOTO 7125
7150 GOSUB 3000
7280 M$=INKEY$
7300 IF A1=3 AND M$="N" THEN RET
URN
7305 IF M$="N" THEN GOTO 7000
7320 IF M$="M" THEN RETURN
7340 GOTO 7280
7360 RETURN
7500 CLS(3):SOUND128,2:SOUND128,
4:PRINT@163,"SINGLE DIGIT MIXED.
";
7520 S=RND(2)
7540 ON S GOSUB 6420,7040
7542 IF M$="M" THEN RETURN
7545 GOTO 7500
8000 CLS(3):SOUND128,2:SOUND128,
4:PRINT@37,"DOUBLE DIGIT PROBLEM
S.";
8010 GOSUB 2000
8055 SC=0:P=0
8060 A2$=INKEY$
8070 A2=VAL(A2$)
8080 IF A2=4 THEN RETURN
8090 IF A2<1 OR A2>3 THEN 8060
8095 GOSUB 2090
8100 ON A2 GOSUB 8200,8800,9400
8110 GOTO 8000
8120 RETURN
8200 CLS(3):SOUND128,2:SOUND128,
4:PRINT@101,"DOUBLE DIGIT ADDITI
ON";
8220 X=RND(99):Y=RND(99)
8222 IF IN$="Y" THEN GOSUB 2200
8225 Z=X+Y
8230 XX$=" +"
8240 PA=178-LEN(STR$(X))
8242 PRINT@160,"":PRINT@PA,X
8250 PA=210-LEN(STR$(Y))
8255 PRINT@192,"":PRINT@PA-2,"+
"Y
8260 PRINTTAB(13)"-----":PRINT
8265 GOSUB 4000
8270 Z2$=INKEY$
8275 IF Z2$=CHR$(8) THEN Z1$=""
8280 IF Z2$=CHR$(8) THEN 8270
8285 Z1$=Z1$+Z2$:Z1=VAL(Z1$)
8290 IF Z<0 THEN PA=274-LEN(STR$
(Z)) ELSE PA=275-LEN(STR$(Z))
8295 PRINT@PA,Z1$
8300 IF Z<0 AND LEN(Z1$)=LEN(STR
$(Z)) THEN 8340
8310 IF Z>=0 AND LEN(Z1$)=LEN(ST
R$(Z))-1 THEN 8340
8320 GOTO 8270
8340 GOSUB 3000
8350 M$=INKEY$
8355 IF A2=3 AND M$="N" THEN RET
URN
8360 IF M$="N" THEN GOTO 8200
8370 IF M$="M" THEN RETURN
8380 GOTO 8350
8390 RETURN
8800 CLS(3):SOUND128,2:SOUND128,
4:PRINT@100,"DOUBLE DIGIT SUBTRA
CTION";
8820 Y=RND(99)
8830 X=RND(99)
8840 IF IN$="N" AND X<Y THEN8830
8845 IF IN$="Y" THEN GOSUB 2200
8850 Z=X-Y
8855 XX$=" -"
8860 PA=178-LEN(STR$(X))
8862 PRINT@160,"":PRINT@PA,X
8870 PA=210-LEN(STR$(Y))
8872 PRINT@192,"":PRINT@PA-2,"-
"Y
8880 PRINTTAB(13)"-----":PRINT
8885 GOSUB 4000
8890 Z2$=INKEY$
8895 IF Z2$=CHR$(8) THEN Z1$=""
8900 IF Z2$=CHR$(8) THEN 8890
8905 Z1$=Z1$+Z2$:Z1=VAL(Z1$)
```

```
8910 IF Z<0 THEN PA=274-LEN(STR$
(Z)) ELSE PA=275-LEN(STR$(Z))
8912 PRINT@PA,Z1$
8920 IF Z<0 AND LEN(Z1$)=LEN(STR
$(Z)) THEN 8950
8922 IF Z)=0 AND LEN(Z1$)=LEN(ST
R$(Z))-1 THEN 8950
8930 GOTO 8890
8950 GOSUB 3000
8960 M$=INKEY$
8965 IF A2=3 AND M$="N" THEN RET
URN
8970 IF M$="N" THEN 8800
8980 IF M$="M" THEN RETURN
8990 GOTO 8960
8995 RETURN
9400 CLS(3):SOUND128,2:SOUND128,
4:PRINT@103,"DOUBLE DIGIT MIXED"
;
9420 S=RND(2)
9430 ON S GOSUB 8220,8820
9432 IF M$="M" THEN RETURN
9435 GOTO 9400
10000 CLS(3):SOUND128,2:SOUND128
,4:PRINT@37,"TRIPLE DIGIT PROBLE
MS";
10010 GOSUB 2000
10055 SC=0:P=0
10060 A3$=INKEY$
10070 A3=VAL(A3$)
10080 IF A3=4 THEN RETURN
10090 IF A3<1 OR A3>3 THEN 10060
10095 GOSUB 2090
10100 ON A3 GOSUB 10200,10800,11
400
10110 GOTO 10000
10120 RETURN
10200 CLS(3):SOUND128,2:SOUND128
,4:PRINT@101,"TRIPLE DIGIT ADDIT
ION.";
10220 X=RND(999):Y=RND(999)
10225 IF X<10 OR Y<10 THEN 10220
10227 IF IN$="Y" THEN GOSUB 2200
10230 Z=X+Y
10235 XX$=" +"
10240 PA=178-LEN(STR$(X))
10245 PRINT@160,"":PRINT@PA,X
10250 PA=210-LEN(STR$(Y))
10255 PRINT@192,"":PRINT@PA-2,"+
 "Y
10260 PRINTTAB(12)"-------":PRIN
T
10265 GOSUB 4000
10270 Z2$=INKEY$
10275 IF Z2$=CHR$(8) THEN Z1$=""
10277 IF Z2$=CHR$(8) THEN 10270
10280 Z1$=Z1$+Z2$:Z1=VAL(Z1$)
10285 IF Z<0 THEN PA=274-LEN(STR
$(Z)) ELSE PA=275-LEN(STR$(Z))
10287 PRINT@PA,Z1$
10290 IF Z<0 AND LEN(Z1$)=LEN(ST
R$(Z)) THEN 10300
10295 IF Z)=0 AND LEN(Z1$)=LEN(S
TR$(Z))-1 THEN 10300
10297 GOTO 10270
10300 GOSUB 3000
10305 M$=INKEY$
10307 IF A3=3 AND M$="N" THEN RE
TURN
10310 IF M$="N" THEN GOTO 10200
10320 IF M$="M" THEN RETURN
10330 GOTO 10305
10340 RETURN
10800 CLS(3):SOUND128,2:SOUND128
,4:PRINT@99,"TRIPLE DIGIT SUBTRA
CTION";
10820 X=RND(999)
10830 Y=RND(999)
10840 IF IN$="N" AND X<Y THEN 10
830
10850 IF X<10 OR Y<10 THEN 10820
10852 IF IN$="Y" THEN GOSUB 2200
10855 Z=X-Y
10857 XX$=" -"
10860 PA=178-LEN(STR$(X))
10862 PRINT@160,"":PRINT@PA,X
10870 PA=210-LEN(STR$(Y))
10872 PRINT@192,"":PRINT@PA-2,"-
 "Y
10880 PRINTTAB(12)"-------":PRIN
T
10885 GOSUB 4000
10890 Z2$=INKEY$
10892 IF Z2$=CHR$(8) THEN Z1$=""
10895 IF Z2$=CHR$(8) THEN 10890
10897 Z1$=Z1$+Z2$:Z1=VAL(Z1$)
10900 IF Z<0 THEN PA=274-LEN(STR
$(Z)) ELSE PA=275-LEN(STR$(Z))
10905 PRINT@PA,Z1$
10910 IF Z<0 AND LEN(Z1$)=LEN(ST
R$(Z)) THEN 10920
10912 IF Z)=0 AND LEN(Z1$)=LEN(S
TR$(Z))-1 THEN 10920
10915 GOTO 10890
10920 GOSUB 3000
10930 M$=INKEY$
10935 IF A3=3 AND M$="N" THEN RE
TURN
10940 IF M$="N" THEN 10800
10950 IF M$="M" THEN RETURN
10960 GOTO 10930
10970 RETURN
11400 CLS(3):SOUND128,2:SOUND128
,4:PRINT@103,"TRIPLE DIGIT MIXED
";
11420 S=RND(2)
11430 ON S GOSUB 10220,10820
11432 IF M$="M" THEN RETURN
11435 GOTO 11400
12000 X$=STR$(X)
12010 Y$=STR$(Y)
12015 WW$=" = "
12020 YY$=X$+XX$+Y$+WW$+Z1$
12030 AN=AN+1
12040 IF AN)50 THEN 14000
12050 WA$(AN)=YY$
12060 RETURN
13000 CLS
13010 PRINT@32,"TO PRINT OUT THE
 PROBLEMS THAT  HAD INCORRECT RE
SPONSES PLEASE  BE SURE THAT PAP
ER PRINTER IS  PROPERLY CONNECT
ED AND TURNED  ON.  WHEN YOU AR
E READY, PRESS <ENTER> AND THE
PROBLEMS WITH  THE INCORRECT RE
SPONSE WILL BE  PRINTED."
13015 PRINT:PRINT"IF YOU WISH TO
 DO MORE PROBLEMS OR YOU DO NOT
WISH TO PRINT THE INCORRECT RESP
ONSES, PRESS <M>  AND THEN <ENTE
R> TO RETURN TO  MAIN MENU."
13020 INPUT RE$
13025 IF RE$="M" THEN RETURN
13030 PRINT#-2,"THESE ARE THE PR
OBLEMS THAT THE STUDENT"
13032 PRINT#-2,"ANSWERED INCORRE
CTLY. ALSO GIVEN ARE THE"
13034 PRINT#-2,"INCORRECT RESPON
SES THE STUDENT GAVE."
13036 PRINT#-2,"THESE PROBLEMS S
HOULD BE WORKED ON WITH"
13038 PRINT#-2,"THE INSTRUCTOR."
13040 PRINT#-2,"----------------
----------------------"
13042 PRINT#-2,"STUDENTS NAME  "
SN$
13044 PRINT#-2,"----------------
----------------------"
13050 FOR NA=1TOAN
13060 PRINT#-2,WA$(NA)
13070 NEXT NA
13075 IF AN=0 THEN PRINT#-2,"ALL
 THE PROBLEMS WERE ANSWERED CORR
ECTLY"
13080 AN=0:NA=0
13090 PRINT#-2,"----------------
----------------------"
13100 RETURN
14000 CLS
14010 PRINT"DUE TO THE MAXIMUM C
APACITY OF  THE STORAGE ARRAY, Y
OU MUST NOW PRINT THE INCORRECT
RESPONSES."
14020 INPUT "WHEN READY, PRESS <
ENTER)."RE$
14030 RETURN
32767 END
```

Save the kingdom of Ferra from the blazing perils . . .

Firestorm

By Mike Kilby

A terrible crisis has begun in the small kingdom of Ferra. Great arrows of fire falling from the sky have started to destroy the land. The King has called a meeting of the people to hear suggestions on how to save the kingdom. During the meeting you snicker sarcastically and say to yourself, "With a bucket of water." The King overhears you and thinks it's a wonderful idea, so he chooses you to carry out the task. The King and the kingdom are depending on you!

Your job is to carry a bucket of water and catch the falling fire arrows, which are worth points. To further complicate things, a time bomb has been planted beneath the ground upon which the fire drops. If an arrow hits this bomb the entire kingdom will be destroyed. Also, if fire reaches the ground, points are lost and a hole is left into which you must jump in order to catch the other falling menaces.

Firestorm, requiring Extended Color BASIC, is an arcade-type game involving skill and luck. The high-speed POKE is used in Line 90. For those who cannot use the high-speed POKE, simply remove POKE 65495,0 from the line. The number of men (three are given at the beginning of the game) is indicated by the lines at the top right of the graphics screen. The men are moved side to side by using the right joystick. The game begins after a title and difficulty screen are displayed.

```
240 .....76
470 .....34
670 .....140
920 .....5
1000 .....22
1310 .....194
END .....160
```

The listing: FIRESTRM

```
10 GOTO 1100
20 PMODE 4,1
30 HS=0
40 DIM C(5,5),F(5,5)
50 DRAW"S4BM8,0;DR8UDBDL8D2F2D5U
2RU5RD5RD2U5E2U"
60 GET(0,0)-(24,11),C,G
70 DRAW"BM3,28;D4F2U3D3G2UGHU2F"
80 GET(0,0)-(5,36),F,G
90 PCLS:POKE65495,0
100 FORBH=1TO30STEP7
110 FORBV=150TO190STEP14
120 LINE(BH,BV)-(BH+5,BV+5),PSET
,BF
130 NEXTBV,BH
140 FORBH=5TO30STEP7
150 FORBV=157TO190STEP14
160 LINE(BH,BV)-(BH+5,BV+5),PSET
,BF
170 NEXTBV,BH
180 FORBH=220TO255STEP7
190 FORBV=150TO190STEP14
200 LINE(BH,BV)-(BH+5,BV+5),PSET
,BF
210 NEXTBV,BH
220 FORBH=223TO250STEP7
230 FORBV=157TO190STEP14
240 LINE(BH,BV)-(BH+5,BV+5),PSET
,BF
250 NEXTBV,BH
260 LINE(35,187)-(220,190),PSET,
BF
270 LINE(26,175)-(229,186),PRESE
T,BF
280 SCREEN1,1
290 C=0:L=4:SC=0:M=3
```

```
300 IFQ=1THENN=1 ELSEN=5
310 P=40+RND(170)
320 DRAW"C8BM"+STR$(P)+",189REFR
LGH"
330 COLOR1
340 IFM=3THEN370
350 IFM=2THENLINE(245,2)-(245,6)
,PRESET:GOTO380
360 IF M=1THENLINE(250,2)-(250,6
),PRESET:GOTO390
370 LINE(245,2)-(245,6),PSET
380 LINE(250,2)-(250,6),PSET
390 K1=100:K2=175
400 FORT=1TON
410 R=35+RND(175)
420 FORF1=0TO140 STEP L
430 A=JOYSTK(0):IFA(10THENK1=K1-
8:ELSEIFA)50THENK1=K1+8
440 IFK1(35THENK1=35:ELSEIFK1+24
)218THENK1=195
450 B=PEEK(65280)
460 IF(B=126ORB=254)ANDA(10THEN
GOSUB870
470 IF(B=126ORB=254)ANDA)50THEN
GOSUB940
480 PUT(R,F1)-(R+5,F1+36),F,PSET
490 PUT(K1,K2)-(K1+24,K2+11),C,P
SET
500 IFPPOINT(K1+12,K2+12)=0THEN1
010
510 NEXTF1
520 IF(R)K1+4)AND(R(K1+16)THENSO
UND250,1:SC=SC+(10*L):ELSE 630
530 LINE(R,F1)-(R+5,F1+36-L),PRE
SET,BF
540 NEXTT
550 IFQ=1THENC=RND(6)ELSEC=C+1:N
=N+5
560 IFC=1THENL=5
570 IFC=2THENL=7
580 IFC=3THENL=10
590 IFC=4THENL=14
600 IFC=5THENL=20
610 IFC=6THENL=28
620 GOTO 400
630 SC=SC-30
640 LINE(R,F1)-(R+5,F1+36-L),PRE
SET,BF
650 DRAW"BM"+STR$(R)+",187;C8FD2
R4U2ERL2GDL3UHR4DL2"
660 PLAY"T255V31D"
670 IF R+3(P OR R+3)P+5 THEN540
680 PLAY"V31T100O1GFFEDGO3GDBEFD
O4GDFEGDFEBCEGDFEGDT150O3DFEGDFE
GDCBDCGEFDEGDT200O1BBBB8DGEFDGBB
BEGDFEDGDEGEDBBAG"
690 FORD=1TO200STEP2
700 CIRCLE(P,188),D,,.25
710 NEXTD
720 COLOR1
730 PLAY"T10V31O3CDEFGFDED"
740 FORD=1TO200:NEXTD
750 IF SC)HS THEN HS=SC ELSE 780
760 CLS:PRINT270,"ENTER INITIALE
S";:INPUTN$
770 IF LEN(N$))3THEN 760
780 CLS:PRINT2137,"SCORE:";SC
790 PRINT2232,"************"
800 PRINT2232+32,"*";N$
810 PRINT2275,"*"
820 PRINT2268,HS;
830 PRINT2232+64,"************"
840 INPUT"PLAY AGAIN";A$
850 IFA$="Y"THEN1320
860 IFA$="N"THENENDELSE840
870 LINE(K1,K2)-(K1+36,K2+11),PR
ESET,BF
880 K1=K1-4:K2=K2-11
890 PUT(K1,K2)-(K1+24,K2+11),C,P
SET
900 LINE(K1,K2)-(K1+24,K2+11),PR
ESET,BF
910 K1=K1-4:K2=K2+11
920 PUT(K1,K2)-(K1+24,K2+11),C,P
SET
930 RETURN
940 LINE(K1,K2)-(K1+24,K2+11),PR
ESET,BF
950 K1=K1+4:K2=K2-11
960 PUT(K1,K2)-(K1+24,K2+11),C,P
SET
970 LINE(K1,K2)-(K1+24,K2+11),PR
ESET,BF
980 K1=K1+4:K2=K2+11
990 PUT(K1,K2)-(K1+24,K2+11),C,P
SET
1000 RETURN
1010 LINE(K1,K2)-(K1+24,K2+11),P
RESET,BF
1020 DRAW"C1BM"+STR$(K1+8)+",180
;DR8UDBDL8D2F2D5U2RU5RD5RD2U5E2U
"
1030 PLAY"V31O1T25DDEGGFEDDEGGFE
"
1040 FORD=1TO200:NEXTD
1050 M=M-1
1060 IFM(1THEN730
1070 LINE(35,F1)-(220,191),PRESE
T,BF
1080 LINE(35,187)-(220,190),PSET
,BF
1090 GOTO 320
1100 PMODE3,1:PCLS
1110 H=85:V=40
1120 FOR T=1TO5
1130 DRAW"C4BM"+STR$(H)+","+STR$
(V)+";D20U10R5L5U10R10"
1140 DRAW"BM"+STR$(H+20)+","+STR
$(V)+";R10L5D20R5L10"
1150 DRAW"BM"+STR$(H+40)+","+STR
$(V)+";R10D10L10F10H10U10D20"
1160 DRAW"BM"+STR$(H+60)+","+STR
$(V)+";R10L10D10R5L5D10R10"
1170 H=H+1:V=V+1
1180 NEXT T
1190 H=75:V=120
1200 FOR T=1TO 5
1210 DRAW"BM"+STR$(H)+","+STR$(V
)+";R10L10D10R10D10L10"
1220 DRAW"BM"+STR$(H+20)+","+STR
$(V)+";R10L5D20"
1230 DRAW"BM"+STR$(H+40)+","+STR
$(V)+";R10D20L10U20"
1240 DRAW"BM"+STR$(H+60)+","+STR
$(V)+";R10D10L10F10H10U10D20"
1250 DRAW"BM"+STR$(H+80)+","+STR
$(V)+";D20U20F5E5D20"
1260 H=H+1:V=V+1
1270 NEXT T
1280 DRAW"S8BM120,75;D10RU10RD10
F3U6D6G2L4H2U6D5REFR2EFH6LD5EL2"
1290 SCREEN1,0
1300 PLAY"T3V31O1DDGDDFDDGDDCDDC
DDT1D"
1310 FORD=1TO1200:NEXTD
1320 CLS
1330 PRINT29,"************"
1340 PRINT241,"*fire storm*"
1350 PRINT273,"************"
1360 PRINT2136,"RANDOM  LEVELS"
1370 PRINT2201,"     OR"
1380 PRINT2263,"INCREASING LEVEL
S"
1390 PRINT2360,"INPUT(R/I)";
1400 INPUTQ$
1410 IFQ$="R"THENQ=1 ELSEIFQ$="I
"THENQ=2:ELSE1390
1420 IFA$="Y"THEN90ELSE20
```

The Zoom-Bloom Plant

By Bill Bernico

Look at a flower. How did it get there? Like every other plant, it started out as a seed. The seed popped open one day and a sprout found its way to the surface and reached skyward. The Seed visually illustrates nature's process. It shows a seed in the ground popping open to produce a sprout, and the sprout grows upward untill buds appear. Petals form around the buds and you have a flower. Ssounds simple, I know, but there are a few neat programming tricks that put the flower on the screen.

First, we DRAW and PAINT the sun (every plant needs sunlight). Next, we draw the ground - red ground, but ground nonetheless. The title is placed in the upper right corner. Keep this title, SEED, in mind as the program comes to a finish. If you look closely in the soil, you'll see a tiny seed. After a couple of seconds you'll hear and see the seed pop open and begin to grow.

With the help of a handy FOR/NEXT loop, we created the illusion of movement; that's what makes the stem grow and branch off. From here we used CIRCLE, PAINT and more sound effects to produce the petals. Once all the petals are neatly in place, watch the upper right corner again. The word SEED is replaced by the word FLOWER. After all, what started out as a seed is now a flower!

The Listing: THE SEED

```
10 'THE SEED
20 'BY BILL BERNICO
30 '708 MICHIGAN AVE.
40 'SHEBOYGAN, WI 53081
50 '(414) 459-7350
55 'CONCEPT BY KATHIE BERNICO
60 '
70 PMODE3,1:SCREEN1,0:PCLS:DRAW"
BM0,170R255":PAINT(3,179),4,4:CO
LOR2:CIRCLE(30,30),25:PAINT(30,3
0),2,2:DRAW"BM30,30M50,80M30,30M
60,60M30,30M80,50M30,30M15,110M3
0,30M110,15M30,30M110,15":COLOR
3:PSET(128,184)
80 DRAW"BM190,5L20R10D20BR10U20D
10R12U10D20BR8NR10U10NR7U10R10BM
180,40U4H4L8G4D4F4R8F4D4G4L8H4U4
BD8BR23NR10U12NR7U12R10BR7NR10D1
2NR7D12R10BR9R10E3U10H3L10D24":G
OSUB360
90 CIRCLE(128,184),4:PLAY"T25L25
5V31O1ADCFBAGEDV16ACGADV4EABCAED
B":GOSUB360
100 DRAW"BM128,184U5":GOSUB370
110 FORG=1TO50:DRAW"U":GOSUB370:
GOSUB390:NEXTG
120 FORQ=1TO6:DRAW"HLH":GOSUB370
:GOSUB390:NEXTQ:DRAW"BF18
130 FORQ=1TO6:DRAW"ERE":GOSUB370
:GOSUB390:NEXTQ:DRAW"BG18BU12
140 FORG=1TO43:DRAW"U":GOSUB370:
GOSUB390:NEXTG
150 CIRCLE(128,80),8:PAINT(128,8
0),2,3:GOSUB380:GOSUB360
160 CIRCLE(103,115),8:PAINT(103,
115),2,3:GOSUB380:GOSUB360
170 CIRCLE(155,119),8:PAINT(155,
119),2,3:GOSUB380:GOSUB360
180 CIRCLE(138,91),9:PAINT(138,9
1),4,3:GOSUB400:GOSUB360
190 CIRCLE(143,75),9:PAINT(143,7
5),4,3:GOSUB400:GOSUB360
200 CIRCLE(128,66),9:PAINT(128,6
6),4,3:GOSUB400:GOSUB360
210 CIRCLE(115,75),9:PAINT(115,7
5),4,3:GOSUB400:GOSUB360
220 CIRCLE(119,91),9:PAINT(119,9
1),4,3:GOSUB400:GOSUB360
230 CIRCLE(111,127),9:PAINT(111,
127),4,3:GOSUB400:GOSUB360
240 CIRCLE(118,112),9:PAINT(118,
112),4,3:GOSUB400:GOSUB360
250 CIRCLE(106,101),9:PAINT(106,
101),4,3:GOSUB400:GOSUB360
260 CIRCLE(90,106),9:PAINT(90,10
6),4,3:GOSUB400:GOSUB360
270 CIRCLE(91,124),9:PAINT(91,12
4),4,3:GOSUB400:GOSUB360
280 CIRCLE(139,116),9:PAINT(139,
116),4,3:GOSUB400:GOSUB360
290 CIRCLE(147,133),9:PAINT(147,
133),4,3:GOSUB400:GOSUB360
300 CIRCLE(164,131),9:PAINT(164,
131),4,3:GOSUB400:GOSUB360
310 CIRCLE(170,115),9:PAINT(170,
115),4,3:GOSUB400:GOSUB360
320 CIRCLE(155,103),9:PAINT(155,
103),4,3:GOSUB400:GOSUB360
330 CIRCLE(128,80),3:GOSUB410:GO
SUB360:CIRCLE(103,115),3:GOSUB41
0:GOSUB360:CIRCLE(155,119),3:GOS
UB410
340 DRAW"BM180,40C1U4H4L8G4D4F4R
8F4D4G4L8H4U4BD8BR23NR10U12NR7U1
2R10BR7NR10D12NR7D12R10BR9R10E3U
10H3L10D24BM160,32C3L10D8NR7D12B
R10NR10U20BR16BD4D12F4R6E4U12H4L
6G4BR20BU4D16F4E4NU6F4E4U16BR8NR
10D10NR7D10R10BR6U20R8F4D3G4L7F1
0
350 GOTO 350
360 FORX=1TO500:NEXTX:RETURN
370 FORX=1TO100:NEXTX:RETURN
380 PLAY"V30ST4L4;12T50V20;12;12
;10":RETURN
390 PLAY"T200V31O1DAC":RETURN
400 PLAY"L255T255V28O5CGCEC":RET
URN
410 SOUND240,1:RETURN
```

Keep up with the pace in this reptile race

Sidewinding Skirmishes With Video Vipers

By Robert E. Rice

Simple in concept and playable by young children (although Mom and Dad will get a kick out of it), *Snake Chase* is loosely based on the game *Colorful Maneuvers* by James Wood. Two snakes, under control of the right and left joysticks, scurry about the screen leaving ever-growing trails in their wake. The game ends when one of the snakes attempts to run over either one of the trails or the screen border. Scoring is updated and displayed by name after each round.

Only 3.6K in length, *Snake Chase* runs on the smallest of CoCos. It is fully playable on non-Extended BASIC machines, but lines 60 and 390 will have to be changed to:

```
60 IF INKEY$="" THEN 60 ELSE
FOR Q=1 TO 1500:NEXT Q

390 CLS8:A$="":V=70:C$=CHR$
(159):FORQ=1 TO 31:C1$=C1$+C$:
NEXTQ:C2$=C$+C$
```

Also, change the words EITHER FIREBUTTON in Line 720 to read ANY KEY. This allows the game to be started using a key press (followed by a three-second delay) instead of the firebutton.

The POINT function is used in lines 110 and 160 to determine the existence of a trail or border (test for "pixel on"). In this way, a player can lose when running over his own trail as well. This precludes the chance of a stalemate by not allowing players to continually back up over the top of themselves.

Despite the use of the slow SET/RESET graphics, the game tends to move along quite rapidly. Too quickly, it turned out, for my children to keep up with. Line 730 offers the option of a fast or slow speed. The speed of either mode can be adjusted by varying the value of 'S' in Line 740. 'S' is used to establish the delay duration for successive snake movements in Line 180.

Once motion has begun, the snakes continue in those directions until instructed to do otherwise. This holds true even when the joysticks are neutralized. Variables H0, H3, V0 and V3 keep track of this activity.

As you can see, the game itself is contained entirely within lines 30-330. The remaining two-thirds of the program provides frills and fluff.

Beginning at Line 380, the title screen routine uses nested (one inside the other) FOR/NEXT loops and DATA statements. This turned out to be a simple means to display large block letters on a text screen. It also alleviated the need to type CHR$ 180 times. The screen was first laid out on a standard page of PRINT@ locations copied from the CoCo manual. That information was then transposed into the respective block graphics codes and placed in DATA statements in lines 460-540. Each DATA line corresponds to one full printed line of 20 graphics blocks. The FOR/NEXT loop in Line 420 reads the data, converts them to CHR$ and consolidates them into A$. Lines 410 and 430 control the sequence of data reads and PRINTing positions for A$.

The music, lines 560-590, plays the familiar snake charmer's tune using the SOUND command. This technique greatly simplified the process of transposing the tune into corresponding SOUND values. With the CoCo manual as a reference, I wrote the note values on top of the keys of my daughter's toy organ. While pecking out the melody, I wrote down the values as I went, storing them in DATA statements. The data is arranged with tone and duration values adjacent. Since the data is read note by note with a FOR/NEXT loop (Line 560), there is sufficient delay in processing to keep identical adjacent tones from blending together. The results are every bit as good as can be produced by Extended Color BASIC's PLAY command. This method also saves a lot of typing.

Variables	Function
E	Miscellaneous
Q	FOR/NEXT loops
H1, V1, H2, V2	Direction of snake head movements
H0, H3, V0, V3	Snake head motion
H, V	Read joystick positions
S, SP	Control speed of game
C1$, C2$	Text borders
R, L	Player scores
R$, L$	Player names
A	Identifies winner
A$	Housekeeping

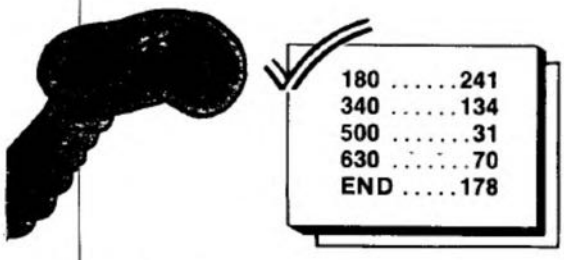
Due to the "straight through" design of the program, the only subroutine needed is the one that draws the border around text screens. Two multicharacter graphics strings (C1$ and C2$) are

formed in Line 390. The border subroutine itself resides in lines 340-370. It prints C1$ at the top and bottom of the screen. C2$ is printed along the far right border with a FOR/NEXT/STEP loop. This forces a screen wrap-around that places half of C2$ on each edge of the screen. PRINTing to the last screen position would cause a line feed for the entire screen; this is why C1$ is only 31 characters long.

The POKE in Line 360 puts the same information into that block of screen RAM without scrolling the image. By changing the STRING$ and POKE values, you can create a border using any available graphics or alphanumeric character with equal ease. Variables will also work. Try swapping them out to see what happens.

Feel free to "hack" away at this program, modifying and improving it to your own tastes. If you find some techniques here that you like, jot them down for ready reference in the future. □

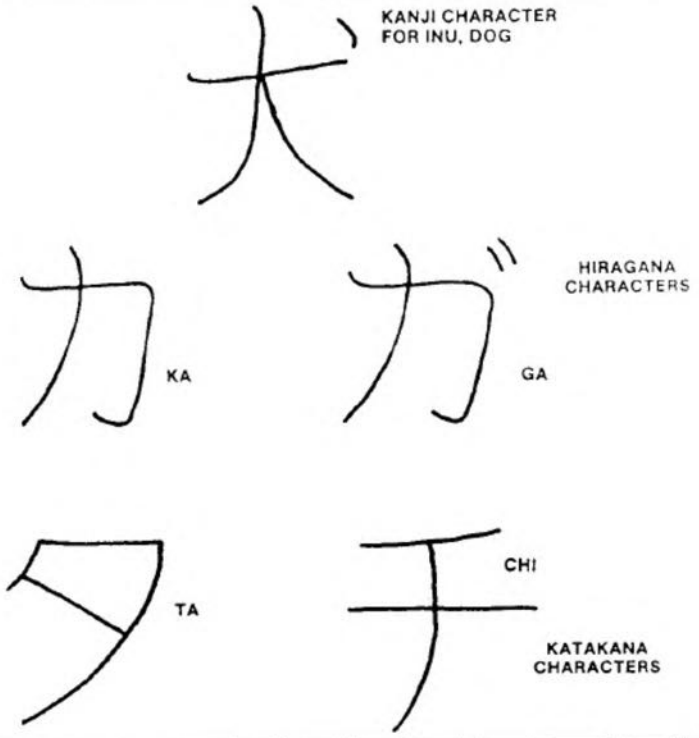
The listing: SNKCHASE




```
0 '**SNAKE CHASE/ROBERT E. RICE/
JUNE 1984
10 GOTO 380
20 '**GAME ROUTINE
30 SOUND240,5:CLS0:E=RND(8):FORQ
=0TO63:SET(Q,0,E):SET(Q,31,E):NE
XTQ
40 FOR Q=0TO31:SET(0,Q,E):SET(63
,Q,E):NEXTQ
50 H1=46:V1=3:H2=15:V2=28:SET(H1
,V1,3):SET(H2,V2,4)
60 E=PEEK(65280):IF E=255 OR E=1
27 THEN 60
70 H0=0:V0=1:H3=0:V3=-1
80 H=SGN(INT((JOYSTK(0)-5)/58)):
V=SGN(INT((JOYSTK(1)-5)/58))
90 IF H<>0 OR V<>0 THEN H0=H:V0=
V
100 H1=H1+H0:V1=V1+V0
110 IF POINT(H1,V1)<>0 THEN A=1:
GOTO 190
120 SET(H1,V1,3)
130 H=SGN(INT((JOYSTK(2)-5)/58))
:V=SGN(INT((JOYSTK(3)-5)/58))
140 IF H<>0 OR V<>0 THEN H3=H:V3
=V
150 H2=H2+H3:V2=V2+V3
160 IF POINT(H2,V2)<>0 THEN A=2:
GOTO 190
170 SET(H2,V2,4)
180 FOR SP=1 TO S:NEXT SP:GOTO 8
0
190 '**WIN/LOSE ROUTINE
200 SOUND10,10
210 IF A=2 THEN 230
220 FOR Q=1TO25:RESET(H1,V1):SOU
ND200,1:SET(H1,V1,3):SOUND240,1:
NEXT Q:GOTO 240
230 FOR Q=1TO25:RESET(H2,V2):SOU
ND200,1:SET(H2,V2,4):SOUND240,1:
NEXT Q
240 FOR Q=1TO1000:NEXT Q
250 CLS3:GOSUB 340:IF A=1 THEN P
RINT@173-LEN(L$)/2,L$" WINS!";:L
=L+1 ELSE PRINT@173-LEN(R$)/2,R$
" WINS!";:R=R+1
260 FOR Q=1TO1500:NEXT Q
270 PRINT@298,"TOTAL WINS:";:PRI
NT@365-LEN(L$)/2,L$"="L;:PRINT@4
29-LEN(R$)/2,R$"="R;
280 FOR Q=1TO1500:NEXT Q
290 PRINT@42,"PLAY AGAIN?";:SOUN
D240,1
300 A$=INKEY$:IF A$="" THEN 300
310 IF A$="N" THEN PRINT@11,"((
BYE ))";:SOUND190,3:FOR Q=1 TO 2
000:NEXT Q:END
320 IF A$<> "Y" THEN 300
330 GOTO 30
340 PRINT@0,C1$;
350 FORQ=31 TO 479 STEP32:PRINT@
Q,C2$;:NEXT Q
360 PRINT@480,C1$;:POKE1535,159
370 RETURN
380 '**TITLE SCREEN
390 CLS8:A$="":V=70:C1$=STRING$(
31,159):C2$=STRING$(2,159)
400 GOSUB 340
410 FOR Q=1 TO 9
420 FOR E=1 TO 20:READ A:A$=A$+C
HR$(A):NEXT E
430 PRINT@V,A$;:A$="":V=V+32
440 NEXT Q
450 PRINT@425,"ROBERT E. RICE";:
GOTO560
460 DATA 133,140,140,136,133,130
,128,138,128,134,137,128,133,128
,129,136,133,140,140,136
470 DATA 133,131,131,138,133,133
,138,138,133,131,131,138,133,131
,136,128,133,131,130,128
480 DATA 128,128,128,138,133,128
,137,138,133,128,128,138,133,128
,137,128,133,128,128,128
490 DATA 132,140,140,136,132,128
,128,136,132,128,128,136,132,128
,128,136,132,140,140,136
500 DATA 128,128,128,128,128,128
,128,128,128,128,128,128,128,128
,128,128,128,128,128,128
510 DATA 133,140,140,138,133,128
,128,138,128,134,137,128,133,140
,140,136,133,140,140,136
520 DATA 133,128,128,128,133,131
,131,138,133,131,131,138,133,131
,131,130,133,131,130,128
530 DATA 133,128,128,130,133,128
,128,138,133,128,128,138,128,128
,128,138,133,128,128,128
540 DATA 132,140,140,136,132,128
,128,136,132,128,128,136,132,140
,140,136,132,140,140,136
550 '**SNAKE MUSIC
560 FOR Q=1 TO 500:NEXT Q:FOR Q=
1 TO 42:READ A,E:SOUNDA,E:NEXT Q
570 DATA 89,3,108,3,117,6,108,6,
89,6,89,3,108,3,117,3,147,3,108,
3,117,3,89,6
580 DATA 117,3,133,3,147,3,147,1
,147,1,147,3,153,3,147,3,133,3,1
08,3,117,3,133,3,133,1,133,1,133
,3
590 DATA 147,3,133,3,117,3,89,3,
108,3,117,6,108,6,89,6,89,3,108,
3,117,3,147,3,108,3,117,3,89,10
600 '**INTRO
610 CLS3:GOSUB 340:FOR Q=1TO500:
NEXT Q
620 PRINT@68,"NAME OF PLAYER ON
LEFT?";:PRINT@300,"";:LINEINPUT
L$:SOUND190,2
630 CLS3:GOSUB 340:FOR Q=1 TO 50
:NEXT Q:PRINT@68,"NAME OF PLAYER
 ON RIGHT?";:PRINT@300,"";:LINEI
NPUT R$:SOUND190,2
640 CLS3:GOSUB 340:PRINT@164,"DO
 YOU WANT INSTRUCTIONS?";
650 PRINT@269,"y OR n";
660 A$=INKEY$:IF A$="" THEN 660
ELSE IF A$<>"Y" THEN 730
670 CLS4:PRINT@32,"  EACH PLAYER
 HAS A SNAKE. TO      WIN, YOU MU
ST KEEP YOURS        MOVING WITH
OUT BUMPING INTO     ANY LINES."
```

continued on Page 62.....

JOYSTICKS, TOUCH PADS AND DIGITIZING THE WORLD



By William Barden, Jr.

I hate computers. They're cantankerous, unreliable and hard to use. Ah, but the things you can do with them! My latest series of misadventures began with a course I'm taking in the Japanese language. The written Japanese language uses three types of symbols — Hiragana, Katakana and Kanji. Kanji is derived from Chinese symbols and uses "ideographs" to represent objects. The symbol shown in Figure 1, for example, represents a dog. Hiragana and Katakana, on the other hand, represent phonetic sounds, such as "ga" or "ka" as shown in the same figure.

As usual, I was attempting to use a computer to solve a real-world problem. Wouldn't it be neat, I reasoned, if I could convert the Japanese symbols to Color Computer screen symbols? Then I could drill myself on them and use the Color Computer as a silicon sensei!

This goal yielded all kinds of interesting results I'd like to share. Some of the results are good and some are bad. They involve joysticks, old and new, touch pads, mouse devices and other techniques. They all fall into an area I call "digitizing patterns."

The techniques talked about can be applied on any Color Computer system, whether it's a 16K cassette-based system or a 64K OS-9 system. We'll stick to the basics in this discussion, so as not to leave any beginners in the dust. The only requirement is that you have Extended Color BASIC in a 16K system.

How to Take the "Joy" Out of Joystick

My first thought in converting the Japanese characters to screen patterns was to try a joystick in free-hand drawing. This proved to be a disaster, however. For one thing, my joysticks, unlike fine wine, have not improved with age. A quick check revealed that they did not reproduce a smooth pattern on the screen.

I used the program, *Joystick Test*, shown in Listing 1 to try them out. This program uses the JOYSTK command to read in joystick values from the 'X' direction (JOYSTK(0)) and from the 'Y' direction (JOYSTK(1)), then plots them on the screen in high resolution mode. (We'll use high resolution mode, PMODE 4, in these examples, assuming that we want the maximum resolution, or number of points on the screen.)

The JOYSTK command reads a value of zero through 63 representing the joystick 'X' or 'Y' position on the screen. Zero is the extreme left or top position, while 63 is the extreme right or bottom position. We multiplied the 'X' value by four to convert it to zero through 252 and the 'Y' value by three to convert it to zero through 189 to get a display on the entire screen, rather than just the upper left quadrant.

To use Listing 1, move the joystick smoothly in the 'X' and 'Y' directions. Make sure all new points appear in the direction the joystick is moving and not "behind" the path, as shown in Figure 2. If points appear behind the path, your joystick is bad, or at least not very good for digitizing. Make certain also that you can get to all corners of the screen.

If you have a newer "Deluxe" joystick (Cat. No. 26-3012), the 'X' and 'Y' "trim tab" controls can be adjusted to center the joystick initially. You should also set the movement to "free" by the switches on the bottom of the joystick.

When I tried the program of Listing 1 on my older joysticks (Radio Shack Cat. No. 26-3018), I found that operation was erratic, with points displayed behind the path and abrupt jumps from one screen position to a new one an eighth of a screen away. I ended up throwing both sticks away after they expired during an emergency operation using contact cleaner.

A Deluxe Solution to the Joystick Problem

My next step in the quest for Kanji

was to get a new Deluxe joystick. The new one proved excellent in smoothness and I had no problem with points appearing behind the path. However, it verified something I should have known all along — a joystick is not a good device to use to reproduce a pattern. For one thing, it cannot trace a pattern, but must be used in a kind of free-hand mode to draw the pattern. My best attempt at reproducing a simple Katakana character by joystick is shown in Figure 3.

The Touch Pad Scores too many Points!

While paging through the new *Radio Shack Computer Catalog*, I came across another solution to my problem. What about the Touch Pad (Cat. No. 26-1185)?

In case you're not familiar with the Touch Pad, it's a "Koala Pad" in Tandy clothing. Like a joystick, it returns an X/Y position, but instead of a stick that moves in two directions, you use a "stylus" on a tablet. As the stylus touches the pad, the proper X/Y coordinates are returned. It sounded perfect for my needs, as I could simply place a small sheet over the pad and trace a Japanese character.

The instruction book for the Touch Pad is akin to an operator's manual for a new car — it tells you how to empty the ash trays but not how to time the engine. I was looking for the "resolution" of the pad, the number of individual points per inch. "Would it give results as good as a joystick?" I asked the Computer Center salesman.

"No problem," he vowed, putting his hand on a Color Computer BASIC manual as an ersatz bible.

I hooked up the Touch Pad to my Color Computer and was tracing the first pattern. As I drew the first line, I noticed a duplicate set of points somewhat displaced on the screen, as shown in Figure 4. "Must be a bad pad," I mused as I boxed it up again and drove back to the Computer Center.

After trying a second Touch Pad, I was convinced the problem was in the design of the pad. Thumbing through some reviews of the Touch Pad in my magazine collection and a brief conversation with another computer writer confirmed my suspicions. The Touch Pad can possibly be used to select one square out of a matrix of squares, as shown in Figure 5, but is simply not a good "digitizing" device. Fortunately, Radio Shack cheerfully refunds your money.

To Catch a Mouse, Make a Noise Like a CoCo

I looked at an alternative device, the Color Computer Mouse (Cat. No. 26-3025). This mouse is a "country" mouse, unlike the high-fashion "city" mouse from Bellevue for the Tandy 1000. The 1000 mouse requires a Digi-Mouse Controller board that costs as much as the mouse itself ($99.95). The CoCo mouse, however, is a steal at $49.95.

I plugged the mouse into the joystick port. I might add here that the mouse is really just another form of a joystick; it returns values of zero through 63 like the joystick and can be read the same way with the JOYSTK commands. I put a "head" on the mouse and used it to follow the lines of the Japanese character. The results are shown in Figure 6. They are not bad, but still not as good as I was looking for.

Figure 2: Joystick Testing

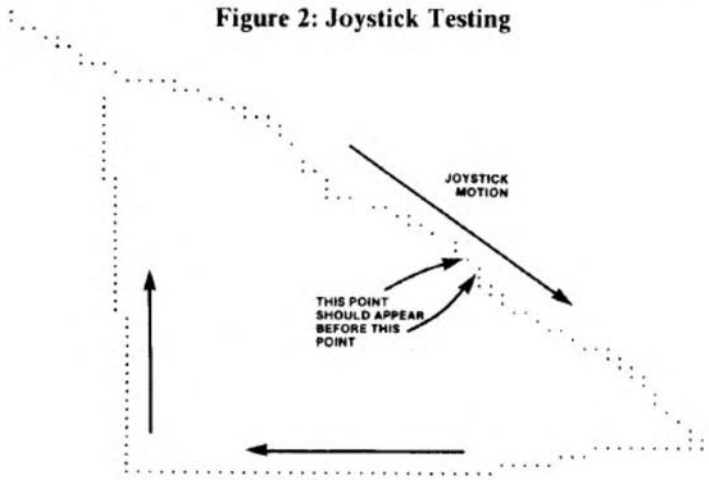


Figure 3: Katakana Characting Using Joystick

Figure 4: Touch Pad and Spurious Points

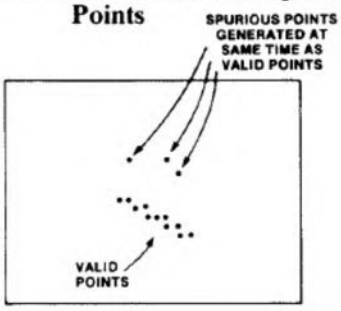


Figure 5: Touch Pad Use in Selecting an Area

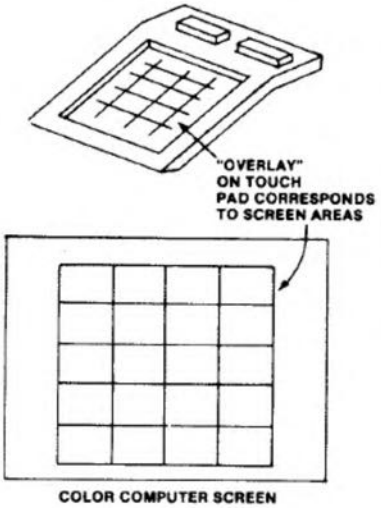
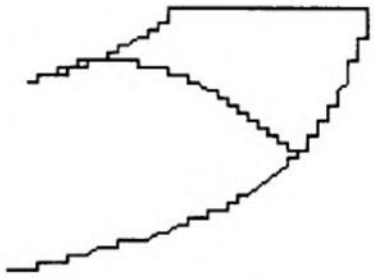


Figure 6: Katakana Character Using a "Mouse"

Figure 7: Using a "Line" Program to Fill in Points



Problems with Joysticks

At this point, I realized I had rushed into a solution before thinking the problem through. I asked myself, "*Are* joysticks good to use for digitized patterns and shapes? Or are there better ways?"

One of the problems with the joystick, Touch Pad or mouse on the Color Computer is that the best resolution is only 64 by 64 points on the screen. As we'd like 256 points horizontally by 192 points vertically, basic joystick operation leaves something missing, namely three points in between each joystick point! We can "pad out" those points by drawing line segments, but the results are still not the best. (See Listing 2, *Joystick Line Tracer*, for a program to fill in the missing points and Figure 7 for the results.)

An alternative is to use the three devices on only one screen quadrant at a time, as shown in Listing 3, *Joystick Quadrant*, and Figure 8. Each press of the space bar switches to the next screen quadrant. Holding down the firebutton on the joystick or mouse inhibits writing on the screen so the screen position can be changed.

Another problem with a joystick-type device is that it's virtually impossible to produce a pattern without "garbage" points that must be cleaned up, as shown in Figure 9. What's really needed is a "pattern editor."

Hardware Solutions to Digitization

One possible solution to the digitization problem is a piece of equipment from Colorware, Inc., known as *CoCo Max*. This product circumvents the low resolution capability of the joystick input by using its own joystick electronics. A total of 256 points instead of 64 points can be read in from a standard joystick or mouse to match the the maximum screen resolution.

In addition to the increased joystick resolution, *CoCo Max* contains a complete graphics package for higher screen resolution. It allows shapes and patterns to be drawn and manipulated in methods very similar to the MacIntosh's *MacPaint* program. (To be honest, I have not tried the product but it looks very interesting.)

A second hardware solution to the digitizing problem is a better resolution graphics tablet. Radio Shack has one with the GT-2000 Graphics Tablet, and it allows up to 200 pixels per inch. However, its cost of $599.95 is most likely prohibitive to the typical Color Computer user.

Another possible hardware solution is a digitizing television device. These are available from RAINBOW advertisers, such as The MicroWorks — their DS-69 Video Digitizer is $149.95. Video digitizers take a television image from a black-and-white camera or VCR and convert the television picture into patterns for display on the Color Computer screen. Again, this solution involves quite a bit of cash outlay when a television camera and other options are added in. Also, the resulting digitized picture takes a good deal of memory or disk space.

Along the same lines, Forrest Mims III, of Radio Shack *Engineer's Notebook* fame, did some experiments using a photocell device that digitized images from paper put into a "plotter." The same technique can be used by attaching the device to the print head of a Color Computer printer. The print head

Figure 8: Quadrant Program Operation

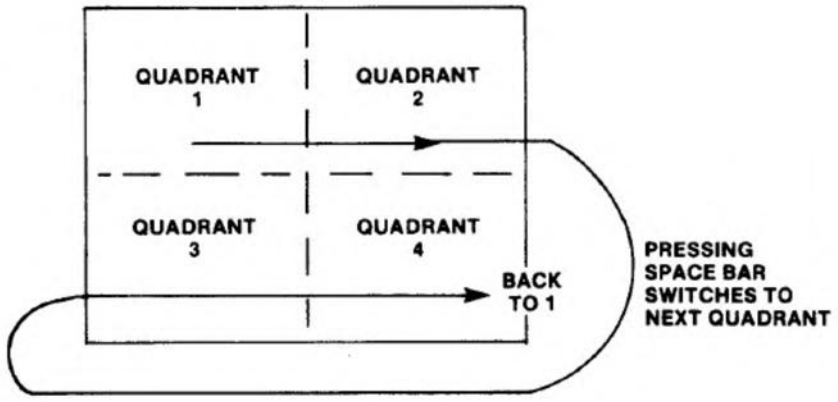
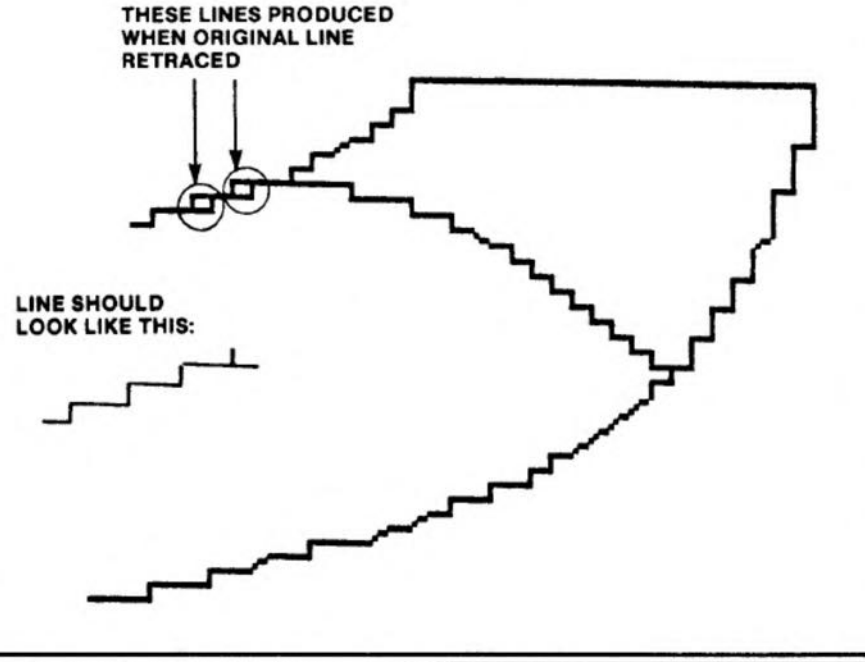


Figure 9: "Garbage" Points on Screen



of the printer is moved a character position at a time (or less, if the graphics mode is used) under program control. For each new character position, the output of the photocell is converted into a black or white level, which is read by the Color Computer from the joystick port.

This scheme is not new, but is particularly attractive for the Color Computer because of the built-in analog-to-digital converter logic of the joystick port. For those who are interested, see *Forrest Mims' Computer Projects* (McGraw Hill, 1985).

A Software Solution to Digitization

All along we've been considering only "hardware" solutions to the problem of digitizing data. There's a whole set of software solutions as well, ranging from using PSETs and DRAW commands to large applications programs from a variety of vendors. There are a large number of graphics drawing programs that allow you to create and edit patterns and shapes on the screen.

The solution to my problem is not the most sophisticated, but it does work fairly well. It offers full resolution of 256 points horizontally by 192 points vertically. It is fairly quick and allows editing of the data. It works well for the problem I was interested in, and should work for similar types of problems you might have in digitizing data. I call it *Digitize*; the program is shown in Listing 4. It works on any Extended BASIC CoCo with at least 16K of memory.

Digitize uses cursor positioning keys to move a graphics cursor around the screen, as shown in Figure 10. As the cursor is moved, it draws a line. The cursor can also be moved without drawing a line by "toggling" a pen up/pen down condition.

Many times you'd like to draw a broader line than just one pixel width, so the size of the line can be increased by selecting a larger "penpoint." The size of the penpoint is shown on the screen by a circle on the screen bottom. If the circle is filled in, the pen is down; if the circle is blank, the pen is up. Changing the penpoint size enables you to fill in gaps that have not been digitized.

Once the pattern or picture is digitized, you can save the screen to a cassette or disk file by pressing 'S'. The graphics screen is then replaced by a text screen display so you can name the file to be used. Pressing the 'R' key at any time reads in a previous file so further editing can be done. As the screen files are simply dumps of the graphics screens, the files created in your own BASIC programs can be used for further processing.

Digitize allows you to easily do digitizations of simple shapes, and while admittedly is not a full-fledged graphics program, it is short, easy to use and foolproof. I've used *Digitize* to digitize Japanese characters (see Figure 11) and other patterns, shapes and drawings, and use the following approach for rapid entry: Using a copier service, make a transparency of the pattern to be reproduced. Tape this transparency to the television screen and use the cursor controls to trace the pattern.

Alternatively, you can simply tape a piece of plastic wrap to the screen and use a grease pencil to draw the pattern to be digitized upon the plastic wrap. *Digitize* works much faster than constructing elaborate sequences of DRAW commands to produce patterns and shapes.

In my quest to reproduce Kanji and Katakana patterns I have gone through several approaches and learned that computers can't do everything. Some human intervention is necessary unless you're willing to lay out hundreds of dollars for television digitization systems or digitizing tablets, and even then the results would probably not be as expected.

I really do hate computers, but I've learned to live with my Color Computer, and *Digitize* is another concession on my part for using the power of the Color Computer to solve a perplexing real-world problem. □

Figure 10: Cursor Positioning and Option Keys

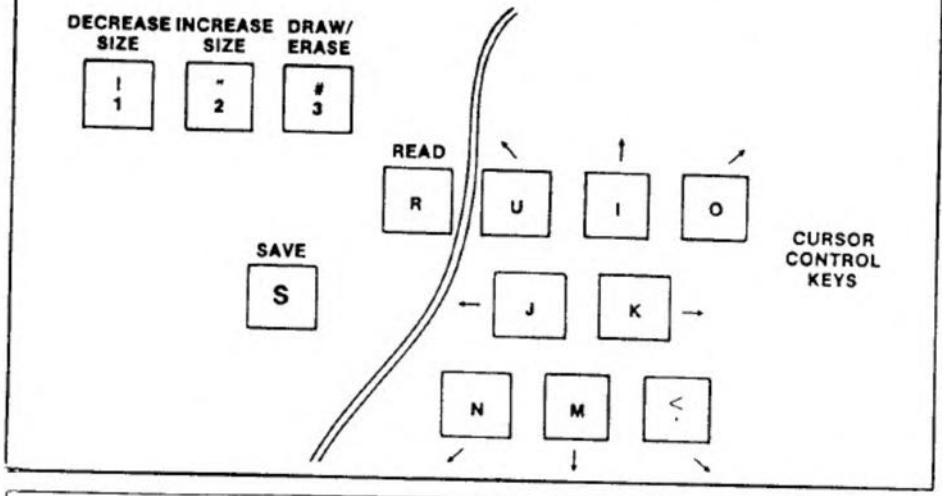


Figure 11: *Digitize* Sample

Listing 1: JSTKTEST

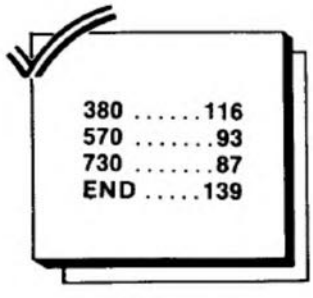
```
100 '*****************************
110 '* JOYSTICK TEST ROUTINE     *
120 '* USE WITH RIGHT JOYSTICK   *
130 '*****************************
140 SCREEN 1,0
150 PMODE 4,1
160 PCLS
170 X=JOYSTK(0)*4
180 Y=JOYSTK(1)*3
190 PSET (X,Y)
200 GOTO 170
```

Listing 2: JSTKLINE

```
100 '*****************************
110 '* JOYSTICK LINE TRACER      *
120 '* FILLS IN LINES BETWEEN    *
130 '* POINTS. HIT ANY KEY TO    *
140 '* CLEAR SCREEN.             *
150 '*****************************
160 SCREEN 1,0
170 PMODE 4,1
180 PCLS
190 XX=JOYSTK(0)
200 YY=JOYSTK(1)
210 A$=INKEY$:IF A$()""THEN PCLS
220 X=JOYSTK(0): Y=JOYSTK(1)
230 LINE (XX*4,YY*3)-(X*4,Y*3),
    PSET: XX=X: YY=Y
240 GOTO 210
```

Listing 3: JSTKQUAD

```
100 '****************************
110 '* JOYSTICK QUADRANT ROU-   *
120 '* TINE. PRESS SPACE BAR TO*
130 '* MOVE TO NEXT QUADRANT.   *
140 '* PRESS CLEAR TO CLEAR     *
150 '* SCREEN. PRESS JOYSTICK   *
160 '* "FIRE" BUTTON TO MOVE    *
170 '* WITHOUT DRAWING LINE.    *
180 '* USE WITH RIGHT JOYSTICK *
190 '****************************
200 SCREEN 1,0
210 PMODE 4,1
220 PCLS
230 QUAD=0
240 XB=0: YB=0
250 XX=JOYSTK(0): YY=JOYSTK(1)
260 A$=INKEY$
270 IF A$=CHR$(12)
    THEN PCLS: GOTO 330
280 IF A$<>" " THEN 330
290 QUAD=QUAD+1
300 IF QUAD=4 THEN QUAD=0
310 IF QUAD=0 OR QUAD=2
    THEN XB=0 ELSE XB=128
320 IF QUAD=0 OR QUAD=1
    THEN YB=0 ELSE YB=96
330 X=JOYSTK(0): Y=JOYSTK(1)
340 IF (PEEK(&HFF00) AND 1)=0
    THEN 360
350 IF (PEEK(&HFF00) AND 1)=1
    THEN LINE(XX*2+XB,YY*1.5+YB)
    -(X*2+XB,Y*1.5+YB),PSET
360 XX=X: YY=Y
370 GOTO 260
```



Listing 4: DIGITIZE

```
100 '****************************
110 'DIGITIZE PROGRAM. DIGITIZES
120 'BY MOVING SCREEN TRACE UN-
130 'DER TRANSPARENT OVERLAY.
140 ' I=MOVE UP; O=MOVE UP,RGHT;
150 ' K=MOVE RIGHT; <=MOVE DWN,
160 ' RIGHT; M=MOVE DOWN;N=MOVE
170 ' DOWN, LEFT; J=MOVE LEFT;
180 ' U=MOVE UP, LEFT; S=SAVE;
190 ' R=READ; 1=DECREASE LINE
200 ' SIZE; 2=INCREASE LINE
210 ' SIZE; 3=TOGGLE CURSOR OFF
220 ' AND ON.
230 '****************************
240 CLS
250 SCREEN 1,0
260 PMODE 4,1
270 PCLS
280 RADIUS=0
290 PEN=1
300 X=128
310 Y=96
320 GOSUB 380
330 GOSUB 800
340 GOSUB 450
350 GOTO 330
360 '****************************
370 'SUBROUTINE TO DRAW LEGEND
380 LINE (0,165)-(30,191),
    PRESET,BF
390 IF RADIUS=0 AND PEN=1
    THEN PSET(15,180)
400 IF RADIUS<>0 AND PEN=0
    THEN CIRCLE (15,180),RADIUS:
    GOTO 420
410 IF RADIUS<>0 AND PEN=1
    THEN CIRCLE (15,180),RADIUS:
    PAINT (15,180)
420 RETURN
430 '****************************
440 'SUBROUTINE TO READ KEY.
450 A$=INKEY$: IF A$="" THEN 450
460 IF A$<>"I" THEN 480
    ELSE Y=Y-1
470 GOTO 770
480 IF A$<>"O" THEN 510
    ELSE Y=Y-1: IF Y<0 THEN Y=0
490 X=X+1: IF X=256 THEN X=255
500 GOTO 770
510 IF A$<>"K" THEN 530
    ELSE X=X+1: IF X=256
    THEN X=255: GOTO 770
520 GOTO 770
530 IF A$<>"<" AND A$<>","
    THEN 560 ELSE X=X+1:
    IF X=256 THEN X=255
540 Y=Y+1: IF Y=192 THEN Y=191
550 GOTO 770
560 IF A$<>"M" THEN 580
    ELSE Y=Y+1: IF Y=192
    THEN Y=191
570 GOTO 770
580 IF A$<>"N" THEN 610
    ELSE Y=Y+1: IF Y=191
    THEN Y=191
590 X=X-1: IF X=-1 THEN X=0
600 GOTO 770
610 IF A$<>"J" THEN 630 ELSE
    X=X-1: IF X=-1 THEN X=0
620 GOTO 770
630 IF A$<>"U" THEN 660 ELSE
    X=X-1: IF X=-1 THEN X=1
640 Y=Y-1: IF Y=-1 THEN Y=1
650 GOTO 770
660 IF A$<>"I" THEN 680:
    Y=Y-1: IF Y=-1 THEN Y=0
670 GOTO 770
680 IF A$<>"2" THEN 700
    ELSE RADIUS=RADIUS+1:
    IF RADIUS>10 THEN RADIUS=10
690 GOSUB 380: GOTO 770
700 IF A$<>"1" THEN 720
    ELSE RADIUS=RADIUS-1:
    IF RADIUS=-1 THEN RADIUS=0
710 GOSUB 380: GOTO 770
720 IF A$<>"3" THEN 740
    ELSE IF PEN=0 THEN PEN=1
    ELSE PEN=0
730 GOSUB 380: GOTO 770
740 IF A$<>"S" THEN 750
    ELSE GOSUB 850: GOTO 770
750 IF A$<>"R" THEN 760
    ELSE GOSUB 950
760 IF A$<>"C" THEN 770 ELSE 240
770 RETURN
780 '****************************
790 'SUBROUTINE TO DRAW POINT
     OR CIRCLE
800 IF RADIUS<>0 THEN CIRCLE
    (X,Y),RADIUS,PEN: GOTO 820
810 IF RADIUS=0 THEN IF PEN=0
    THEN PRESET(X,Y) ELSE
    PSET(X,Y)
820 RETURN
830 '****************************
840 'SUBROUTINE TO SAVE GRAPHICS
     SCREEN
850 SCREEN 0,0
860 CLS
870 INPUT "SAVE FILE NAME"; A$
880 IF A$="" THEN 910
890 SAVEM A$,&H0E00,&H25FF,
    &H0E00
900 'USE "CSAVEM A$,&H0600,
     &H1DFF" FOR CASSETTE SYSTEM
910 SCREEN 1,0
920 RETURN
930 '****************************
940 'SUBROUTINE TO READ GRAPHICS
     SCREEN
950 SCREEN 0,0
960 CLS
970 INPUT "READ FILE NAME"; A$
980 IF A$="" THEN 1010
990 LOADM A$
1000 'USE "575 CLOADM A$" FOR
      CASSETTE SYSTEMS
1010 SCREEN 1,0
1020 RETURN
```

SLEEP TIGHT

Your CoCo Is Awake Tonight

Part 2

This month, we'll finish up the alarm circuit by building a sophisticated local alarm system

By Dennis H. Weide

Last month, I showed you a simple alarm circuit that allowed scanning an alarm system in your house or apartment while you're away from home. This month, we'll use the same basic circuits to build a sophisticated local alarm system. This one has a hardware clock, an outside alarm bell and continuous alarm scanning. All alarms and activity are logged on disk. It's turned on and off at the front door via a key switch and has an LED indicator to show on/off status.

The Program

Listing 1 is the local alarm program. Six commands allow you to control all functions of the program. A menu is displayed when the program is first started and anytime the ENTER key is pressed. The prompt has been changed to "GO >" for easier recognition. Let's take a look at the commands.

Set Time

The correct time can be set with the SET TIME command. The clock should never vary more than a minute a day if built correctly. However, each time the alarm program is loaded and run, the clock must be set in software. Follow the prompts to set the time.

Alarm Log

All activity on the system is recorded on disk in the alarm log. The log is a direct access file, so the amount of disk space required is kept to a minimum. When the ALARM LOG command is entered, the program accesses the disk and prints all information in the log. All alarms (attempted break-ins) are printed on the screen along with the date and time the alarm is detected.

Print Log

The PRINT LOG command prints a hard copy that gives a permanent record of the log. You can use this record in the event of a break-in. It's a good idea to clear the log each time the program is started.

Clear Log

The CLEAR LOG command is used to erase the alarm log when you're sure it's no longer needed. The file is erased and a new one is created with the entry log cleared and the date and time.

Bell Off

The bell on the outside of the house alerts the neighborhood of an attempted break-in. When an alarm is tripped, the bell rings until shut off by the BELL OFF command. This command releases the cassette motor relay (the MOTOR OFF command) and silences the bell.

Sign Off

When you wish to stop program execution, use the SIGN OFF command. This stops the program and suspends the alarm system. If you wish to start the system again, type CONT and ENTER. The program jumps to start and you can set up the system again.

Command Entry

All commands are entered as a single number only. Enter the number next to the command to execute it. The program scans the clock and alarm circuits continuously, except when reading or writing to disk or executing a command. Now let's look at the hardware we'll be using.

(Dennis Weide is an avid computer hobbyist who teaches programming on the CoCo and IBM PC. He has written for all the major CoCo magazines, including many programs in RAINBOW. Dennis lives in Albuquerque, N.M.)

Table 1
Typical Circuit Values for Clock Circuit

Sec/Cycle	R1 Value	R2 Value	C2 Value
60	68K	4.3M	10 uF
30	27K	2.151M	10 uF
15	56K	1.56M	10 uF

Formulas

High output Time 1=.693*(R1+R2)*C1
Low output Time 2=.693*R2*C1
Total cycle time = Time 1+time 2 in seconds

Table 2
Clock Circuit Parts List

Item	Price	Radio Shack No.
555 IC Timer	$1.19 ea.	276-1723
68K ohm Resistor	.39 pkg/5	271-1345
4.3M ohm Resistor (see text)		
10 Mfd. Elect. Capacitor	.59 ea.	272-1013
.01 Mfd. Capacitor	.59 pkg/2	272-1065
220 ohm Resistor (2 ea.)	.39 pkg/5	271-1313
TLR-107 Hi-Brite LED	.89 pkg/2	276-033

Table 3
Alarm Circuit Parts List

Item	Price	Radio Shack No.
N/C Window Switches	$ 3.49 ea.	49-495
N/O Tamper Switch	1.39 ea.	49-528
Door Lock Switch	9.95 ea.	49-511
N/C 120 ft. Foil	5.99 roll	49-502
N/C Foil Connectors	2.59 pkg/6	49-504
Joystick Plugs (2 ea.)	1.19 ea.	274-020
Cassette Plug (1 ea.)	1.49 ea.	274-003
33K ohm Resistors (3 ea.)	.39 pkg/5	271-1341
68K ohm Resistors (3 ea.)	.39 pkg/5	271-1345
100K ohm Resistors (3 ea.)	.39 pkg/5	271-1347
220 ohm Resistor (1 ea.)	.39 pkg/5	271-1313
TLR-107 Hi-Brite LED	.89 pkg/2	276-033
Hook-up and Alarm Wire	N/A	N/A
Alarm Bell	19.95 ea.	49-498

Table 4
Power Supply Parts List

Item	Price	Radio Shack No.
12.6V Mini Transformer	$3.59 ea.	273-1365
1000 uF Elect. Capacitor	1.59 ea.	272-1019
.01 uF Epoxy Capacitor	.59 pkg/2	272-1065
Full Wave Bridge Rectifier	.89 ea.	276-1161
7805 Fixed IC Regulator	1.59 ea.	276-1770
Sub-Mini SPST Toggle Switch	1.59 ea.	275-612
Fuse Holder	.99 ea.	270-367
Fuses	.69 pkg/3	270-1271
Project Box for Power Supply and Alarm Board	3.99 ea.	270-252

The Clock Circuit

Schematic 1 shows the hardware clock circuit that provides the date and time for all disk writes. A 555 timer is used to generate a square wave signal with a cycle time of one minute. This cycle time requires the computer to scan the clock circuit only once every 30 seconds to be accurate. To assure accuracy of the clock, use the component values shown on the schematic. Everytime the clock output goes high, clock memory is incremented. The output of the clock is connected to Pin 1 of the right joystick (JOYSTK(0)). Clock ground is connected to Pin 3 (ground) of the same joystick.

Power for the clock is provided by the external +5 volt supply (Schematic 3), which is also connected to the joystick ground. The LED and two 220 ohm resistors (R3 and R4) provide a visual indication when the clock output goes high (LED lighted). Use this indicator and resistor R2 when adjusting the timing of the circuit.

Resistors R1 and R2 and capacitor C2 are used to determine the cycle time. The values shown generate a square wave that is almost perfect. Table 1 shows the cycle time and the resistor and capacitor values required to achieve them. R1 is a standard value resistor but R2 is not. I used four 1M ohm resistors in series with a 500K ohm pot to allow the cycle time to be adjustable.

For those who require perfection, you can add a 7473 chip (J-K Flip-Flop) to the output to get a 50 percent duty cycle (a perfect square wave). The output from the flip-flop will be one-half the clock cycle time. Therefore, you must build the timer circuit to generate a clock signal with a cycle time of 30 seconds. Table 1 shows the formulas to use to determine cycle time for the clock output. Table 2 is a parts list for the clock circuit.

The Clock Program

A machine language subroutine (Listing 2) is used to read the hardware clock. The program scans the clock on JOYSTK(0) and the alarms on JOYSTK(1) to JOYSTK(3). The clock is totally controlled by the machine language program while the alarm values are passed to BASIC for computing the alarm conditions.

Lines 2100 to 4300 of the BASIC program (Listing 1) contain the machine language programs in BASIC DATA

statements. There is no need to enter the assembly language listing.

If you want to use the clock and program in other applications, execute Hex address 7100 to initialize the clock and Hex address 7129 every time you wish to scan the clock. You can build your own timer for any cycle time desired. Just remember that the clock circuit must be scanned at least twice each cycle to be accurate.

Local Alarm Loops

The local alarm loops are shown in Schematic 2. They're similar to the basic circuit shown last month. Switches S1 and S2 represent the on/off switch and the tamper switch. They work the same way as the remote alarm system except relays are not needed since the program scans the alarm circuit continuously. When an alarm is detected, the condition is registered in memory so it's not necessary to hold an alarm relay operated.

Switches S3 through S6 are the window and door switches. As shown last month, they are a series of switches — one for each door and window on the specified side of the house. They should be closed when the door or window is closed. The schematic also shows the joystick assignments for the four sides of the house. Table 3 is a parts list for the local alarm.

External Power Supply

Last month, I showed the schematic of an external power supply used to prevent loading of the CoCo power supply. I have included the schematic again for reference (Schematic 3) and a parts list (Table 4). Be sure to ground the power supply to Pin 3 of either joystick port. Failure to do so could end in disaster. For an explanation of the power supply, see last month's edition of RAINBOW (Page 58).

The Alarm Bell Circuit

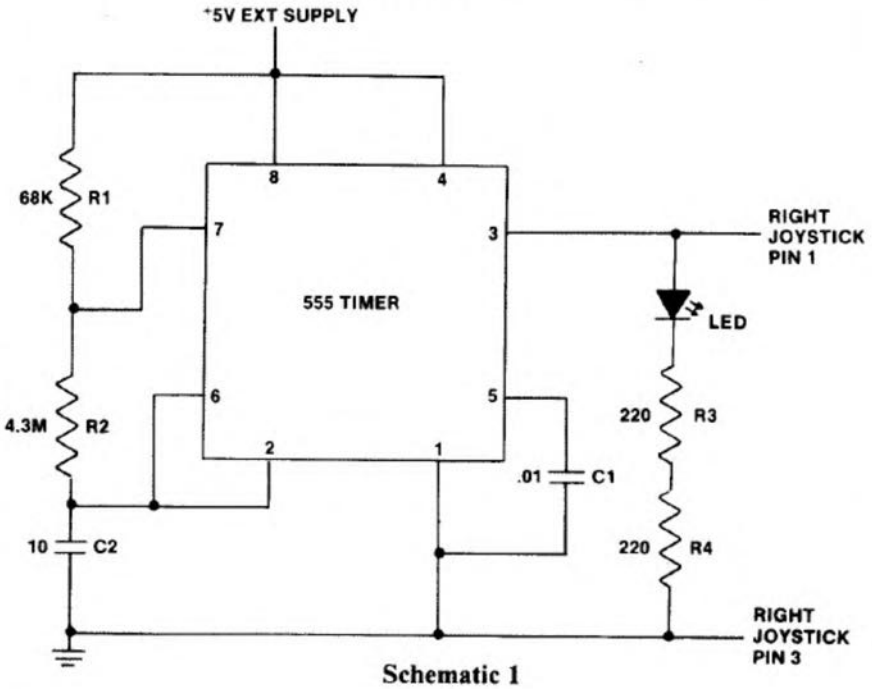
This alarm has another feature not used in last month's project. An alarm bell is located outside the house in an inaccessible place. When an alarm is activated, the bell rings.

Connection to the computer is made via the cassette jack on the back of the computer. This is the cassette motor relay, which is operated and released by the MOTOR ON and MOTOR OFF commands. The BELL OFF command silences the bell.

The circuit is simple. I use a car battery (60 amp-hours), which can ring the bell for several days if necessary. You must charge the battery at regular intervals, but be careful not to overcharge it. Those experienced in electronics can build a small charging circuit to provide a continuous charge. Place the battery in a plastic battery box and coat all connections with clear Karo syrup to prevent corrosion. The syrup will harden and, when necessary, can be removed with warm water. It's an excellent anti-corrosive.

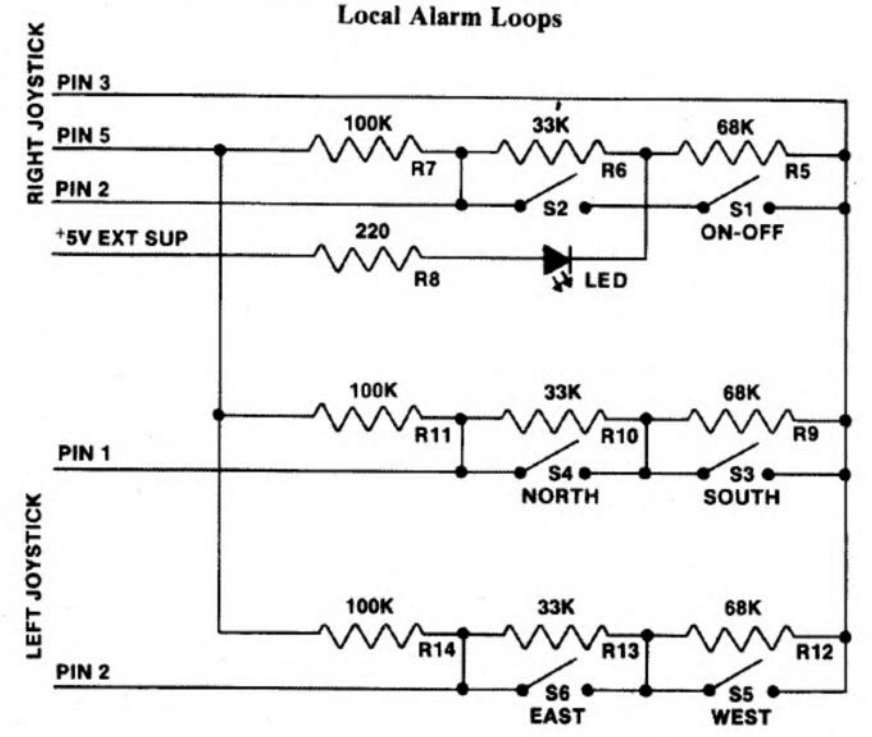
I used two standard Radio Shack alarm bells that draw ½ amp each at 12 volts. Locate your bells where they are inaccessible so they can't be tampered with.

Clock Circuit

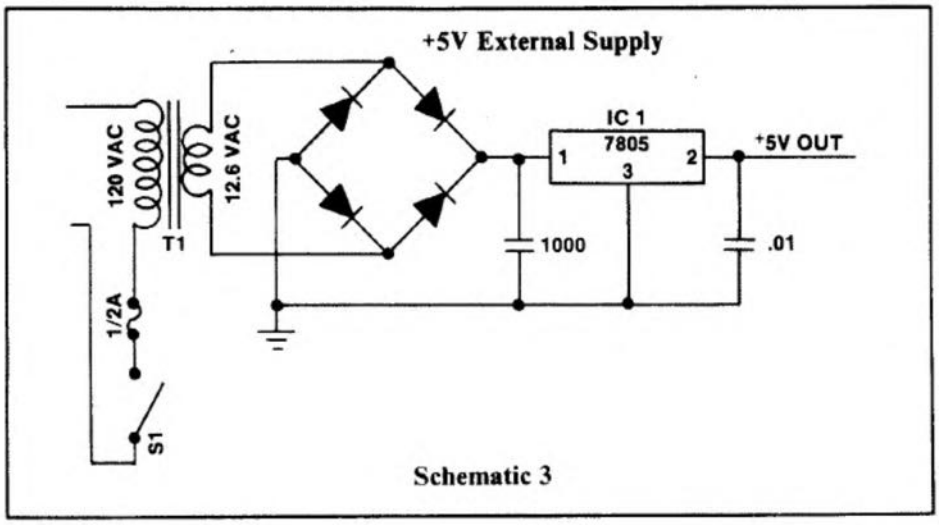


Schematic 1

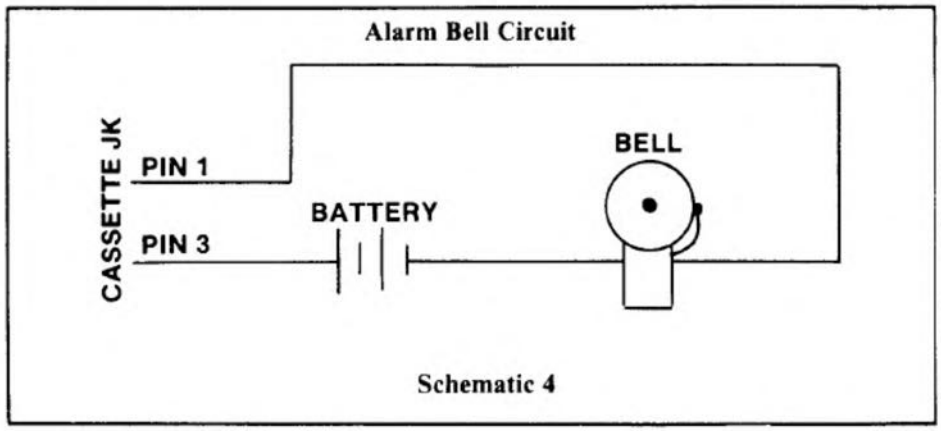
Local Alarm Loops



Schematic 2



Schematic 3



Schematic 4

The Heat Is On

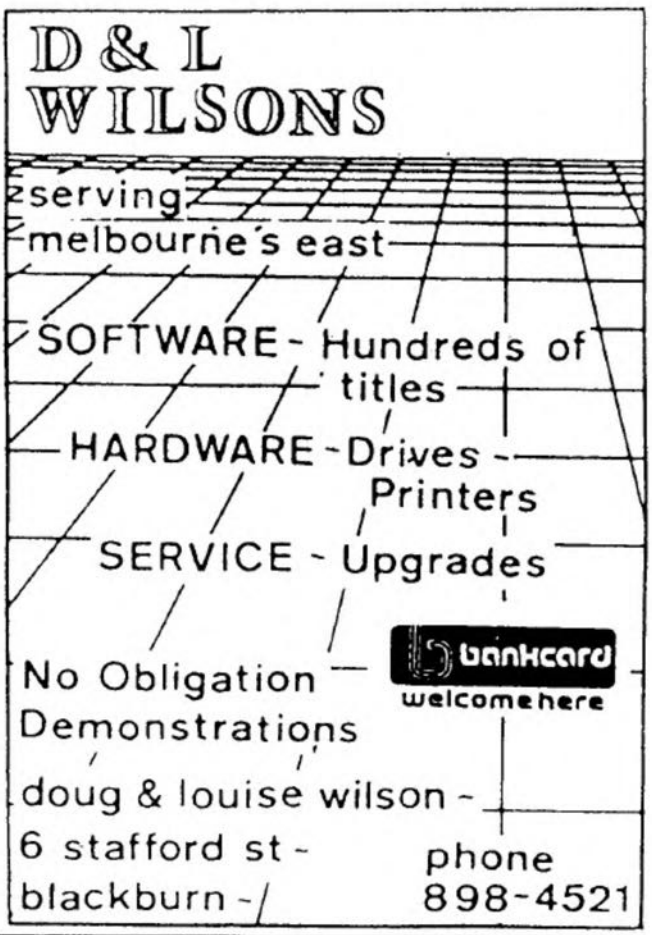
Those who are worried about the CoCo overheating can purchase one of several fan attachments to keep it cool. However, if you leave the computer on for several hours with no ill effects, you probably don't have anything to worry about. I've had two CoCos and several other computers, and they all ran for over 16 hours a day for weeks on end. As long as the computer is properly ventilated, overheating shouldn't be a problem.

A Few More Notes

This alarm system was built first and later modified to provide the one shown last month. No matter which one you choose, the feeling of security you'll get is well worth the effort.

A parts list is included for each phase of the project (Tables 1 through 4). You may be able to scavenge many of the parts or order them from parts houses for less. You can also wait until Radio Shack puts the parts that are needed on sale. That's what I did and I saved a bundle.

Anyone having questions or comments can write to me at 14201 Marquette N.E., Albuquerque, NM 87123. Please enclose an SASE if you wish a reply. □



Tandy ELECTRONICS

COLOR COMPUTER THERMAL PRINTER

Ideal For Use With Your Tandy Color Computer 2

Save $60

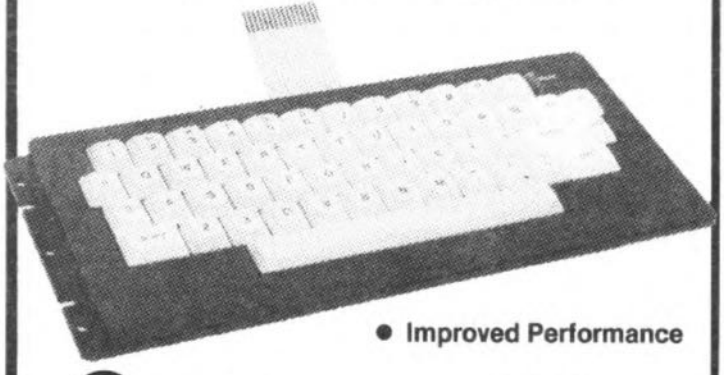
Reg **159.95**

99.95

- **Prints Graphics And Alphanumerics**
- **Repeat Function For Easy Graphics**
- **Whisper Quiet**

An affordable way to expand your computer system and print reports, notes, letters, diagrams or graphs! This compact dot-matrix unit prints 30 characters per second on 10.47 cm wide thermal paper, and includes an elongated mode for expanded type. Easy to use. 26-1261

Low Profile Keyboard



- **Improved Performance**

Save $20

Reg **59.95**

39.95

Upgrade your outdated grey pushbutton-style Color Computer keyboard with this new typewriter-style one! Easier to use, greater efficiency. 26-3016

Sound/Speech Program Pak

159.95

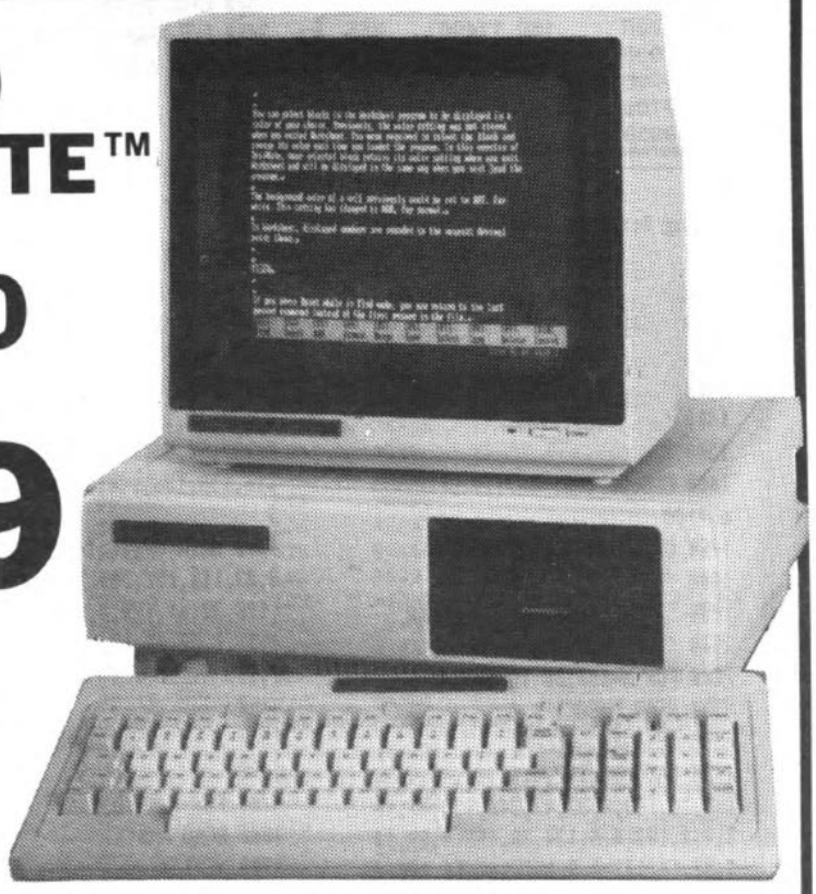
Add sound and voices to your BASIC programs using PEEK and POKE commands. It even lets you add text to the speech in your programs. 26-3144

TANDY 1000 WITH DESKMATE™

Save $200

Reg **1999.00**

NEW LOW PRICE **$1799**



The IBM-compatible Tandy 1000 buisness computer and Deskmate application software have been packed at an unbeatable price! Tandy 1000 has 128K RAM, and easily expands to 640K. Unique Deskmate software provides you with six most-wanted programs: calander, text, worksheet, filer, telecom and mail. Tandy 1000 can also access the world's leading range of MS-DOS programs 25-1000 *IBM TM of International Business Machines

VM-2 Monochrome Monitor. 26-3211 ..... **349.95**

Monitor Sold Separately

The Superior Tandy 1000HD

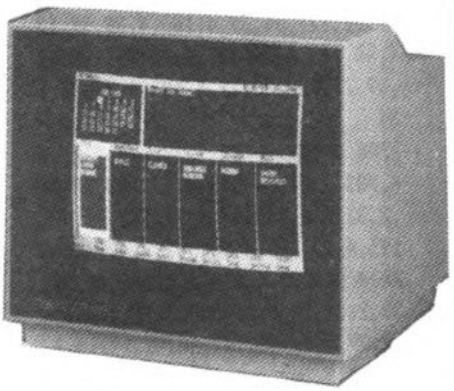
NEW!

$3999

Here's our new high-capacity business computer at a very competitive price! With 256K RAM and a built-in 10 megabyte hard disk drive, it has 10 million characters of fast-access storage for advanced business use! 25-1001

Color Monitor

- **High-Resolution**
- **Easy To Install**



$899

Enjoy a bright, clear full-color screen for your business diagrams, programs or graphics! Easily connects to your Tandy 1000 computer. 26-3212

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*

```
2700 .....160 11400 ....255
3900 .....194 13700 .....73
5700 ......35 16400 .....15
7700 ......89 END ......24
9300 ......55
```

Listing 1: LCLALARM

```
1000 '      LOCAL BURGLAR ALARM
1100 '      BY DENNIS H. WEIDE
1200 '      14201 MARQUETTE N.E.
1300 '      ALBUQUERQUE,NM  87123
1400 '      (C) 1985
1500 '
1600 CLS:DEFUSR1=&H7129:DEFUSR2=
&H7100
1700 CLEAR 2000,&H7000
1800 '
1900 '      CLOCK AND CALENDAR
2000 '
2100 FOR X=&H7100 TO &H7247
2200 READ A:POKE X,A
2300 NEXT X
2400 DATA 16,142,4,0,134,48,167,
160,16,140,4,31,47,246,134,47,18
3,4,2,183,4,5,134,58,183,4,29,13
4,32,16,142,4,8,167,160,16,140,4
2500 DATA 26,47,248,189,169,222,
182,1,90,177,112,255,39,2,32,1,5
7,183,112,255,129,40,46,1,57,182
,4,31,76,129,58,39,4,183,4,31,57
2600 DATA 134,48,183,4,31,182,4,
30,76,129,54,39,4,183,4,30,57,13
4,48,183,4,30,182,4,28,76,129,52
,39,21,129,58,39,4,183,4,28,57,1
34
2700 DATA 48,183,4,28,182,4,27,7
6,183,4,27,57,246,4,27,193,50,39
,2,32,226,16,142,4,27,142,48,48,
191,4,27,191,4,30,182,4,4,76,183
2800 DATA 4,4,190,4,0,140,49,50,
39,2,32,28,16,190,4,3,16,140,51,
50,39,2,32,16,142,48,49,191,4,0,
191,4,3,182,4,7,76,183,4,7,190,4
2900 DATA 3,140,50,57,39,22,140,
51,49,39,12,140,51,50,39,2,32,13
,189,114,43,32,8,189,114,14,32,3
,189,113,247,182,4,4,129,58,39,1
3000 DATA 57,134,48,183,4,4,182,
4,3,76,183,4,3,57,16,190,4,0,16,
140,48,50,39,1,57,142,48,49,191,
4,3,134,51,183,4,1,57,16,190,4,0
3100 DATA 16,140,48,52,39,19,16,
140,48,54,39,13,16,140,48,57,39,
7,16,140,49,49,39,1,57,142,48,49
,191,4,3,182,4,1,76,183,4,1,190
3200 DATA 4,0,140,48,58,39,1,57,
142,49,48,191,4,0,57
3300 EXEC &H7100
3400 FOR X=&H420 TO &H43F
3500 READ A:POKE X,A
3600 NEXT X
3700 DATA 32,32,32,32,32,32,32,3
2,32,32,2,21,18,7,12,1,18,32,1,1
2,1,18,13,32,32,32,32,32,32,32,3
2,32
3800 FOR X=&H7000 TO &H7049
3900 READ A:POKE X,A
4000 NEXT X
4100 EXEC &H7000
4200 DATA 190,1,104,175,140,12,4
8,140,12,191,1,104,134,57,167,14
0,239,57,85,85,2,52,23,13,111,38
,16,158,136,140,5,224,45,9,129,1
3
4300 DATA 39,11,140,5,255,39,6,5
3,23,110,157,255,225,166,140,224
,198,32,61,195,4,0,48,140,9,52,1
6,31,1,52,22,126,163,78,10,137,3
2,225
4400 '
4500 '      READ ALARM CONFIG
4600 '
4700 FOR X=4 TO 9
4800 READ AM$(X)
4900 NEXT X
5000 DATA NORTH SIDE ALARM,SOUTH
 SIDE ALARM,NORTH/SOUTH ALARM
5100 DATA EAST SIDE ALARM,WEST S
IDE ALARM,EAST/WEST ALARM
5200 DIM DR$(68)
5300 GOTO 9000
5400 '
5500 '
5600 '
5700 '      CLOCK UPDATE
5800 '
5900 D=USR1(0)
6000 '
6100 '      START OF ALARM SCAN
6200 '
6300 FOR X=1 TO 3
6400 A(X)=PEEK(&H15A+X)
6500 NEXT X:Y=4
6600 '
6700 '      COMPUTE ALARM VALUE
6800 '
6900 IF A(1))39 AND F3=0 THEN RE
TURN
7000 IF A(1))39 AND F3=1 THEN AR
$="ALARM DEACTIVATED":GOSUB 1490
0:F3=0
7100 IF A(1))9 AND A(1)(15 AND F
3=0 THEN AR$="ALARM ACTIVATED":G
OSUB 14900:F3=1
7200 IF A(1)=0 AND TJ=0 THEN AR$
="TAMPER ALARM":GOSUB 14900:T1=1
:AR$="BELL ON":GOSUB 14900:MOTOR
 ON
7300 IF A(1))9 AND A(1)(15 AND T
1=1 THEN AR$="TAMPER CLEAR":GOSU
B 14900:TJ=0
7400 FOR X=2 TO 3
7500 IF A(X)=0 THEN AF(Y)=0:AF(Y
+1)=0:AF(Y+2)=0:FF=FF+1:GOTO 790
0
7600 IF A(X))9 AND A(X)(15 AND A
F(Y)=0 THEN AR$=AM$(Y):GOSUB 149
00:AF(Y)=1:F1=1:GOTO 7900
7700 IF A(X))20 AND A(X)(28 AND
AF(Y+1)=0 THEN AR$=AM$(Y+1):GOSU
B 14900:AF(Y+1)=1:F1=1:GOTO 7900
7800 IF A(X))28 AND AF(Y+2)=0 TH
EN AR$=AM$(Y+2):GOSUB 14900:AF(Y
+2)=1:F1=1
7900 Y=Y+3:NEXT X
8000 IF F1=1 AND F2=0 THEN MOTOR
 ON:F2=1:AR$="BELL ON":GOSUB 149
00
8100 IF F1=1 AND FF=2 THEN AR$="
ALARM CLEARED":F2=0::GOSUB 14900
:F1=0
8200 FF=0
8300 RETURN
8400 '
8500 '      PASSWORD AND COMMANDS
8600 '
8700 '
8800 '      SET CLOCK
8900 '
9000 D=USR2(0):PRINT2480
9100 INPUT"ENTER TIME (HHMM)";A$
9200 INPUT"ENTER DATE (MMDDYY)";
B$
9300 POKE &H41F,VAL(MID$(A$,4,1)
)+48
9400 POKE&H41E,VAL(MID$(A$,3,1))
+48
9500 POKE&H41C,VAL(MID$(A$,2,1))
+48
9600 POKE&H41B,VAL(MID$(A$,1,1))
+48
9700 POKE&H0480,VAL(MID$(B$,1,1)
)+48
9800 POKE&H0481,VAL(MID$(B$,2,1)
)+48
9900 POKE&H0483,VAL(MID$(B$,3,1)
)+48
10000 POKE&H0484,VAL(MID$(B$,4,1
))+48
10100 POKE&H0486,VAL(MID$(B$,5,1
))+48
10200 POKE&H0487,VAL(MID$(B$,6,1
))+48
10300 PRINT
10400 AR$="SET CLOCK":GOSUB 1490
```

```
0
10500 GOTO 12000
10600 '
10700 '     COMMAND MODULE
10800 '
10900 CM$="":PRINTTAB(10)"PROGRA
M MENU"
11000 PRINTTAB(10)"------------"
11100 PRINT
11200 PRINTTAB(10)"1. SET TIME"
11300 PRINTTAB(10)"2. ALARM LOG"
11400 PRINTTAB(10)"3. PRINT LOG"
11500 PRINTTAB(10)"4. CLEAR LOG"
11600 PRINTTAB(10)"5. BELL OFF"
11700 PRINTTAB(10)"6. SIGN OFF"
11800 PRINT:PRINT
11900 PRINT"GO )"
12000 CM$=INKEY$
12100 GOSUB 5900
12200 IF CM$="" THEN 12000
12300 IF CM$=CHR$(13) THEN PRINT
:GOTO 10900
12400 CM=VAL(CM$):IF CM=3 THEN P
N=-2 ELSE PN=0
12500 IF CM)0 AND CM<7 THEN 1330
0
12600 CM$=""
12700 PRINT"INVALID COMMAND"
12800 GOTO 12000
12900 '
13000 '     START PROGRAM HERE
13100 '     USE "ON CM GOSUB"
13200 '
13300 ON CM GOSUB 9000,16000,160
00,20100,19400,17900
13400 IF F5=1 THEN F5=0:GOTO 109
00
13500 GOTO 12000
13600 '
13700 '     READ AND WRITE ALARM
S TO DISK
13800 '
13900 '
14000 '
14100 '     OPEN BUF #1
14200 '
14300 OPEN"D",#1,"ALARM.DAT",40
14400 FIELD#1,18 AS AL$,22 AS TI
$
14500 RETURN
14600 '
14700 '     WRITE ALARMS TO DISK
14800 '
14900 PRINT:PRINTTAB(6)"*****ALA
RM ENTRY*****":GOSUB 18300:PRINT
TAB(6) CL$:PRINTTAB(8) AR$
15000 GOSUB 14300
15100 LSET AL$=AR$
15200 LSET TI$=CL$
```

Listing 2: CALCLOCK

```
                    0010 *     CLOCK & CALENDAR PROGRAM
                    0011 *     BY DENNIS H. WEIDE
                    0012 *     1420 MARQUETTE N.E.
                    0013 *     ALBUQUERQUE,NM 87123
                    0014 *     (C) 1985
7100                0015       ORG     $7100
7100 108E 0400      0016 START LDY     #$0400  START OF SCREEN
7104 86   30        0017 ZEROS LDA     #$30    LOAD SCREEN
7106 A7   A0        0018       STA     ,Y+     WITH ZEROS
7108 108C 041F      0019       CMPY    #$041F  END OF SCREEN?
710C 2F   F6        0020       BLE     ZEROS
710E 86   2F        0021       LDA     #$2F    PRINT SLASHES
7110 B7   0402      0022       STA     $0402
7113 B7   0405      0023       STA     $0405
7116 86   3A        0024       LDA     #$3A    PRINT COLON
7118 B7   041D      0025       STA     $041D
711B 86   20        0026       LDA     #$20    LOAD BLANK SPACE
711D 108E 0408      0027       LDY     #$0408  START OF SCREEN BLANKS
7121 A7   A0        0028 BLANKS STA     ,Y+     BLANK OUT SCREEN
7123 108C 041A      0029       CMPY    #$041A
7127 2F   F8        0030       BLE     BLANKS
7129 BD   A9DE      0031 JOYSTK JSR     $A9DE   READ JOYSTKS
712C B6   015A      0032       LDA     $015A   COMPARE JOYSYK(0)
712F B1   70FF      0033       CMPA    $70FF   WITH LAST READING
7132 27   02        0034       BEQ     RET     RETURN IF EQUAL
7134 20   01        0035       BRA     STORE   IF NOT EQUAL
7136 39             0036 RET   RTS
7137 B7   70FF      0037 STORE STA     $70FF   STORE NEW
713A 81   28        0038       CMPA    #$28    JOYSTK(0) VALUE
713C 2E   01        0039       BGT     LOAD    BRANCH IF HIGH
713E 39             0040       RTS
713F B6   041F      0041 LOAD  LDA     $041F   INCREMENT MINUTE
7142 4C             0042       INCA            AND CHECK IF
7143 81   3A        0043       CMPA    #$3A    GREATER THAN 9
7145 27   04        0044       BEQ     MINTEN  BRANCH IF GREATER
7147 B7   041F      0045       STA     $041F   STORE MIN ON SCREEN
714A 39             0046       RTS
714B 86   30        0047 MINTEN LDA    #$30    STORE ZERO IN
714D B7   041F      0048       STA     $041F   MINUTE LOCATION
7150 B6   041E      0049       LDA     $041E   GET MINUTES
7153 4C             0050       INCA            TENS AND INCREMENT
7154 81   36        0051       CMPA    #$36    SIXTY MINUTES?
7156 27   04        0052       BEQ     HRUNIT  BRANCH IF YES
7158 B7   041E      0053       STA     $041E
715B 39             0054       RTS
715C 86   30        0055 HRUNIT LDA    #$30
715E B7   041E      0056       STA     $041E   SET MINUTES TO ZERO
7161 B6   041C      0057       LDA     $041C
7164 4C             0058       INCA            INCREMENT HOURS TENS
7165 81   34        0059       CMPA    #$34    HRUNIT=4?
7167 27   15        0060       BEQ     CHECK   BRANCH IF YES
7169 81   3A        0061 RETURN CMPA    #$3A    HRUNIT>9?
716B 27   04        0062       BEQ     HRTEN   BRANCH IF YES
716D B7   041C      0063       STA     $041C   PRINT NEW HRUNIT
7170 39             0064       RTS
7171 86   30        0065 HRTEN LDA     #$30    SET HRUNIT
7173 B7   041C      0066       STA     $041C   TO ZERO
7176 B6   041B      0067       LDA     $041B   ADD ONE
7179 4C             0068       INCA            TO HRTEN
717A B7   041B      0069       STA     $041B   PRINT NEW HRTEN
717D 39             0070       RTS
717E F6   041B      0071 CHECK LDB     $041B   HOURS=24?
7181 C1   32        0072       CMPB    #$32
7183 27   02        0073       BEQ     SETDAY  BRANCH IF YES
7185 20   E2        0074       BRA     RETURN
7187 108E 041B      0075 SETDAY LDY    #$041B  START OF TIME
718B 8E   3030      0076       LDX     #$3030  SET TIME
```

```
15300 PUT#1,LOF(1)+1
15400 CLOSE#1
15500 PRINT:PRINT"GO )"
15600 RETURN
15700 '
15800 '     READ ALARMS FROM DIS
K
15900 '
16000 GOSUB 14300
16100 PRINT#PN,STRING$(32,"-")
16200 FOR X=1 TO LOF(1)
16300 GET#1,X
16400 FOR ZZ=1 TO 500:NEXT ZZ
16500 PRINT#PN
16600 PRINT#PN,TAB(6)"******LOG
ENTRY******"
16700 D=USR1(0)
16800 PRINT#PN,TAB(6) TI$
16900 D=USR1(0)
17000 PRINT#PN,TAB((32-LEN(AL$))
/2) AL$
17100 D=USR1(0)
17200 NEXT X
17300 CLOSE#1
17400 PRINT:PRINT"GO )"
17500 RETURN
17600 '
17700 '     END PGM
17800 '
17900 STOP:RETURN
18000 '
18100 '     READ TIME
18200 '
18300 CL$=""
18400 FOR X=&H400 TO &H40F
18500 CL$=CL$+CHR$(PEEK(X))
18600 NEXT X
18700 FOR X=&H41B TO &H41F
18800 CL$=CL$+CHR$(PEEK(X))
18900 NEXT X
19000 RETURN
19100 '
19200 '     SILENCE ALARM
19300 '
19400 MOTOR OFF
19500 AR$="BELL OFF"
19600 GOSUB 14900
19700 RETURN
19800 '
19900 '     CLEAR LOG
20000 '
20100 PRINT:PRINT"ARE YOU SURE (
Y/N)"
20200 IP$=INKEY$:GOSUB 5900
20300 IF IP$="" THEN 20200
20400 IF IP$<>"Y" THEN PRINT:PRI
NT"GO )":RETURN
20500 KILL"ALARM.DAT":AR$="LOG C
LEARED":GOTO 14900
```

```
718E BF   041B      0077D        STX    $041B   TO 00:00
7191 BF   041E      0078D        STX    $041E
7194 B6   0404      0079D        LDA    $0404   ADD
7197 4C             0080D        INCA           ONE
7198 B7   0404      0081D        STA    $0404   DAY
719B BE   0400      0082D        LDX    $0400   GET MONTH
719E 8C   3132      0083D        CMPX   #$3132  IS MONTH=12?
71A1 27   02        0084D        BEQ    ENDYR   BRANCH IF YES
71A3 20   1C        0085D        BRA    DAY29
71A5 10BE 0403      0086D ENDYR  LDY    $0403   GET DAY
71A9 108C 3332      0087D        CMPY   #$3332  ID DAY=32?
71AD 27   02        0088D        BEQ    NEWYR   BRANCH IF YES
71AF 20   10        0089D        BRA    DAY29
71B1 8E   3031      0090D NEWYR  LDX    #$3031  SET NEW
71B4 BF   0400      0091D        STX    $0400   DATE 01/01
71B7 BF   0403      0092D        STX    $0403
71BA B6   0407      0093D        LDA    $0407   GET YEAR
71BD 4C             0094D        INCA           ADD 1 YEAR
71BE B7   0407      0095D        STA    $0407   PRINT YEAR
71C1 BE   0403      0096D DAY29  LDX    $0403   IS DAY
71C4 8C   3239      0097D        CMPX   #$3239  29?
71C7 27   16        0098D        BEQ    MOTEST  BRANCH IF YES
71C9 8C   3331      0099D        CMPX   #$3331  IS DAY=31?
71CC 27   0C        0100D        BEQ    THIRTY  BRANCH IF YES
71CE 8C   3332      0101D        CMPX   #$3332  IS DAY=32?
71D1 27   02        0102D        BEQ    LSTDAY  BRANCH IF EQUAL
71D3 20   0D        0103D        BRA    COMP
71D5 BD   722B      0104D LSTDAY JSR    RESET
71D8 20   08        0105D        BRA    COMP
71DA BD   720E      0106D THIRTY JSR    MSHORT
71DD 20   03        0107D        BRA    COMP
71DF BD   71F7      0108D MOTEST JSR    FEBCHK
71E2 B6   0404      0109D COMP   LDA    $0404
71E5 81   3A        0110D        CMPA   #$3A    IS DAY UNIT>9?
71E7 27   01        0111D        BEQ    DAYTEN  BRANCH IF YES
71E9 39             0112D        RTS
71EA 86   30        0113D DAYTEN LDA    #$30    SET DAY UNIT
71EC B7   0404      0114D        STA    $0404   TO ZERO
71EF B6   0403      0115D        LDA    $0403   ADD ONE
71F2 4C             0116D        INCA           TO
71F3 B7   0403      0117D        STA    $0403   DAY TEN
71F6 39             0118D        RTS
71F7 10BE 0400      0119D FEBCHK LDY    $0400   IS MONTH
71FB 108C 3032      0120D        CMPY   #$3032  EQUAL TO 02?
71FF 27   01        0121D        BEQ    FEBSET  BRANCH IF YES
7201 39             0122D        RTS
7202 8E   3031      0123D FEBSET LDX    #$3031  SET DAY
7205 BF   0403      0124D        STX    $0403   TO ZERO
7208 86   33        0125D        LDA    #$33    SET MONTH
720A B7   0401      0126D        STA    $0401   TO 03
720D 39             0127D        RTS
720E 10BE 0400      0128D MSHORT LDY    $0400   IS MONTH
7212 108C 3034      0129D        CMPY   #$3034  EQUAL TO 04?
7216 27   13        0130D        BEQ    RESET   BRANCH IF YES
7218 108C 3036      0131D        CMPY   #$3036  MONTH 06?
721C 27   0D        0132D        BEQ    RESET   BRANCH IF YES
721E 108C 3039      0133D        CMPY   #$3039  MONTH 09?
7222 27   07        0134D        BEQ    RESET   BRANCH IF YES
7224 108C 3131      0135D        CMPY   #$3131  MONTH 11?
7228 27   01        0136D        BEQ    RESET   BRANCH IF YES
722A 39             0137D        RTS
722B 8E   3031      0138D RESET  LDX    #$3031  SET DAY
722E BF   0403      0139D        STX    $0403   EQUAL TO ZERO
7231 B6   0401      0140D        LDA    $0401   ADD ONE
7234 4C             0141D        INCA           TO
7235 B7   0401      0142D        STA    $0401   MONTH
7238 BE   0400      0143D        LDX    $0400   GET MONTH
723B 8C   303A      0144D        CMPX   #$303A  IS MONTH>9?
723E 27   01        0145D        BEQ    MONTEN  BRANCH IF YES
7240 39             0146D        RTS
7241 8E   3130      0147D MONTEN LDX    #$3130  SET MONTH
7244 BF   0400      0148D        STX    $0400   TO TEN
7247 39             0149D        RTS
          0000      0150D        END
00000 TOTAL ERRORS
```

A Beginner's Hardware Course

Part 1

By Tony DiStefano

This being the Beginners issue, I will start a multi-part article on how a computer works, starting from "simple theory" to "how to build one of my projects." This month, we will begin with basic concepts: what is a bit, what does digital mean, what is analog, how does it differ from digital, and a look at a different numbering system.

The dictionary meaning of analog is "proportionate." When speaking, you can speak loud or low. Light can be dark or bright, or any shade in-between. Radio waves and TV pictures are all said to be analog signals. These are examples of analog wave shapes — continuously changing. When we talk about a digital system, there are no shades or continuous motion. There are only two states in a digital signal: ON or OFF. There is no in-between. This is the core of computing. Everything your computer does is accomplished using these two states. OK, let's expand on these states.

First, there is ON. It is also known as "high" (Hi or 'H'), "plus," "one" (or '1'), "mark," "voltage" and many others. The two terms I use most often are Hi and '1'; these are the terms I will use throughout these articles. In most microcomputers, the operating voltage for the hardware is five volts. Virtually all the mircocomputer and support chips work with five volts. It is pretty much a norm. Given this, a Hi measures about five volts on a voltage meter, but 4.5 volts is also considered Hi. There are limits to how low the voltage can be before it is considered invalid. In fact, any voltage greater than two volts is considered to be a logic level Hi or '1'.

Next is the OFF state. It, too, has many names: "low" (Lo or 'L'), "minus," "zero" (or '0'), "space" and "ground," just to name a few. To keep consistent, I will use Lo and '0' to mean OFF. A low state is considered to have zero volts and when measured with a voltage meter, nothing registers. Under certain conditions, a small voltage can be present. Any voltage below .8 volts is considered to be a logic level '0'. Any voltage greater than .8 volts or less than two volts is not a valid logic state and results are, at least, unpredictable.

Now we know about the highs and lows of digital operation. The next step is a "bit." A bit is one piece of logic information. It has, as we now know, two states, either Lo or Hi. It's also known as a binary digit, binary meaning two. The two states are:

State 0 = 0 (Low)
State 1 = 1 (High)

But, just two pieces of information is not very much to work with. If we use two bits side by side, and considered every combination of 0's and 1's, there are four separate combinations.

State 0 = 00
State 1 = 01
State 2 = 10
State 3 = 11

If you have three bits side by side, there are eight different combinations.

State 0 = 000
State 1 = 001
State 2 = 010
State 3 = 011
State 4 = 100
State 5 = 101
State 6 = 110
State 7 = 111

Can you see a pattern start to develop? Every time one more bit is added, you double the amount of different combinations possible. This is known as Base 2 or binary numbering system. Most of us are more familiar with Base 10 or decimal numbering system. In short, Base 10 numbers, unlike Base 2 numbers, have 10 different combinations per digit.

State 0 = 0
State 1 = 1
State 2 = 2
State 3 = 3
State 4 = 4
State 5 = 5
State 6 = 6
State 7 = 7
State 8 = 8
State 9 = 9

I am sure you recognize these numbers. Once the top of the number ladder is reached, you add another digit to the left of it. Each number added raises the value of that digit in the number by a factor of 10.

```
                3   2   1   5
               /    |   |    \
10³ - 1Ø E 3   1Ø E 2   1Ø E 1   1Ø E Ø - 10⁰
       or       or      or       or
       |        |       |        |
       3ØØØ  +  2ØØ  +  1Ø  +  5   = 3215
```

When large numbers are to be represented, there are more digits. Each new digit added means adding another power of 10. Numbers ranging in the millions require only seven digits in Base 10 numbers, but require many digits in Base 2 since every added digit is only to the power of 2.

```
                1   Ø   1   1   Ø
               /    |   |    \
2³ - 2 E 4   2 E 3   2 E 2   2 E 1   2 E Ø - 2⁰
       or       or      or       or       or
       |        |       |        |        |
       16  +    Ø   +   4   +    2   +    Ø   =  22
```

You can see that a Base 2 number adds up to a lot less than Base 10. There is yet a better-suited numbering system for computers, but first let's look at a bit more (ha, ha).

The Color Computer (all versions) has an eight-bit CPU. That means all data, program code and characters are stored in eight-bit values. These groups are better known as bytes. A byte can hold any value from 00000000 (Base 2) to 11111111, or in decimal, from zero to 255. If you convert 1111111 to decimal, it works out to 255. Each byte in the CoCo is one memory location. A byte can hold one ASCII character, one piece of data or one machine language code. We'll look more at memory later on.

In the computer environment there is another numbering system. It is most used and is called the hexadecimal numbering system, or Hex for short. The Hex system, as the name implies, is a Base 16 number. This means there must be 16 symbols before the carry over to the next digit. In Hex, the symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Just as the next digit after '3' is '4' (3+1), the next digit after '9' (9+1) is 'A'. Remember that A, B, C, D, E and F are digits, not letters, in the hexadecimal system. The following table exemplifies the different numbering systems described.

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

As you can see from the table, the Hex numbering system is the most efficient because of its highest base number. The decimal system takes two characters to the one character needed by Hex; binary takes four characters. Since the CoCo has an eight-bit bus (a memory byte), you can represent a memory location with eight bits (11111111) or three decimal digits (255) or a two-digit Hex number ($FF). From now on we will use all three numbering systems, which ever happens to be the best for the occasion. When using Hex, however, I will put the character '$' in front of it. Some like to put an 'h' at the rear of the number — both are correct, I just prefer the dollar sign.

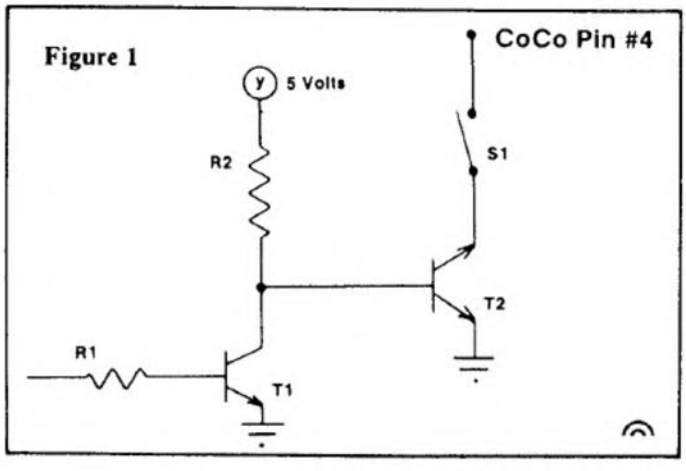
Understanding the Hex and binary numbering systems and what they stand for in a computer is the base from where your knowledge of the CoCo will grow. I will not cover adding and subtracting or conversion from one base to another in this article, but if you want to learn more on numbering systems, your local library should have numbering systems in the math section.

One of the command functions built into Extended BASIC is HEX$, pronounced "Hex string." This command transfers a normal decimal value into a string variable in hexadecimal format. The syntax for this command is HEX$(X) where 'X' can be a direct value or any numeric expression. As an example, to get the Hex equivalent of the decimal value 207, type PRINT HEX$(207) and ENTER. This prints CF and is the Hex equivalent of 207. A very handy command to have.

On the other hand, how would we change a Hex value into a decimal value? Extended BASIC comes to the rescue again, for it has another function that allows entry of Hex values, the &H sign. Anytime you need to enter a value in Hex, use the &H in front of the value. For instance, if you have a line that sets the value of 'X' to the Hex value FF, you can calculate $FF to a decimal value or you can enter it as 100 X = &HFF. Another use of the function &H is to convert a Hex number to decimal. Since all numbers printed are done in decimal, to convert a Hex number to decimal all you have to do is PRINT &HX and ENTER, where 'X' is any Hex number and the result is printed in decimal on the screen. If you are to substitute the letter 'O' instead of 'H', all values will be in octal, or Base 8.

* * *

I got a letter from a reader just this week. He pointed out a problem with "Turn of the Screw" in the November 1984 issue. There is mention of a switch in the text, but no such switch existed in the diagram. Figure 1 shows where this switch goes.



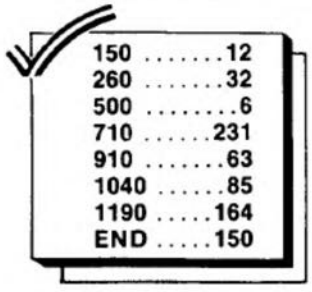
Presenting The Star Of The Show — Screen Alternatives

By Bill Bernico

Tired of the same ol' beginnings to your programs? You know — CLS:PRINT @ 100, "MY PROGRAM". Well, now you can spruce up the title page and discover new ways to clear your screen instead of using CLS.

Upon running *Screen*, you are presented with a menu containing six choices. Selection numbers 1, 3, 4 and 5 show alternative ways to present your title page, and numbers 2 and 6 demonstrate new ways to clear the screen. Each selection is in itself a stand-alone program. That is, you can take lines 300-510, for example, and adapt them to your own program. These lines make up selection number 1. Selection number 2 can be "pulled out" of the main program by using lines 530-590, and so on.

I'm always working on new ways to do old tasks, but these are my six favorites so far. Feel free to use them or improve them in any way you like. Make your program stand out and give it the recognition it deserves.



The listing: SCREEN

```
10 'SCREEN PRESENTATIONS
20 'BY BILL BERNICO
30 '708 MICHIGAN AVE.
40 'SHEBOYGAN, WI 53081
50 '(414) 459-7350
60 '
70 PMODE 4,1:PCLS5:SCREEN 1,1:CO
LOR 0,1
80 DRAW"BM40,28S2;U12H4L28G4D28F
26D4G4L12H4U4L8D12F4R32E4U12H26U
8E4R8F4R8":PAINT(39,27),8,8:SOUN
D 100,2
90 DRAW"BM76,28S2;U12H4L28G4D57F
4R28E4U16L8D8G4L12H4U40E4R12F4D4
R8":PAINT(74,45),8,8:SOUND 120,2
100 DRAW"BM92,28S2;D65R12U25F25R
16H25R12E8U24H8L40":DRAW"BM98,26
;D20R20E4U12H4L20":PAINT(93,21),
8,8:SOUND 140,2
110 DRAW"BM132,28S2;D65R32U12L16
U14R8U14L8U14R16U12L32":PAINT(13
3,21),8,8:SOUND 160,2
120 DRAW"BM162,28;D65R32U12L16U1
4R8U14L8U14R16U12L32":PAINT(163,
21),8,8:SOUND 180,2
130 DRAW"BM192,28S2;D65R16U36F36
R14U65L16D39H38L18":PAINT(193,21
),8,8:SOUND 200,2
140 DRAW"BM28,98S2;D48R25E7U14H7
L25BR40F23BE23G46BE46BR40BU38D66
R25E9U19H9L25BR25E9U19H9L25BR64D
66BU66BR34D66R43BU66BR34D66R43BL
219BD20D66R25E9U19H9L25BR25E9U19
H9L25BR75L25D66R25BU33BL10L15BR4
0BD33U66R25F9D18G9L25F33BR20U66F
46D20U66BR20D66
150 DRAW"BR53BU66BD8U8L33D66R33U
8BD8BR20U66R33D66L33
160 FOR X=1 TO 1500:NEXT X
170 C$=CHR$(128):V$=CHR$(159):B$
=CHR$(175):N$=CHR$(191):M$=CHR$(
207):K$=CHR$(239):L$=CHR$(255)
180 CLS0:PRINT@33,"choose";C$;"f
rom";C$;"these";C$;"design";C$;"
ideas";C$;
190 PRINT@131,V$;"(1) BORDER FLA
SH ";V$;
200 PRINT@164,B$;"(2) DISAPPEARI
NG ";B$;
210 PRINT@197,N$;"(3) FLASH ORAN
GE ";N$;
220 PRINT@230,M$;"(4) COLOR TITL
ES ";M$;
230 PRINT@263,L$;"(5) REPLACEMEN
TS ";L$;
240 PRINT@296,K$;"(6) ROLLING BA
CK ";K$;
250 PRINT@329,M$;"**************
***";M$;
260 PRINT@362,V$;"choose 1-6 OR
eND";V$;
270 A$=INKEY$:IF A$="E"THEN CLS:
END
280 A=VAL(A$):ON A GOTO 300,530,
610,830,1060,1320
290 GOTO 270
300 CLS0:PRINT STRING$(32,201);
310 FOR X=63 TO 510 STEP 32
320 PRINT@X,CHR$(201);:NEXT X
330 PRINT@480,STRING$(31,201);
340 POKE 1535,201
```

```
350 FOR X=448 TO 32 STEP-32
360 PRINT@X,CHR$(201);:NEXT X
370 FOR Y=1 TO 3
380 POKE 359,126:SOUND 191,2
390 PRINT@74,"bill";C$;"bernico"
;
400 PRINT@140,"presents";
410 PRINT@199,"another";C$;"amaz
ingly";
420 PRINT@268,"simple";
430 PRINT@359,"SCREEN";C$;"PRESE
NTATION";
440 PRINT@428,"PROGRAM";
450 FOR X=1 TO 350:NEXT X
460 POKE 359,57:SCREEN 0,1
470 SOUND 159,2
480 FOR X=1 TO 350:NEXT X
490 NEXT
500 POKE 359,126:CLS
510 FOR X=1 TO 300:NEXT X
520 GOTO 170
530 CLS:PRINT@100,"TRY THIS METH
OD FOR              CLEARING THE
SCREEN.             IT'S CERTAINL
Y OUT               OF THE ORDINA
RY AND              IT HOLDS YOUR
 ATTEN-             TION BETTER T
HAN A               PLAIN OLD 'CL
S' COM-             MAND.
540 PRINT@392,"HIT ANY KEY
550 EXEC 44539
560 FOR Y=100 TO 402 STEP 2:PRIN
T@Y,CHR$(143);
570 PRINT@Y,CHR$(32);:NEXT Y
580 FOR O=1 TO 700:NEXT O
590 FOR P=101 TO 464 STEP 2:PRIN
T@P,CHR$(143);:NEXT P
600 GOTO 170
610 FOR Y=1 TO 3
620 FOR X=1 TO 1000:NEXT X:CLS0
630 SOUND 140,1
640 PRINT@3,STRING$(7,255);
650 PRINT@69,STRING$(9,255);
660 PRINT@135,STRING$(6,255);
670 PRINT@201,STRING$(6,255);
680 PRINT@267,STRING$(12,255);
690 PRINT@333,STRING$(2,255);
700 PRINT@399,STRING$(4,255);C$;
STRING$(7,255);
710 FORX=1 TO 500:NEXT X:CLS0
720 SOUND 191,1
730 PRINT@3,"another";
740 PRINT@69,"amazingly";
750 PRINT@135,"simple";
760 PRINT@201,"screen";
770 PRINT@267,"presentation";
780 PRINT@333,"by";
790 PRINT@399,"bill";;C$;"bernic
o";
800 NEXT Y
810 FOR X=1 TO 500:NEXT X
820 GOTO 170
830 CLS0
840 FOR X=3 TO 13:SET(X,4,1):NEX
T X
850 FOR X=4 TO 11:SET(8,X,1):SET
(7,X,1):SET(18,X,2):SET(19,X,2):
SET(28,X,3):SET(29,X,3):SET(38,X
,4):SET(39,X,4):SET(50,X,5):SET(
51,X,5):NEXT X
860 FOR X=24 TO 33:SET(X,4,3):NE
XT X
870 FOR X=39 TO 45:SET(X,11,4):N
EXT X
880 FOR X=51 TO 57:SET(X,11,5):S
ET(X,4,5):NEXT X
890 FOR X=51 TO 55:SET(X,8,5):NE
XT X
900 FOR X=16 TO 23:SET(12,X,5):S
ET(13,X,5):SET(48,X,2):SET(49,X,
2):NEXT X
910 FOR X=14 TO 17:SET(X,16,5):S
ET(X,20,5):NEXT X
920 FOR X=17 TO 19:SET(18,X,5):S
ET(19,X,5):NEXT X
930 FOR X=17 TO 23:SET(24,X,7):S
ET(25,X,7):SET(30,X,7):SET(31,X,
7):NEXT X
940 FOR X=26 TO 29:SET(X,16,7):S
ET(X,20,7):NEXT X
950 FOR X=17 TO 22:SET(36,X,8):S
ET(37,X,8):NEXT X
960 FOR X=38 TO 41:SET(X,16,8):S
ET(X,23,8):NEXT X
970 FOR X=40 TO 43:SET(X,21,8):N
EXT X:SET(42,17,8):SET(43,17,8):
SET(42,22,8):SET(43,22,8)
980 FOR X=50 TO 55:SET(X,16,2):S
ET(X,23,2):NEXT X
990 FOR X=50 TO 53:SET(X,20,2):N
EXT X
1000 Y=29:FOR X=1 TO 30
1010 SET(6,Y,1):SET(12,Y,2):SET(
18,Y,3):SET(24,Y,4):SET(30,Y,5):
SET(36,Y,6):SET(42,Y,7):SET(48,Y
,8):SET(54,Y,1):SET(60,Y,2)
1020 EXEC 43345
1030 RESET(6,Y):RESET(12,Y):RESE
T(18,Y):RESET(24,Y):RESET(30,Y):
RESET(36,Y):RESET(42,Y):RESET(48
,Y):RESET(54,Y):RESET(60,Y)
1040 NEXT X
1050 GOTO 170
1060 CLS:PRINT@131,STRING$(4,C$)
;" ";STRING$(2,C$);" ";STRING$(6
,C$);" ";STRING$(6,C$);" ";STRIN
G$(4,C$)
1070 GOSUB 1310
1080 PRINT@131,"this";B$;"is";B$
;"design";B$;"number";B$;"five
1090 GOSUB 1310
1100 PRINT@195,STRING$(3,C$);" "
;STRING$(6,C$);" ";STRING$(6,C$)
;" ";STRING$(2,C$);" ";STRING$(4
,C$)
1110 GOSUB 1310
1120 PRINT@195,"the";B$;"visual"
;B$;"effect";B$;"of";B$;"this";B
$;
1130 GOSUB 1310
1140 PRINT@259,STRING$(7,C$);" "
;STRING$(2,C$);" ";STRING$(4,C$)
;" ";STRING$(6,C$);" ";STRING$(3
,C$)
1150 GOSUB 1310
1160 PRINT@259,"pattern";B$;"is"
;B$;"very";B$;"unique";B$;"for
1170 GOSUB 1310
1180 PRINT@323,STRING$(6,C$);" "
;STRING$(12,C$);" ";STRING$(4,C$
)
1190 GOSUB 1310
1200 PRINT@323,"screen";B$;"pres
entation";B$;"uses";B$;B$;
1210 GOSUB 1310
1220 PRINT@99,STRING$(26,B$);
1230 FOR X=125 TO 380 STEP 32:PR
INT@X,B$;:NEXT X
1240 FOR X=381 TO 354 STEP-1:PRI
NT@X,B$;:NEXT X
1250 FOR X=354 TO 98 STEP-32:PRI
NT@X,B$;:NEXT X
1260 PRINT@163,STRING$(27,B$);
1270 PRINT@227,STRING$(27,B$);
1280 PRINT@291,STRING$(27,B$);
1290 GOSUB 1310
1300 GOTO 170
1310 FOR X=1 TO 800:NEXT X:EXEC
43345:RETURN
1320 CLS:PRINT@69,"THIS PROGRAM
WILL HELP           YOU TO SPRUCE
 UP YOUR            OWN PROGRAMS
BY GIVING           YOU A FLASHY
SCREEN              PRESENTATION
TO REPLACE          THE DULL, DRA
B, ORDINARY         TITLE PAGE.
1330 PRINT@293,"YOU CAN CHOOSE F
ROM SIX             EYE-APPEALING PA
TTERNS              TO BE YOUR TITLE
 PAGE.
1340 PRINT@421,"PRESS enter TO
                    RETURN TO MENU
1350 EXEC 44539
1360 FOR Q=475 TO 69 STEP -1
1370 PRINT@ Q, CHR$(32)
1380 EXEC 43345:NEXT Q
1390 GOTO 170
```

CoBBS:
Setting Up Various Files To Make The System Operate

By Richard Duncan

Welcome back! For the past two months we have been looking at the bulletin board software and what makes it tick. This month we will use the editors required to set up the various files that make *CoBBS* operate.

System Control File Editor

The System Control File Editor (*SCF/EDI*) is the first editor to use in setting up the system. This editor creates a file that tells the system certain functions. The editor file is stored on disk in an open area of the directory file so no additional disk space is used. Upon loading and executing the editor, you are asked a group of questions.

1) LOG-ON TYPE? This specifies how far you want a new caller to get on the system.

- Type 1 — No access unless the user has an entry in the user-log.
- Type 2 — User may register if not in the userlog but will not be able to access the main system.
- Type 3 — Operates like Type 2. User is shown *NEW USER/TXT* if available.
- Type 4 — User may register if not in the userlog and have access to the main board.

2) MENU TO ENTER IN? This is the menu that will be loaded when a user gets past logon. This menu number is initially set in the user's log entry when first logging on. This may later be changed by the user through the Menu Control Editor.

3) NEW USER TIME-OUT? This sets the time a new user is allowed to use the system. The value may be from zero to 255. Time is figured as five minutes times the value, so four, for instance, would be 20 minutes.

4) NEW USER PRIVILEGE? Sets the privilege level of a new user. May be any value from zero to 255.

5) NEW USER P1 FLAGS? Sets any flags desired for a new user. Reply with the actual set of flags, such as "00000000." Only a '0' and/or '1' is allowed.

6) NEW USER P2 FLAGS? Same as P1 flags, except this is for the last eight of 16 flags.

7) CHECK FOR MESSAGES? If answered "yes," the system checks to see if there are any messages on the system for that user after logging in and before loading the main board routine. If answered "no," then this routine is skipped and the system goes directly to the main board routine after logging in the user.

8) USER PROMPT? Sets the new user's prompting level. Refer to the information on the Menu Control Editor for the different prompts.

9) TRACE USER'S OPERATION? The tracer option maintains a log of a user's operation while on the system. The file to which the information is printed is determined by this flag. The options are:

- 0 — No trace
- 1 — Cassette (reserved)
- 2 — Disk drive
- 3 — Printer (available if using the Pak)

10) DRIVE FOR HDR/SYS? Specify which drive the message header is on.

11) DRIVE FOR MSG/SYS? Specify which drive the message text is on.

12) DRIVE FOR USERL/SYS? Which drive for the userlog.

13) DRIVE FOR MENU/SYS? Drive for the board's menu file.

14) DRIVE FOR TRACER/SYS? Drive for the system's trace file if used.

15) DRIVE FOR REGISTER/SYS? Drive for registration file.

After answering all questions, the system shows all the selections. If they are correct, the file is saved to the disk's directory. This will be the system disk that *must* be in Drive 0. Any disk used for the system in Drive 0 must have a control file on it or the program will not run.

To save a file, type YES at the first prompt, "Ready to Save?" Insert the disk that will be the Drive 0 system disk. At the next prompt, "Ready to Save?" type Y and ENTER. You now have your system parameters for the system. These can easily be changed by running *SCF/EDI* again.

System Menu File Editor

The next step is to use the System Menu File Editor (*SMF/EDI*) to set up the various menus. Note: To use any editor other than *SCF/EDI* you must have booted up the system as far as the start-up file. Error trap and *COTERM* must be in memory. Also, before running any of the editors locally, type POKE 4658,0 before running or the system will try to reboot.

The menu control file contains all the information needed to present a user with a choice of commands. Each menu has a number, and the number may be from zero to 255. When a menu is printed, the number of that menu appears with its name just before the various commands. The number of the menu appears in asterisks (*). Using this editor the system is told not only which

commands should be available, but what the privilege requirement is and if a user flag must be set to see this menu or command.

To be safe, there should always be a menu zero, with zero privilege and *no* flags set. This prevents a user from having difficulty upon certain system errors or errors in setting up the new user privileges.

Upon loading and running the program the following menu appears:

1) Create a new menu
2) Edit existing menu
3) Exit editing
4) Return to BBS

1) Create a new menu — Choosing this option allows the SysOp to create a menu. The system will ask if you desire to kill the old menu file. If answered "yes," the complete menu file is deleted with all menus. After answering the prompt the editor's menu appears.

2) Edit existing menu — This function allows the SysOp to edit any menu in the file. If an improper menu number is given, then the first menu reappears.

3) Exit editing — Exits *SMF/EDI* and goes into BASIC.

4) Return to BBS — Exits *SMF/EDI*, loads and runs the main board system.

After passing the first menu the SysOp is presented with the editor menu. The following menu appears:

#(menu) (menu name)
COMMANDS: (number)
1) ADD A COMMAND
2) CHANGE A COMMAND
3) LIST MENU
4) ADD/CHANGE BOARD MENU
5) LIST/EDIT HEADER
6) DELETE A COMMAND
7) EXIT AND WRITE
8) EXIT

The first step is actually Function 4, but we will look at them in the order listed.

1) ADD A COMMAND — After choosing this, the system asks for INSERT AFTER:. Pressing 'L' adds the command to the end of the list, or the SysOp may specify which command the new one will come after. The system asks for the needed information of each command.

TEXT: Enter command text up to 16 characters
KEY: Enter key the system looks for to execute the command
PRIV: Privilege level required to see this command
P1: First eight of 16 flags. Enter either '0' or '1' only
P2: Same as P1, except second eight flags
TYPE: Type of command
DATA: Optional data required by some commands

2) CHANGE A COMMAND — The function allows editing of a command. You are asked which command number you wish to edit. While the various command information is presented, pressing ENTER retains the original entry while typing new information replaces the original.

3) LIST MENU — Allows the listing of each command or the text as it appears on the system under full text prompting. While looking at the various full command listing, use a 'Q' to exit the listing, or any key to continue to the next one.

4) ADD/CHANGE BOARD MENU — Option 4 determines what menu number is entered and whether it is a new one or change menu number and retains all commands. The privilege level and flags set those values for that particular menu. A user must meet or exceed these specifications to be able to access it. When asked if the board is postable, the system sets a flag to tell the board program whether or not the menu may be listed as one that may have a message posted to it in reply or as a new message. This prevents the possiblilty of a user who has access to that menu from posting a message to it if there is no read command available for the number.

5) LIST/EDIT HEADER — This function lifts the menu header text and also allows modification of that text.

6) DELETE A COMMAND — Deletes a command from the list. The command is shown and verification requested before deleting the command.

7) EXIT AND WRITE — Exits the editor and writes the new or modified menu to the menu control file. Returns to the first menu.

8) EXIT — Exits the editor and returns to the first editor menu. No data in the menu file is modified.

Use command #4 first when setting up a menu, then use the #1 command to add whatever commands you want on that menu. Pressing 'L' adds the command to the end of the list. Determine which command you want on each menu. After pressing the ADD function, you are asked the text to display, the key character to look for, privilege, flags to set (if any), the type of command and then the optional data, which may or may not be needed depending on the particular type of command. The other commands are for modifying, deleting, saving, etc. A program to set up a simple menu file automatically is available on my BBS.

Userlog Editor

The System Userlog Editor (*SUL/EDI*) is used to modify a user's privilege or other pertinent information. Upon loading and running the editor the following menu appears:

USERS: (NUMBER)

<S>EARCH	<G>O TO USER #
<N>EXT	<L>AST ENTRY
<M>ODIFY FIELD	<R>EGISTER
<A>CTIVATE	<D>ELETE
<P>RINT LISTING	<B>ACKUP
<E>NTER USER	<U>SER CLASS
<Q>UIT	<H>ARD COPY

COMMAND>

Note: A user's information must be initially printed on the screen before any modifications may be made.

<S>EARCH — Searches for a user by string comparison. After pressing the 'S' the system prompts for the test string. The test string may be any target in the name or from. Use SHIFT-CLEAR to separate name and from in text string.

<G>O TO USER # — Jumps to user specified. The user number changes when the userlog is backed up and there are deleted entries.

<N>EXT — Goes to the next user in the log.

<L>AST ENTRY — Goes to the previous user in the log.

<M>ODIFY FIELD — Used to change one of the user listings information. Some listings may not be modified. After choosing 'M' the system asks for field to be modified (see the section on field modification).

<R>EGISTER — Sets the registration flag where the user is classified as either a New, Probationary or Registered user. This has no bearing on the operation of the system for that user as all those controls are in the command and user privilege information, but the user may be shown a text file at logon that only new users will see.

<A>CTIVATE — "Undeletes" a user from the log if deleted by accident.

<D>ELETE — Flags a user to be deleted from the log the next time a backup is performed.

<P>RINT LISTING — Prints the user's record on the screen and shows all pertinent information in his file.

<B>ACKUP — Backup userlog file. Copies entries over to the file *USERL/BAK*. While copying, the system will not copy over any entries that are flagged to be deleted. The system asks if you desire to delete and copy the new file over to the system's userlog file, *USERL/SYS*.

<E>NTER USER — Allows entering a user into the log without that user having to call the system and register.

<U>SER CLASS — Not used at this time.

<Q>UIT — Exit the editor.

<H>ARD COPY — Prints the full userlog entries to the printer.

A user's information must be initially printed before any modification can be done. This is normally done except when first running the editor.

When choosing the <M>ODIFY option, the system asks for which field to be modified. A user's entry is formatted as follows:

USER # (number of record)
A) user name
B) password
C) calling from
D) registration flags
E) privilege ievel
F) number of times called
G) first 8 user flags
H) second 8 user flags
I) initial log on menu #
J) not used
K) not used
L) user prompting level
M) time out value
N) not used
O) last msg received
P) caller # on last call
Q) last time on

Choosing one of the fields to modify the system prints the original data. If ENTER is pressed by itself, the data is not changed; while typing the modifications that data is modified. Not all entries can be modified. The user's listing need not be printed out after each modification. After performing a modification the SysOp is returned to the COMMAND> prompt and may perform any of the valid commands. The command help table may be printed by pressing '?'.

How a user is upgraded depends on how you have the system set up and how much access you want that user to have. With the simple menu system included, to upgrade a caller to a normal user's access (Leave messages) you modify the privilege and upgrade it to 30. You would also like to show him registered, so use the 'R' command and choose the type of user you want him to be (New, Probationary or Registered).

System Message Purge

This routine is used to eliminate the deleted messages from the message base file as it grows in size. The routine is self-prompting and really does not need any additional information.

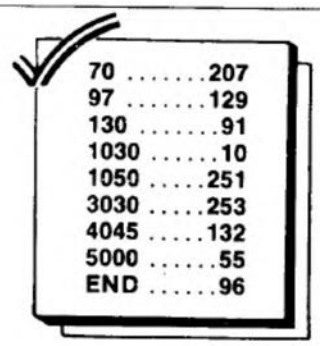
Booting up

Your modem must be an auto-answer modem. Some manuals mention a special setup for the CoCo, but you should ignore this. When using the Pak, you operate like the Model III and many other systems. The main thing is that you set the modem to be controlled by the DTR line as this is the line that hangs up the modem. If you have a Modem II, you should check out CompuServe or another system using that modem, as it does require some changes to the system that I do not have.

The first step in getting the system up is to initialize the Drive 0 disk with *SCF/EDI*, then set up the menu file with *SMF/EDI*. Next, get the system into 64K mode, type PCLEAR1, and load and run *STARTUP/BAS*. Everything is self-prompting from there. You enter the date/day/time, whether or not you want the system protected (for CoCo I's) and then if you want the C/R mod.

The protection causes the modem to hang up if the BBS goes into BASIC any way other than under a command. The other modification is actually done to the driver routine and, if installed, modifies it so a CHR$(13) is the output when there is no carrier detect. If using 1200 Baud, this is required and is good at anytime to make certain the system reboots if a carrier is lost while in use. But, when installed, it also means you cannot break into BASIC without removing it. This is handled anytime you properly go into BASIC from the standby screen, or exit into BASIC with one of the commands. Logging on locally from the keyboard requires that you first remove the modification (SHIFT-up arrow at the standby screen) then RUN the program and use it normally. Another way is to use the SHIFT-up arrow and type POKE 4658,0:CONT. To reinstall the modification from BASIC, type GOTO10000 in either *USER/SYS* or *COBBS/SYS*.

If you have initialized all the disk correctly, created a menu file and booted up with *STARTUP/BAS*, you are now ready for a call. We have covered in this installment (and the preceding two) all the files required to get *CoBBS* up and running. Because of the space limitation there was not much room to get into a detailed discussion of the board operation, but with a little experimentation you should be able to get everything going. Take everything one step at a time rather than trying to get a "full-blown" system up at once.



Listing 1: SCF EDI

```
0 '-SCF/EDI (C) 1985 BY RICHARD
DUNCAN
5 CLEAR1000:DIM SY$(50)
10 CLS:PRINT:PRINT
20 PRINTTAB(5);"SYSTEM CONTROL E
DITOR"
30 PRINT:PRINT
40 PRINT"LOG-ON TYPE (4): ";:GOS
UB650:CH=VAL(CH$):SY$(1)=CH$:IFC
H)4THEN40ELSE1FCH=0 THEN SY$(1)=
"4"
50 PRINT"MENU TO ENTER IN (0): "
;:GOSUB650:CH=VAL(CH$):SY$(2)=CH
$:IFCH)255ORCH<0THEN50ELSEIFCH$=
""THENSY$(2)="0"
55 PRINT"NEW USER TIME OUT (4):
";:GOSUB650:CH=VAL(CH$):IFCH)255
THEN55ELSESY$(3)=CH$:IFSY$(3)=""
THENSY$(3)="4"
60 PRINT"NEW USER PRIVILEGE (25)
: ";:GOSUB650:CH=VAL(CH$):IFCH<0
ORCH)255THEN60ELSESY$(4)=CH$:IFC
H$=""THENSY$(4)="25"
65 PRINT"NEW USER FLAG 1 (000000
00): ";:GOSUB650:IFLEN(CH$)<>8TH
EN65ELSESY$(5)=CH$
67 FORA=1TO8:X$=MID$(CH$,A,1):IF
X$="1"ORX$="0"THENNEXTA:ELSE65
70 PRINT"NEW USER FLAG 2 (000000
00): ";:GOSUB650:IFLEN(CH$)<>8TH
EN70ELSESY$(6)=CH$
72 FORA=1TO8:X$=MID$(CH$,A,1):IF
X$="1"ORX$="0"THENNEXTA:ELSE70
74 PRINT"NEW USER PROMPTING (1):
 ";:GOSUB650:IFCH$=""THENSY$(7)=
```

```
"1"ELSECH=VAL(CH$):IFCH<1ORCH)4T
HEN74ELSESY$(7)=CH$
75 PRINT"CHECK FOR MESSAGES? (Y)
 ";:GOSUB650:IFCH$=""THENSY$(8)=
"Y"ELSEK=INSTR("YyNn",CH$):IFK)2
THENSY$(8)="N"ELSE75
80 PRINT"TRACE USER'S OPERATION?
 (2) ";:GOSUB650:IFCH$=""THENCH$
="2"ELSEIFLEN(CH$)<>1THEN80ELSEC
H=VAL(CH$)
82 IFCH)3THEN80ELSESY$(9)=CH$
85 PRINT"SECURITY PASSWORD? (Y):
 ";:GOSUB650:IFCH$=""THENSY$(10)
="Y"ELSEK=INSTR("YyNn",CH$):IFK=
0THEN85ELSEIFK<3THENSY$(10)="Y"E
LSESY$(10)="N"
90 PRINT"REGISTER NEW USERS? (Y)
 ";:GOSUB650:IFCH$=""THENSY$(11)
="Y"ELSEK=INSTR("YyNn",CH$):IFK=
0THEN90ELSEIFK<3THENSY$(11)="Y"E
LSESY$(11)="N"
95 PRINT"DISPLAY NEWUSER/TXT? (Y
) ";:GOSUB650:IFCH$=""THENSY$(12
)="Y"ELSEK=INSTR("YyNn",CH$):IFK
=0THEN95ELSEIFK<3THENSY$(12)="Y"
ELSESY$(12)="N"
97 PRINT"DISPLAY POSTLOG/TXT? (Y
) ";:GOSUB650:IFCH$=""THENSY$(13
)="Y"ELSEK=INSTR("YyNn",CH$):IFK
=0THEN97ELSEIFK<3THENSY$(13)="Y"
ELSESY$(13)="N"
100 PRINT"BLANK IDLE SCREEN? (Y)
 ";:GOSUB650:IFCH$=""THENSY$(14)
="Y"ELSEK=INSTR("YyNn",CH$):IFK=
0THEN100ELSEIFK<3THENSY$(14)="Y"
ELSESY$(13)="N"
110 PRINT"MESSAGE HEADER DRIVE (
0): ";:GOSUB650:CH=VAL(CH$):IFCH
<0ORCH)3THEN110ELSESY$(15)=CH$:I
FCH$=""THENSY$(15)="0"
115 PRINT"MESSAGE TEXT DRIVE (0)
: ";:GOSUB650:CH=VAL(CH$):IFCH(0
ORCH)3THEN115ELSESY$(16)=CH$:IFC
H$=""THENSY$(16)="0"
120 PRINT"USERLOG DRIVE (0): ";:
GOSUB650:CH=VAL(CH$):IFCH(0ORCH)
3THEN120ELSESY$(17)=CH$:IFCH$=""
THENSY$(17)="0"
125 PRINT"BOARD MENU DRIVE (0):
";:GOSUB650:CH=VAL(CH$):IFCH(0OR
CH)3THEN125ELSESY$(18)=CH$:IFCH$
=""THENSY$(18)="0"
130 'PRINT"CALLER LOG DRIVE (0):
 ";:GOSUB650:CH=VAL(CH$):IFCH(0O
RCH)3THEN130ELSESY$(19)=CH$:IFCH
$=""THENSY$(19)="0"
135 IFVAL(SY$(9))=2THENPRINT"SYS
TEM TRACE DRIVE (0): ";:GOSUB650
:CH=VAL(CH$):IFCH(0ORCH)3THEN135
ELSESY$(20)=CH$:IFCH$=""THENSY$(
20)="0"
140 IFSY$(11)="Y"THENPRINT"REGIS
TER DRIVE (0): ";:GOSUB650:CH=VA
L(CH$):IFCH(0ORCH)3THEN140ELSESY
$(21)=CH$:IFCH$=""THENSY$(21)="0
"
500 GOTO1000
650 '-LINE INPUT-
655 LINEINPUTCH$:RETURN
675 GOSUB655:G1$="":IFCH$=""THEN
RETURN
680 FOR G=1 TO LEN(CH$)
685 G1=ASC(MID$(CH$,G,1)):IFG1)9
6ANDG1<123THEN G1=G1-32
690 G1$=G1$+CHR$(G1):NEXTG:CH$=G
1$:RETURN
1000 '
1005 CLS:PRINT:PRINT
1010 PRINT"LOG ON TYPE: ";SY$(1)
1015 PRINT"MENU ENTRY: ";SY$(2)
1020 PRINT"NEW USER TIME OUT: ";
SY$(3):PRINTTAB(10)"MINUTES:";5*
VAL(SY$(3))
1025 PRINT"NEW USER PRIVILEDGE:
";SY$(4)
1030 PRINT"NEW USER FLAG 1: ";SY
$(5)
1035 PRINT"NEW USER FLAG 2: ";SY
$(6)
1040 PRINT"NEW USER PROMPTING: "
;SY$(7)
1045 PRINT:PRINT
1050 PRINT"(Q)UIT TO RESTART ";:
GOSUB650:IFLEFT$(CH$,1)="Q"THENR
UN
1055 CLS:PRINT:PRINT
1060 PRINT"CHECK MESSAGES: ";:IF
SY$(8)="Y"THENPRINT"YES"ELSEPRIN
T"NO"
1065 PRINT"TRACER: ";SY$(9)
1070 PRINT"SECURITY PASSWORD: ";
:IFSY$(10)="Y"THENPRINT"YES"ELSE
PRINT"NO"
1075 PRINT"REGISTER USERS: ";:IF
SY$(11)="Y"THENPRINT"YES"ELSEPRI
NT"NO"
1080 PRINT"NEW USER TEXT: ";:IFS
Y$(12)="Y"THENPRINT"YES"ELSEPRIN
T"NO"
1085 PRINT"POST LOG TEXT: ";:IFS
Y$(13)="Y"THENPRINT"YES"ELSEPRIN
T"NO"
1090 PRINT"BLANK IDLE SCREEN: ";
:IFSY$(14)="Y"THENPRINT"YES"ELSE
PRINT"NO"
1095 PRINT:PRINT"(Q)UIT OR (ENTE
R)";:GOSUB650:IFLEFT$(CH$,1)="Q"
THENRUN
1100 CLS:PRINT:PRINT
1105 PRINT"HEADER DRIVE: ";SY$(1
5)
1110 PRINT"TEXT DRIVE: ";SY$(16)
1115 PRINT"USERLOG DRIVE: ";SY$(
17)
1120 PRINT"BOARD MENU DRIVE: ";S
Y$(18)
1125 'PRINT"CALLER LOG DRIVE: ";
SY$(19)
1130 PRINT"TRACE DRIVE: ";SY$(20
)
1135 PRINT"REGISTER DRIVE: ";SY$
(21)
1140 PRINT:PRINT"(Q)UIT OR (ENTE
R)";:GOSUB650:IFLEFT$(CH$,1)="Q"
THEN RUN
3000 '-SAVE SYSTEM CONTROLS
3005 CLS:PRINT:PRINT
3010 PRINT"ABOUT TO SAVE FILE!!!
"
3015 PRINT:PRINT"INSERT SYSTEM D
RIVE ZERO"
3020 PRINT"DISK IN DRIVE ZERO. T
HIS"
3025 PRINT"FUNCTION WILL WRITE T
O TRACK"
3030 PRINT"17 SECTOR 18 AND MUST
 BE"
3035 PRINT"THERE FOR COBBS TO OP
ERATE."
3040 PRINT:PRINT:PRINT"READY TO
SAVE? (YES/NO) ";:GOSUB650
3045 IFCH$="YES"THEN4000
3050 CLS:PRINT:PRINT:PRINT
3055 PRINT"1 - RESTART":PRINT"2
- SAVE FILE":PRINT"3 - END"
3060 PRINT" )";:GOSUB650:CH=VAL(
CH$)
3065 IFCH<1ORCH)3THEN3050
3070 ON CH GOTO 3075,3000,3080
3075 RUN
3080 UNLOAD:END
4000 '-SAVE FILE.....
4005 '
4010 CLS8
4015 PRINT@229,"INSERT COBBS SYS
TEM DISK";:GOSUB650
4020 CLS3:PRINT@233,"READY TO SA
VE";:GOSUB650
4025 IFLEFT$(CH$,1)<>"Y"THEN3000
4030 CLS0:PRINT@235,"PROCESSING"
;
4035 S$="COBBS11A"
4040 FOR A=1TO4:S$=S$+CHR$(VAL(S
Y$(A))):NEXTA
4045 F$=SY$(5):GOSUB5040:S$=S$+F
$
4050 F$=SY$(6):GOSUB5040:S$=S$+F
```

```
$
4055 S$=S$+CHR$(VAL(SY$(7)))
4060 IFSY$(8)="Y"THEN S$=S$+CHR$
(255)ELSE S$=S$+CHR$(0)
4065 S$=S$+CHR$(VAL(SY$(9)))
4070 IFSY$(10)="Y"THEN S$=S$+CHR
$(0)ELSES$=S$+CHR$(255)
4075 IFSY$(11)="Y"THEN S$=S$+CHR
$(0)ELSES$=S$+CHR$(255)
4080 IFSY$(12)="Y"THEN S$=S$+CHR
$(0)ELSES$=S$+CHR$(255)
4085 IFSY$(13)="Y"THEN S$=S$+CHR
$(0)ELSES$=S$+CHR$(255)
4090 IFSY$(14)="Y"THEN S$=S$+CHR
$(255)ELSES$=S$+CHR$(0)
4100 FORA=15TO21:S$=S$+CHR$(VAL(
SY$(A))):NEXTA
4900 IFLEN(S$))128 THEN S1$=LEFT
$(S$,128):S2$=RIGHT$(S$,LEN(S$)-
128) ELSE S1$=S$:S2$=STRING$(127
,233)
4905 CLS4:PRINT@232,"SAVING DATA
";
4910 DSKO$ 0,17,18,S1$,S2$
4920 CLS:PRINT:PRINT"SYSTEM CONT
ROL FILE SAVED."
4925 END
5000 '-DECIMAL TO BINARY
5005 F=ASC(F$):E=128:F$=""
5010 FOR Q=1 TO 8
5015 J=INT(F/E)
5020 IF J=0 THEN F$=F$+"0"ELSEF$
=F$+"1"
5025 F=F-(E*J):E=E/2
5030 NEXT Q
5035 RETURN
5040 '-BINARY TO DECIMAL
5045 E=1:F=0
5050 FOR Q=8 TO 1 STEP -1
5055 IFMID$(F$,Q,1)="1"THEN F=F+
E
5060 E=E*2:NEXTQ:F$=CHR$(F)
5065 RETURN
```

```
75 ........74
120 ......149     1250 ......25
216 ......200     1340 .....169
635 ........46    1525 .....116
1022 ......25     1660 .....248
1105 .....178     2040 .....205
1160 ......77     END .....227
```

Listing 2: SMF EDI

```
"10 CLEAR5000:DIM BC$(21),PR$(21
),P1$(21),P2$(21),TY$(21),D$(21)
,KY$(21),TX$(21)
20 CLS:PRINT:PRINT:PRINT"  COBBS
 MENU CONTROL FILE 1.0":PRINT
25 PRINTTAB(5)"(1) CREATE NEW ME
NU"
30 PRINTTAB(5)"(2) EDIT EXISTING
 MENU"
35 PRINTTAB(5)"(3) EXIT EDITING"
:PRINT"     (4) RETURN TO BBS"
40 GOSUB600:A=VAL(CH$)
45 ON A+1 GOTO 40,200,50,46,47
46 CLOSE:UNLOAD:PRINT"     ACCES
S: ";:GOSUB675:IFCH$="REBOOT CLE
ARED" THEN END ELSE RUN
47 LOAD"COBBS/SYS",R
50 PRINT:PRINT:'PRINT"MENU PREFI
X: ";:GOSUB9850
52 PRINT"BOARD NUMBER: ";:GOSUB9
850:IFCH$=""THEN RUN
55 BC=VAL(CH$):IFBC)255THEN20
60 OPEN"D",#1,"MENU/SYS",250:K1=
LOF(1)
65 FIELD#1,1 AS B0$,1 AS B1$,1 A
S B2$,1 AS B3$,16 AS B4$,200 AS
B5$,1 AS B6$,29 AS B7$
70 FOR A=1 TO K1 STEP 4
75 GET#1,A:IF ASC(B0$)=BC THEN R
0=A:GOTO80 ELSE NEXT A:CLOSE:RUN
80 GOSUB9700:PRINT:PRINT"LOADING
 MENU";BC:BC=ASC(B0$):PR=ASC(B1$
):F$=B2$:GOSUB150:F1$=F$:F$=B3$:
GOSUB150:F2$=F$:BN$=B4$:BT$=B5$
81 K=INSTR(BN$,CHR$(0)):IFK=0THE
N82ELSEBN$=LEFT$(BN$,K-1)
82 K=INSTR(BT$,CHR$(0)):IFK=0THE
N85ELSEBT$=LEFT$(BT$,K-1)
83 IF ASC(B6$)=0 THEN PM$="N" EL
SE PM$="Y"
85 FIELD#1,35 AS B$(0),35 AS B$(
1),35 AS B$(2),35 AS B$(3),35 AS
 B$(4),35 AS B$(5),35 AS B$(6),5
 AS B$(7)
90 B=1:X=0:CO=0
92 X=X+1:GET#1,R0+X
95 FOR A=0 TO 6
96 IF B$(A)=STRING$(35,255)THEN1
20
97 CO=CO+1:BC$(A+B)=B$(A):PRINT"
  RECORD #";CO
100 B$=B$(A):PR$(A+B)=STR$(ASC(L
EFT$(B$,1))):F$=MID$(B$,2,1):GOS
UB150:P1$(A+B)=F$:F$=MID$(B$,3,1
):GOSUB150:P2$(A+B)=F$:TY$(A+B)=
MID$(B$,4,1):D$(A+B)=MID$(B$,5,5
):KY$(A+B)=MID$(B$,18,1):TX$(A+B
)=RIGHT$(B$,25):NEXTA
105 B=B+7:IFB)16THEN120ELSE92
120 PRINT"WAIT..."
135 REM
140 GOTO500
150 '-DECIMAL TO BINARY
152 F=ASC(F$):E=128:F$=""
154 FOR Q=1 TO 8
156 J=INT(F/E)
158 IF J=0 THEN F$=F$+"0"ELSEF$=
F$+"1"
160 F=F-(E*J):E=E/2
162 NEXT Q
166 RETURN
175 '-BINARY TO DECIMAL
177 W=LEN(F$):E=1:F=0
179 FOR Q=W TO 1 STEP -1
181 IFMID$(F$,Q,1)="1"THEN F=F+E
183 E=E*2:NEXTQ:F$=CHR$(F)
185 RETURN
200 CLS:PRINT:PRINT:PRINT
201 F$="MENU/SYS":'PRINT"FOUR CH
ARACTER PREFIX: ";:GOSUB9850:IFL
EN(CH$))4THEN201ELSE F$=CH$+"MEN
U/SYS"
202 PRINT"KILL OLD MENU FILE? ";
205 GOSUB675
210 IFCH$="N"THENPRINT"NO":FORT=
1TO500:NEXTT:GOTO220
215 IFCH$="Y"THENPRINT"YES":KILL
 "MENU/SYS":R0=1:GOTO220
216 GOTO205
220 OPEN"D",#1,F$,250:K1=LOF(1)
500 FORT=1TO5:PRINT:NEXT T:CLS:P
RINT"#";BC;TAB(10);BN$
505 GOSUB9700:PRINT"COMMANDS: ";
CO:PRINT
510 PRINT"(1) ADD A COMMAND"
520 PRINT"(2) CHANGE A COMMAND"
525 PRINT"(3) LIST MENU"
530 PRINT"(4) ADD/CHANGE BOARD M
ENU"
535 PRINT"(5) LIST/EDIT HEADER"
540 PRINT"(6) DELETE A COMMAND
545 PRINT"(7) EXIT AND WRITE
550 PRINT"(8) EXIT":GOSUB9700
555 GOSUB600:A=VAL(CH$):IFA(1ORA
)8THEN555
560 GOSUB9700:ON A GOSUB 1100,12
00,1300,1000,1500,1600,2000,9999
565 GOTO500
600 '-SINGLE KEY ENTRY
602 CH$=INKEY$:IFCH$()""THEN610
605 EXEC&H10DA:CH$=CHR$(PEEK(448
1)):IFCH$=CHR$(0)THEN602ELSEIFCH
$=CHR$(13)THENCH$=""
610 GOSUB9700
615 RETURN
625 GOSUB602:IFCH$=""THENRETURNE
LSECH=ASC(CH$)
630 IFCH)96ANDCH(123THEN CH=CH-3
2
635 CH$=CHR$(CH):RETURN
650 '-REMOTE INPUT  **KEYBOARD O
NLY
655 LINEINPUTCH$:GOSUB9700
660 GOSUB9700
665 RETURN
```

```
675 GOSUB655:G$="":IFCH$=""THENR
ETURN
680 FOR A=1 TO LEN(CH$)
685 G=ASC(MID$(CH$,A,1)):IFG)96A
NDG(123THEN G=G-32
690 G$=G$+CHR$(G):NEXTA:CH$=G$:R
ETURN
1000 '-CREATE A MENU-
1005 '
1010 PRINT:PRINT:PRINT:GOSUB9700
:PRINT"(A)DD OR (C)HANGE: ";:GOS
UB625:IFCH$="A"THEN B8=0:GOTO101
5 ELSE IFCH$="C"THEN B8=1 ELSE R
ETURN
1015 R0=LOF(1)+1:CLS:PRINT:PRINT
:PRINT:PRINT"  BOARD #";:GOSUB98
50:IF CH$="" THEN RETURN ELSE X=
VAL(CH$):IF X(0 OR X)254 THEN 10
15
1020 FIELD#1,1 AS B1$,1 AS B2$,1
 AS B3$,1 AS B4$,16 AS B5$,230 A
S B6$:K1=LOF(1):IF K1=0 THEN 103
0
1022 FOR A=1 TO K1 STEP 4:IF ASC
(B1$)=X THEN 1024 ELSE NEXT A:GO
TO1030
1024 CLS:PRINT:PRINT"BOARD EXIST
:":PRINT"NUMBER: ";ASC(B1$):PRIN
T"PRIV: ";ASC(B2$):PRINT"NAME: "
;B5$:PRINT:PRINT"DELETE (Y/N) "
1026 GOSUB9800:IF CH$()"Y"THENRE
TURN
1030 BC=X:R0=A
1031 PRINT"PRIV: ";:GOSUB9850:PR
=VAL(CH$):IF PR(0 OR PR)254 THEN
 1031
1032 PRINT"FLAG 1: ";:GOSUB9850:
IFLEN(CH$)()8THEN1032ELSE F1$=CH
$
1034 PRINT"FLAG 2: ";:GOSUB9850:
IFLEN(CH$)()8THEN1034ELSEF2$=CH$
1036 PRINT"NAME: ";:GOSUB9850:BN
$=LEFT$(CH$+STRING$(16,0),16)
1037 IF B8=0 THEN CO=0
1038 PRINT"POST MESSAGES? ";:GOS
UB625:IFCH$="Y"THEN PM$="Y":PRIN
T"YES"ELSEPRINT"NO": PM$="N"
1040 PRINT"ENTER TEXT, USE SHIFT
-CLEAR KEY":PRINT"FOR (CR). END
WITH (CR) ALONE"
1045 TS$=CHR$(13)+CHR$(8)+"\":GO
TO1574
1050 RETURN
1100 '-ADD A COMMAND-
1105 PRINT:PRINT:PRINT"INSERT AF
TER: ";
1106 GOSUB9850:Y=VAL(CH$):IFY=0
AND CH$="L"THEN Y=CO ELSEIFCH$="
"THENRETURN
1110 Y=Y+1:IFY)21THENPRINT"BUFFE
R FULL.":RETURN
1115 CO=CO+1:IF CO)21 THEN PRINT
"COMMAND BUFFER FULL":RETURNELSE
CLS:PRINT:PRINT
1120 PRINT"BOARD COMMAND #";CO
1125 PRINT"TEXT: ";:GOSUB650:TX$
=LEFT$(CH$+STRING$(25,0),25)
1130 PRINT" KEY: ";:GOSUB9800:KY
$=CH$:PRINTCH$
1135 PRINT"PRIV: ";:GOSUB9850:X=
VAL(CH$):IFX(0ORX)254THEN1135ELS
E PR$=CH$
1140 PRINT"  P1: ";:GOSUB9850:IF
LEN(CH$)()8THEN1140ELSE P1$=CH$
1145 PRINT"  P2: ";:GOSUB9850:IF
LEN(CH$)()8THEN1145ELSE P2$=CH$
1150 PRINT:PRINT"TYPE: ";:GOSUB9
800:IFCH$=""THEN1150ELSEX=ASC(CH
$):IFX(33ORX)90THEN1150ELSE TY$=
CH$:PRINTCH$
1155 PRINT"DATA: ";:GOSUB9850:IF
LEN(CH$))5THEN1155ELSED$=CH$
1160 PRINT:PRINT"DATA ENTERED":F
ORT=1TO500:NEXTT:PRINT"WAIT...":
IF CO=1 OR Y=CO THEN A=CO:GOTO111
80
1165 FOR A=CO-1 TO 1 STEP -1
1170 TX$(A+1)=TX$(A):KY$(A+1)=KY
$(A):PR$(A+1)=PR$(A):P1$(A+1)=PR
$(A):P1$(A+1)=P1$(A):P2$(A+1)=P2
$(A):TY$(A+1)=TY$(A):D$(A+1)=D$(
A)
1175 IF A=Y THEN 1180ELSENEXT A:
A=1:RETURN
1180 TX$(A)=TX$:KY$(A)=KY$:PR$(A
)=PR$:P1$(A)=P1$:P2$(A)=P2$:TY$(
A)=TY$:D$(A)=D$
1185 RETURN
1200 '-CHANGE MENU-
1205 '
1210 '
1215 CLS:PRINT:PRINT
1220 PRINT"MENU NUMBER: ";:GOSUB
9850:IFCH$=""THENRETURN
1225 X=VAL(CH$):IFX)21THEN1220
1230 CLS:PRINT:PRINT:PRINT"MENU
PROMPT #";X:PRINT
1235 PRINT"TEXT: ";TX$(X):PRINT"
   )";:GOSUB9850
1240 IFCH$=""THEN1245ELSETX$(X)=
CH$
1245 PRINT" KEY: ";KY$(X):PRINT"
    )";:GOSUB9800:IFCH$=""THEN12
50ELSEKY$(X)=CH$:PRINTCH$
1250 PRINT"PRIV: ";PR$(X):PRINT"
 )";:GOSUB9850:IFCH$=""THEN1255E
LSEPR$(X)=CH$
1255 PRINT"  P1: ";P1$(X):PRINT"
 )";:GOSUB9850:IFCH$=""THEN1260E
LSEP1$(X)=CH$
1260 PRINT"  P2: ";P2$(X):PRINT"
 )";:GOSUB9850:IFCH$=""THEN1265E
LSEP2$(X)=CH$
1265 PRINT"TYPE: ";TY$(X);:PRINT
" )";:GOSUB9800:IFCH$=""THEN1270
ELSETY$(X)=CH$:PRINTCH$
1270 PRINT:PRINT"DATA: ";D$(X);:
PRINT" )";:GOSUB9850:IFCH$=""THE
N1275ELSED$(X)=CH$
1275 PRINT:PRINT"EDIT COMPLETE"
1280 FORT=1TO500:NEXTT:RETURN
1300 '-LIST MENU-
1305 '
1310 GOSUB9700
1315 CLS:PRINT:PRINT:PRINT
1320 PRINTTAB(5)"(1) FULL LISTIN
G"
1325 PRINTTAB(5)"(2) TEXT LISTIN
G"
1330 PRINTTAB(5)"(3) PRINT LISTI
NG"
1335 GOSUB600:IFCH$=""THEN1335EL
SEA=VAL(CH$)
1340 IFA)3THEN1335
1345 ON A+1 GOTO 1350,1355,1400
1350 RETURN
1355 FOR X=1 TO CO
1356 GOSUB1360:GOTO1396
1360 CLS:PRINT:PRINT:PRINT"MENU
NUMBER: ";X
1365 PRINT:PRINT"TEXT: ";TX$(X)
1370 PRINT" KEY: ";KY$(X)
1375 PRINT"PRIV: ";PR$(X)
1380 PRINT"  P1: ";P1$(X)
1385 PRINT"  P2: ";P2$(X)
1390 PRINT"TYPE: ";TY$(X)
1395 PRINT"DATA: ";D$(X):RETURN
1396 GOSUB625:IFCH$="Q"THEN RETU
RN ELSE NEXT X
1397 PRINT"END OF FILE":FORT=1TO
500:NEXTT:RETURN
1400 CLS:PRINT
1405 PRINT BT$:PRINT
1410 PRINT:PRINTBN$
1415 FOR X=1 TO CO
1420 PRINT TX$(X):GOSUB9700
1425 FORT=1TO100:NEXTT
1430 NEXTX
1435 GOSUB9800::RETURN
1500 '-HEADER-
1505 '
1510 '
1515 CLS:PRINT:PRINT:PRINT
1520 PRINTTAB(5)"(1) LIST HEADER
"
1525 PRINTTAB(5)"(2) ENTER NEW H
EADER":PRINTTAB(5)"(ENTER) TO EX
```

```
IT"
1530 GOSUB600:A=VAL(CH$)
1535 IFA)2THEN1530
1540 ON A+1 GOTO1545,1550,1565
1545 RETURN
1550 CLS:PRINT:PRINT:PRINT
1555 PRINT"HEADER FOR BOARD: ";B
C:PRINT:PRINTBT$
1560 GOSUB9800:GOTO1500
1565 CLS:PRINT:PRINT:PRINT:PRINT
"OLD HEADER READS:"
1570 TS$=CHR$(13)+CHR$(8)+"\":PR
INTBT$:PRINT:PRINT"ENTER NEW HEA
DER...":PRINT:PRINT:BT$=""
1574 PRINT")";
1575 GOSUB600:IFCH$=""THEN1500EL
SEK=INSTR(TS$,CH$):ON K+1 GOTO 1
595,1580,1585,1590
1580 GOTO1500
1585 IFLEN(BT$))1THEN PRINTCHR$(
8);:BT$=LEFT$(BT$,LEN(BT$)-1):GO
TO1575
1590 PRINT:BT$=BT$+CHR$(13):GOTO
1574
1595 BT$=BT$+CH$:PRINTCH$;:GOTO1
575
1600 '-DELETE A COMMAND
1605 '
1610 '
1615 CLS:PRINT:PRINT:PRINT"COMMA
ND #";:GOSUB9850
1620 X=VAL(CH$):IFX=0THEN RETURN
ELSEIFX)CO THEN1615
1625 PRINT:PRINT"COMMAND #";X:PR
INTTX$(X)
1630 PRINT"DELETE (Y/N)? ";
1635 GOSUB9800:IFCH$()"Y"THENRET
URN
1640 PRINT"  DELETING...";
1645 FORA=X TO CO-1
1650 TX$(A)=TX$(A+1):KY$(A)=KY$(
A+1):PR$(A)=PR$(A+1):P1$(A)=P1$(
A+1):P2$(A)=P2$(A+1):TY$(A)=TY$(
A+1):D$(A)=D$(A+1)
1655 NEXTA:PRINT:CO=CO-1:RETURN
2000 '-SAVE MENU
2005 '
2010 '
2015 CLS:PRINT:PRINT
2020 PRINT"SAVING HEADER....."
2025 FIELD#1,1 AS B1$,1 AS B2$,1
 AS B3$,1 AS B4$,16 AS B5$,200 A
S B6$,1 AS B7$,29 AS B8$
2030 IFLOF(1)=0 THEN R0=1
2035 LSET B1$=CHR$(BC):LSET B2$=
CHR$(PR):F$=F1$:GOSUB175:LSET B3
$=F$:F$=F2$:GOSUB175:LSET B4$=F$
2040 LSET B5$=BN$+CHR$(0):LSET B
6$=BT$+CHR$(0):IF PM$="Y"THEN LS
ET B7$=CHR$(255)ELSE LSET B7$=CH
R$(0)
2045 PUT#1,R0
2050 PRINT"SAVING MENU COMMANDS.
.."
2055 FIELD#1,35 AS B$(0),35 AS B
$(1),35 AS B$(2),35 AS B$(3),35
AS B$(4),35 AS B$(5),35 AS B$(6)
,5 AS B$(7):X=0
2057 FOR B=1 TO 16 STEP 7
2060 FOR A=0 TO 6
2065 IF A+B)CO THEN C$=STRING$(3
5,255)ELSEGOSUB2095:PRINT"  SAVI
NG RECORD";A+B
2070 LSET B$(A)=C$:NEXTA:LSET B$
(7)=MKN$(BC)
2075 R0=R0+1:PUT#1,R0
2080 NEXTB:RUN
2095 X=VAL(PR$(A+B)):C$=CHR$(X)
2100 F$=P1$(A+B):GOSUB175:C$=C$+
F$:F$=P2$(A+B):GOSUB175:C$=C$+F$
2105 C$=C$+TY$(A+B):D$=LEFT$(D$(
A+B)+"      ",5)
2110 C$=C$+D$+KY$(A+B)+TX$(A+B)+
STRING$(25,0)
2115 RETURN
9700 '-CD CHECK-
9705 IFPEEK(4658)=0THEN9740
9710 CD=PEEK(65385)AND32
9715 IFCD()0ORPEEK(4657)()0THEN9
750
9740 RETURN
9750 CLOSE:UNLOAD:RUN
9800 '
9805 GOTO625
9806 IFCH$=CHR$(13)THENCH$=""
9810 RETURN
9815 '-FLAG INPUT-
9820 F$=""
9825 FOR X=1 TO 8
9830 GOSUB600:VL=ASC(CH$):IFVL(4
8ORVL)49THEN600 ELSE F$=F$+CH$:P
RINTCH$;:NEXTX
9835 PRINT:RETURN
9850 '-LINE ENTRY-
9855 GOTO675
9860 RETURN
```

610	179	3270	184
925	150	3565	247
1125	105	4020	160
1370	230	6040	113
2005	52	6200	102
2085	229	7010	55
3125	198	8045	133
		END	176

Listing 3: SUL EDI

```
0 '-COBBS (SUL/EDI) (C)1985
    BY RICHARD DUNCAN
10 CLEAR2500
20 DR$(0)="0":DR$(1)="1":DR$(2)=
"2":DR$(3)="3":R0=2
30 CLS:PRINT:PRINT"COBBS USERLOG
 EDITOR":PRINT:PRINT
40 GOSUB9200
45 GOSUB900
50 PRINT:PRINT"COMMAND) ";
60 GOSUB625:K=INSTR("SGNLMRADPBQ
UE?H",CH$):IFK=0THEN60ELSE PRINT
CH$
70 ON K GOSUB 1000,1100,80,90,12
00,1300,1400,1500,1640,2000,7000
,1700,6000,75,1600
71 GOTO50
75 CLS:PRINT:PRINT"    COBBS USE
RLOG EDITOR":PRINT:PRINT:GOSUB90
0:RETURN
80 R0=R0+1:IF R0)LOF(1) THEN R0=
1
85 GET#1,R0:GOSUB8000:RETURN
90 R0=R0-1:IF R0(1 THEN R0=LOF(1
)
95 GET#1,R0:GOSUB8000:RETURN
600 '-SINGLE KEY ENTRY
602 CH$=INKEY$:IFCH$()""THEN610
605 EXEC4314:CH$=CHR$(PEEK(4481)
)
610 GOSUB9700:IFCH$=CHR$(0)THEN6
02
615 RETURN
625 GOSUB602:CH=ASC(CH$)
630 IFCH)96ANDCH(123THEN CH=CH-3
2
635 CH$=CHR$(CH):RETURN
650 '-REMOTE INPUT  **KEYBOARD O
NLY
655 LINEINPUTCH$:GOSUB9700
665 RETURN
675 GOSUB655:G$="":IFCH$=""THENR
ETURN
680 FOR A=1 TO LEN(CH$)
685 G=ASC(MID$(CH$,A,1)):IFG)96A
NDG(123THEN G=G-32
690 G$=G$+CHR$(G):NEXTA:CH$=G$:R
ETURN
900 PRINTTAB(10)"USERS:";LOF(1)-
1:PRINTTAB(0)"(S)EARCH";TAB(16)"
(G)O TO USER #"
905 PRINTTAB(0)"(N)EXT";TAB(16)"
(L)AST ENTRY"
910 PRINTTAB(0)"(M)ODIFY FIELD";
TAB(16)"(R)EGISTER"
915 PRINTTAB(0)"(A)CTIVATE ";TAB
(16)"(D)ELETE
```

```
920 PRINTTAB(0)"(P)RINT LISTING"
;TAB(16)"(B)ACKUP"
925 PRINTTAB(0)"(E)NTER USER";TA
B(16)"(U)SER CLASS"
930 PRINTTAB(0)"(Q)UIT";TAB(16);
"(H)ARD COPY"
990 RETURN
1000 '-SEARCH-
1005 '
1010 '
1015 PRINT:PRINT
1020 PRINTTAB(3)"STRING: ";
1025 GOSUB650:IFCH$=""THENRETURN
1026 K=INSTR(CH$,"\"):IFK=0THEN1
030ELSEMID$(CH$,K,1)=CHR$(0):GOT
01026
1030 K1=LOF(1)
1035 FOR Y=R0 TO K1
1040 GET#1,Y:K=INSTR(UN$,CH$)
1045 IF K=0 THEN NEXT Y:PRINT:PR
INT"USER NOT FOUND":RETURN
1050 R0=Y:GOSUB8000
1055 RETURN
1100 '-GOTO RECORD-
1105 '
1110 '
1115 PRINT:PRINT"RECORD NUMBER:
";
1120 GOSUB650:IF VAL(CH$))LOF(1)
OR VAL(CH$)<1THENRETURN
1125 R0=VAL(CH$):IF R0>LOF(1)THE
NPRINT"RECORD INVALID.":RETURN E
LSE GET#1,R0:GOSUB8000
1130 RETURN
1200 '-MODIFY USER RECORD-
1205 '
1210 PRINT"MODIFY RECORD: ";
1215 GOSUB625
1220 K=INSTR("ABCDEFGHIJKLMNOP",
CH$):IFK=0THENRETURN
1225 PRINTCH$:ON K GOTO 3000,310
0,3050,3120,3150,3200,3250,3300,
3350,3400,3450,3500,3550,3600,36
50,3700,3750
1300 '-REGISTER-
1305 PRINT:PRINT:PRINTNA$:PRINT
1310 PRINT"   1 - NEW USER":PRIN
T"   2 - PROBATION USER":PRINT"
  3 - REGISTERED USER"
1325 PRINT"COMMAND: ";
1330 GOSUB600:C=VAL(CH$):IFC<1 O
R C>3 THEN RETURN ELSE PRINTCH$
1335 ON C GOTO1340,1355,1370:RET
URN
1340 MID$(RF$,2,1)="0":MID$(RF$,
8,1)="0"
1345 C$=RF$:GOSUB4200:LSET UR$=C
$
1350 PUT#1,R0:RETURN
1355 MID$(RF$,2,1)="0":MID$(RF$,
8,1)="1"
1360 GOTO1345
1370 MID$(RF$,2,1)="1":MID$(RF$,
8,1)="1"
1375 GOTO1345
1400 '-ACTIVATE-
1405 MID$(RF$,1,1)="0"
1415 GOTO1345
1500 '-DELETE-
1505 MID$(RF$,1,1)="1"
1510 GOTO1345
1600 '-PRINT
1605 '
1610 CLS:PRINT:PRINT
1615 PRINT"1 - SCREEN":PRINT"2 -
 PRINTER":PRINT"3 - HARD COPY OF
 USERLOG"
1620 PRINT"    )";
1625 GOSUB600:CH=VAL(CH$):IF CH=
0 THEN RETURN ELSE IF CH>3 THEN
1625
1630 PRINTCH$:ON CH GOTO 1640,16
50,1660
1640 SC=0:GOSUB8000:RETURN
1650 SC=-2:GOSUB8000:RETURN
1660 SC=-2
1665 FOR U=2 TO LOF(1)
1670 GET#1,U:GOSUB8000
1675 NEXT U:RETURN
1700 RETURN
2000 '-BACKUP-
2005 PRINT:PRINT:PRINT"USERLOG B
ACKUP"
2010 PRINT:PRINT"DRIVE FOR USERL
/BAK: ";:GOSUB650
2015 IFCH$=""THENRETURNELSE D=VA
L(CH$)
2020 IF D<0 OR D>3 THEN 2010ELSE
PRINT CH$
2025 PRINT"BACKING UP USERLOG":C
LOSE
2030 GOSUB9200
2031 GOSUB9300:IF K2>0THEN CLOSE
:KILL FB$:GOTO2031
2035 GOSUB9225:GET#1,1:LSET U$=P
U$
2040 K2=1:PUT#2,K2
2045 FOR R=2 TO K1
2050 GOSUB9215:GET#1,R:PRINT"USE
R #";R-1;:GOSUB9700
2055 C$=UR$:GOSUB4100:IFLEFT$(C$
,1)="1"THENPRINT"(DELETED)";:GOT
O2065
2060 FIELD#1,96 AS PU$:LSET U$=P
U$:K2=K2+1:PUT#2,K2
2065 PRINT:NEXT R
2070 CLOSE:PRINT:PRINT"VERIFYING
...";
2075 GOSUB9300:FOR R=1 TO K2:GET
#2,R:NEXT R:CLOSE
2080 PRINT"KILL AND COPY? ";:GOS
UB600:IFCH$<>"Y"THENPRINT:GOTO208
0 ELSE PRINT"YES":PRINT:PRINT"K
ILLING OLD FILE";:KILL FS$:PRINT
2085 PRINT"COPYING NEW FILE";:CO
PY FB$ TO FS$:PRINT
2090 CLOSE:GOSUB9200
2095 RETURN
3000 '-CHANGE NAME-
3005 GOSUB4000:PRINT:PRINT"NAME:
 ";NA$:PRINT"CHANGE TO: ";
3010 GOSUB675:IFCH$=""THENRETURN
3015 IF LEN(CH$)+LEN(LO$)>50 THE
NPRINT"NAME TOO LONG!":GOTO3005
3020 LSET UN$=CH$+CHR$(0)+LO$+CH
R$(0):PUT#1,R0
3025 RETURN
3050 '-CHANGE LOCATION-
3055 GOSUB4000:PRINT:PRINT"FROM:
 ";LO$:PRINT"CHANGE TO: ";
3060 GOSUB675:IFCH$=""THENRETURN
3065 IFLEN(NA$)+LEN(CH$)+1>58THE
NPRINT"TOO LONG!":GOTO3055
3070 LSET UN$=NA$+CHR$(0)+CH$+CH
R$(0):PUT#1,R0:RETURN
3100 '-CHANGE PASSWORD
3105 PRINT:PRINT"PASSWORD: ";UP$
:PRINT"CHANGE TO: ";
3110 GOSUB650:IFCH$=""THENRETURN
3115 LSET UP$=CH$+STRING$(8,0):P
UT#1,R0:RETURN
3120 '-CHANGE REGISTER FLAG
3125 C$="":PRINT:PRINT"CHANGE TO
: ";
3130 FOR X=1 TO 8:GOSUB 600:IFCH
$=""THENRETURNELSE C$=C$+CH$:PRI
NTCH$;:NEXTX
3135 GOSUB4200:LSET UR$=C$:PUT#1
,R0:RETURN
3150 '-AUTHORITY CHANGE
3155 PRINT:PRINT"AUTHORITY:";PR:
PRINT"CHANGE TO: ";
3160 GOSUB650:IFVAL(CH$)<256THEN
3165ELSERETURN
3165 IFVAL(CH$)=0 AND CH$<>"0"TH
ENRETURN
3170 LSET UA$=CHR$(VAL(CH$)):PUT
#1,R0:RETURN
3200 '-CHANGE LOG ON #
3205 PRINT:PRINT"NUMBER OF LOGIN
S:";LG:PRINT"CHANGE TO: ";
3210 GOSUB650:IFCH$=""THENRETURN
3215 LSET U5$=MKN$(VAL(CH$)):PUT
#1,R0:RETURN
3250 '-P1 FLAG
3255 C$="":PRINT:PRINT"P1 FLAG:
";F1$:PRINT"CHANGE TO: ";
```

```
3260 FOR X=1 TO 8:GOSUB600:IFCH$
=""THENRETURNELSEC$=C$+CH$:PRINT
CH$;:NEXTX:PRINT
3265 F1$=C$:GOSUB4200:LSET U1$=C
HR$(C):PUT#1,R0
3270 RETURN
3300 '-P2 FLAG
3305 C$="":PRINT:PRINT"P2 FLAG:
";F2$:PRINT"CHANGE TO: ";
3310 FOR X=1 TO 8:GOSUB600:IFCH$
=""THENRETURNELSEC$=C$+CH$:PRINT
CH$;:NEXTX:PRINT
3315 F2$=C$:GOSUB4200:LSET U2$=C
HR$(C):PUT#1,R0:RETURN
3350 '-BOARD ENTRY-
3355 PRINT:PRINT"BOARD ENTRY:";B
C:PRINT"CHANGE TO: ";
3365 GOSUB650:IFVAL(CH$)<256THEN
 LSET UE$=CHR$(VAL(CH$)):PUT#1,R
0:RETURN
3400 PRINT:PRINT"CHANGE TO: ";
3405 GOSUB650:IFCH$=""THENRETURN
3410 IFLEN(CH$))4THEN3400
3415 LSET US$=CH$:PUT#1,R0:RETUR
N
3450 '-
3455 RETURN
3500 '-
3505 RETURN
3550 '-TIME OUT-
3555 PRINT:PRINT"PRESENTLY TIME
OUT:";TM:PRINT"CHANGE TO: ";
3560 GOSUB650:IFCH$=""THENRETURN
3565 C=VAL(CH$):IFC<256THEN LSET
 UO$=CHR$(C):PUT#1,R0
3570 RETURN
3600 '-
3605 RETURN
3650 '-
3655 RETURN
3700 '-
3705 RETURN
3750 '-DATE TIME
3755 PRINT:PRINT"MONTH: ";
3760 GOSUB650:IFCH$=""THENRETURN
ELSE M=VAL(CH$)
3765 PRINT"DAY: ";
3770 GOSUB650:IFCH$=""THENRETURN
ELSE D=VAL(CH$)
3775 PRINT"YEAR: ";
3780 GOSUB650:IFCH$=""THENRETURN
ESLE Y=VAL(CH$)
3785 PRINT"HOUR: ";
3788 GOSUB650:IFCH$=""THENRETURN
ELSE H=VAL(CH$)
3790 PRINT"MINUTE: ";
3795 GOSUB650:IFCH$=""THENRETURN
ELSE MI=VAL(CH$)
3796 LSET UD$=CHR$(M)+CHR$(D)+CH
R$(Y)+CHR$(H)+CHR$(MI):PUT#1,R0
3797 RETURN
4000 '-FIGURE NAME/LOCATION
4005 A1=INSTR(UN$,CHR$(0))
4010 NA$=LEFT$(UN$,A1-1)
4015 A2=INSTR(A1+1,UN$,CHR$(0))
4020 LO$=MID$(UN$,A1+1,A2-A1-1)
4025 RETURN
4100 '-DECIMAL TO BINARY
4105 F=ASC(C$):E=128:C$=""
4110 FOR Q=1 TO 8
4115 J=INT(F/E)
4120 IF J=0 THEN C$=C$+"0"ELSEC$
=C$+"1"
4125 F=F-(E*J):E=E/2
4130 NEXT Q
4135 'PRINTF$
4140 RETURN
4200 '-BINARY TO DECIMAL
4205 W=LEN(C$):E=1:F=0
4210 FOR Q=W TO 1 STEP -1
4215 IFMID$(C$,Q,1)="1"THEN F=F+
E
4220 E=E*2:NEXTQ:C=F:C$=CHR$(C)
4225 RETURN
6000 '-ENTER USER IN LOG
6005 CLS
6010 PRINT:PRINT
6015 PRINT"NAME: ";:GOSUB650
6020 IFCH$=""THENRETURN
6025 NA$=CH$
6030 PRINT"FROM: ";:GOSUB650
6035 IFCH$=""THENRETURN
6040 LO$=CH$:IF LEN(NA$)+LEN(LO$
)+2)50THENPRINT"NAME/LOCATION TO
O LONG!":GOTO6000
6045 PRINT"PASS: ";:GOSUB650
6050 PW$=CH$
6055 PRINT"AUTHORITY: ";:GOSUB65
0
6060 IFCH$=""THENRETURNELSEIFVAL
(CH$))255THEN6055
6065 AU=VAL(CH$)
6070 PRINT"P1 FLAG: ";:GOSUB650
6075 IFCH$=""THENRETURNELSEIFLEN
(CH$)<>8THEN6070
6080 P1$=CH$
6085 PRINT"P2 FLAG: ";:GOSUB650
6090 IFCH$=""THENRETURNELSEIFLEN
(CH$)<>8THEN6085
6095 P2$=CH$
6100 PRINT"REGISTER FLAGS: ";:GO
SUB650
6105 IFCH$=""THENRETURNELSEIFLEN
(CH$)<>8THEN6100
6110 R$=CH$
6115 PRINT"BOARD ENTRY MENU: ";:
GOSUB650
6120 BE$=CH$
6125 PRINT"TIME OUT: ";:GOSUB650
6130 IFCH$=""THENRETURNELSE IFVA
L(CH$))255THEN6125
6135 TI$=CH$
6140 PRINT"SYSTEM MENU: ";:GOSUB
650
6145 IFLEN(CH$))4THEN6140ELSE SM
$=CH$
6200 PRINT"OK TO SAVE (Y/N)? ";
6205 GOSUB650:IFCH$="Y"THEN6210E
LSEIFCH$="N"THEN6000ELSE6205
6210 K1=LOF(1):LSET UM$=MKN$(0):
LSET U7$=CHR$(0):LSET U8$=CHR$(0
):LSET UL$=CHR$(0)
6215 LSET UN$=NA$+CHR$(0)+LO$+CH
R$(0):LSET UP$=PW$+STRING$(8,0)
6220 C$=R$:GOSUB4200:LSET UR$=CH
R$(C):LSET UA$=CHR$(AU)
6225 C$=P1$:GOSUB4200:LSET U1$=C
HR$(C):C$=P2$:GOSUB4200:LSET U2$
=CHR$(C)
6230 LSET UE$=CHR$(VAL(BE$)):LSE
T U5$=CHR$(0):LSET UU$=""
6235 LSET UM$=CHR$(0):LSET UD$=S
TRING$(5,0):LSET U3$="":LSET U4$
=""
6240 LSET UO$=CHR$(VAL(TI$)):LSE
T US$=SM$
6245 LSET SP$=""
6275 K1=LOF(1)
6280 K1=K1+1:PUT#1,K1
6285 IFK1<2THEN6280
6290 R0=K1:RETURN
7000 '-QUIT-
7005 CLOSE:CLS:PRINT:PRINT:PRINT
7010 PRINTTAB(5)"1-RESTART
7015 PRINTTAB(5)"2-RETURN TO BBS
"
7020 PRINTTAB(5)"3-EXIT TO DOS"
7025 '
7030 GOSUB600
7035 IF VAL(CH$))3THEN7030
7040 ON VAL(CH$)+1 GOTO7005,7045
,7050,7055
7045 RUN
7050 LOAD"COBBS/SYS",R
7055 PRINT"     ACCESS: ";:GOSUB
675:IFCH$="REBOOT CLEARED" THEN
POKE&HAC7D,&HA3:POKE&HAC7E,&H90:
END ELSE RUN
8000 '-PRINT LISTING-
8005 '
8010 IFSC=0THENPRINT"USER #";R0-
1:TB=0 ELSE PRINT#SC,"USER #";U-
1:TB=10
8020 GOSUB4000:PRINT#SC,TAB(0)"A
) ";NA$:PRINT"B) ";UP$
8025 PRINT#SC,TAB(TB)"C) ";LO$
8030 C$=UR$:GOSUB4100:PRINT#SC,"
```

```
D) ";C$:RF$=C$
8035 PRINT#SC,TAB(TB)"E) ";ASC(U
A$);TAB(TB+16)"F) ";CVN(U5$):PR=
ASC(UA$):LG=CVN(U5$)
8040 PRINT#SC,TAB(TB)"G) ";:C$=U
1$:GOSUB4100:PRINT#SC,C$;:F1$=C$
:PRINT#SC,TAB(TB+16)"H) ";:C$=U2
$:GOSUB4100:PRINT#SC,C$:F2$=C$
8045 PRINT#SC,TAB(TB)"I) ";ASC(U
E$);TAB(TB+16)"J) ";US$:BC=ASC(U
E$)
8050 PRINT#SC,TAB(TB)"K) ";TAB(T
B+16)"L)";ASC(UL$)
8055 PRINT#SC,TAB(TB)"M) ";ASC(U
O$);TAB(TB+16)"N) ":TM=ASC(UO$)
8060 PRINT#SC,TAB(TB)"O) ";CVN(U
M$);TAB(TB+16)"P) ";ASC(U7$)*256
+ASC(U8$)
8065 PRINT#SC,"Q) LAST FLAG: ";A
SC(LEFT$(UD$,1));"/";ASC(MID$(UD
$,2,1));"/";ASC(MID$(UD$,3,1));"
  ";ASC(MID$(UD$,4,1));":";ASC(M
ID$(UD$,5,1)):PRINT#SC,""
8070 RETURN
9200 '-OPEN USERLOG-
9205 FS$="USERL/SYS"
9210 OPEN"D",#1,FS$,96
9215 FIELD#1,50 AS UN$,8 AS UP$,
1 AS UR$,1 AS UA$,1 AS U1$,1 AS
U2$, 1 AS UU$,1 AS UE$,5 AS U5$,
5 AS UM$,5 AS UD$,1 AS U3$,1 AS
U4$,1 AS UO$,1 AS UL$,4 AS US$,1
 AS U7$,1 AS U8$,7 AS SP$
9220 K1=LOF(1):RETURN
9225 FIELD#1,96 AS PU$:GOTO9220
9300 '-OPEN USERL/BAK-
9305 FB$="USERL/BAK:"+DR$(D)
9310 OPEN"D",#2,FB$,96
9315 FIELD#2,96 AS U$
9320 '
9325 K2=LOF(2):RETURN
9700 '-CD CHECK-
9705 IFPEEK(4658)=0THEN9740
9710 CD=PEEK(65385)AND32
9715 IFCD<>0ORPEEK(4657)<>0THEN9
750
9740 RETURN
9750 LOAD"USER/SYS",R
9800 '
9805 GOTO625
9815 '-FLAG INPUT-
9820 F$=""
9825 FOR X=1 TO 8
9830 GOSUB600:F$=F$+CH$:PRINTCH$
;:NEXTX
9835 PRINT:RETURN
9850 '-LINE ENTRY-
9855 LINEINPUTCH$
9860 RETURN
```

```
1050 .....6
1235 .....23
1345 .....41
9120 .....203
END .....57
```

Listing 4: SMP EDI

```
0 '-COBBS (SMP/EDI) (C)1984
    BY RICHARD DUNCAN
5 CLEAR5000
10 REM
100 GOTO1000
600 LINEINPUTCH$
605 RETURN
650 '-LINE INPUT-
655 LINEINPUTCH$
660 RETURN
1000 '-PURGE MESSAGE BAS-
1005 '
1010 '
1015 PRINT"READY TO PURGE?";:GOS
UB650
1020 IFLEFT$(CH$,1)="Y"THEN1025E
LSEEND
1025 PRINT:PRINT:PRINT:PRINT"HDR
/SYS DRIVE: ";:GOSUB600:HS=VAL(C
H$):PRINTHS
1030 PRINT"HDR/BAK DRIVE: ";:GOS
UB600:HB=VAL(CH$):PRINTHB
1035 PRINT"MSG/SYS DRIVE: ";:GOS
UB600:MS=VAL(CH$):PRINTMS
1040 PRINT"MSG/BAK DRIVE: ";:GOS
UB600:MB=VAL(CH$):PRINTMB
1042 PRINT"MINIMUM MESSAGE: ";:G
OSUB650:MG=VAL(CH$)
1045 PRINT:PRINT"  CORRECT? ";:G
OSUB650
1050 IFLEFT$(CH$,1)="Y"THEN1060E
LSE1000
1060 CLS:PRINT:PRINT:PRINT"PURGI
NG HEADERS"
1065 GOSUB9000:GOSUB9100
1100 '-PURGE HEADER-
1105 GET#1,1:LSET I1$=H1$:LSET I
2$=H2$:LSET I3$=H3$:PUT#2,1:K2=1
1110 FOR R=2 TO K1
1115 GET#1,R:N=CVN(H1$):PRINT"RE
CORD #";N;
1120 GOSUB8000
1125 IFMID$(FL$,2,1)="1" OR N<MG
 THENPRINT"<DELETED)":GOTO1140
1130 K2=K2+1:LSET I1$=H1$:LSET I
2$=H2$:LSET I3$=H3$
1135 PUT#2,K2:PRINT
1140 NEXT R
1145 CLOSE:PRINT:PRINT"MESSAGE H
EADERS PURGED."
1200 '-PURGE MESSAGE BASE-
1205 PRINT:PRINT"PURGING MESSAGE
 BASE"
1210 GOSUB 9150:RX=LOF(2):CLOSE
1215 FOR R=2 TO RX
1220 GOSUB9150:GET#2,R
1225 RS=CVN(H6$):RE=CVN(H7$):PRI
NT"MESSAGE #";CVN(H1$);
1230 CLOSE:GOSUB9200:GOSUB9300:R
1=K2+1
1235 FOR A=RS TO RE
1240 GET#1,A:LSET MX$=MG$
1245 K2=K2+1:PUT#2,K2
1250 NEXT A:R2=K2:CLOSE
1255 GOSUB9150:GET#2,R:LSET H6$=
MKN$(R1):LSET H7$=MKN$(R2)
1260 PUT#2,R:CLOSE:PRINT:NEXT R
1270 PRINT:PRINT"VERIFYING FILES
":PRINT"   HDR/BAK";:GOSUB9100
1275 FOR A=1 TO LOF(2):GET#2,A:N
EXTA:CLOSE:PRINT
1280 PRINT"   MSG/BAK";:GOSUB930
0
1285 FOR A=1 TO K2:GET#2,A:NEXT:
CLOSE:PRINT:PRINT
1300 '-KILL OLD FILES-
1305 '
1310 '
1315 PRINT:PRINT:PRINT"KILL AND
COPY FILES? ";
1320 GOSUB600:IF CH$<>"Y"THEN EN
D
1325 PRINT"YES":PRINT:PRINT
1330 PRINT"KILLING OLD HEADER";
1335 FH$="HDR/SYS:"+RIGHT$(STR$(
HS),1):KILL FH$:PRINT
1340 PRINT"KILLING OLD TEXT";
1345 FM$="MSG/SYS:"+RIGHT$(STR$(
MS),1):KILL FM$:PRINT
1350 PRINT"COPYING NEW HEADER";
1355 HB$="HDR/BAK:"+RIGHT$(STR$(
HB),1):COPY HB$ TO FH$:PRINT
1360 PRINT"COPYING NEW BASE";
1365 MB$="MSG/BAK:"+RIGHT$(STR$(
MB),1):COPY MB$ TO FM$:PRINT
1370 PRINT:PRINT
1390 END
8000 '-DECIMAL TO BINARY-
8005 '
8010 F=ASC(H2$):E=128:FL$=""
8015 FOR Q=1 TO 8
8020 J=INT(F/E)
8025 IF J=0 THEN FL$=FL$+"0"ELSE
 FL$=FL$+"1"
8030 F=F-(E*J):E=E/2
8035 NEXT Q:RETURN
```

continued on Page 62.....

GRAPHS

(PLEASE NOTE: You need the Delbourgo's Expanded Color Basic to be able to use this program. G.)

So many of you have purchased EXPANDED COLOR BASIC that, after consultation and agreement with Graham, we thought it might be a good idea to include an occasional program written in this enhanced language. In this way, many of you will be able to benefit from your aquisition and perhaps you may be encouraged to submit a program yourselves to prove the good use you have made of it.

This is a program to kick off, so to speak. It makes use of the graphic and *EXTRA commands and a few other things, like scrolling and boarders. But best of all it uses the facility of being able to write on graphic screens without painful DRAW strings of the ordinary EXTENDED BASIC. Indeed the whole accent of the program is on graphs with captions; four options are available:

1. Bar Graphs or Histograms,
2. Pie Graphs,
3. Charts, &
4. Function Plots.

If you go through the Listing you should be able to adapt the program to suit your own needs, though the construction is rather general and allows for a variety of inputs on your part.

_ The emphasis being on EXP, let us highlight a few novel features in the Listing. We have used FILL and BORDER commands to make an interesting Title Card. Then we have moved to writing on the graphic screen in various PMODEs. At the end of each option we used PSCRolling before returning to the main menu. But perhaps the most interesting thing to people who are already familiar with EXP's syntax is that we have applied the power of the DOT command (see Lines 1170 and 1190) to allow one to enter any function as a string and then execute the string! In this respect EXP is quite unique.

Two final points. You will of course need to run "G" before anything and you will also have to load the EXTRA commands so as to enable the DOT command afterwards. We hope this EXPANDED offering will stimulate you to more exhilarating programming and we shall look forward to reading other submissions.

THE LISTING:

```
1 'GRAPHS IN EXPANDED BASIC BY D
aniel and Bob Delbourgo, Oct. 19
85
2 GOTO10
3 CSAVE"GRAPHS"
4 IFINKEY$=""THEN4
9 'REMEMBER TO RUN"G" AND LOAD *
EXTRA BEFORE RUNNING THIS PROGRA
M
10 TEXTON:FILL1024,1535,254
15 B=142
20 BORDER(0,0)-(31,15),B
30 PRINT@68,"A GRAPHICAL EXHIB
ITION";:PRINT@132,"IN EXPANDED
COLOR BASIC";
40 PRINT@228,"DANIEL AND BOB DEL
BOURGO";
50 PRINT@324,"HIT ANY KEY TO
 START";
90 IFINKEY$=""THENB=B+16:IFB=270
THENB=142
92 FORT=1TO50:NEXTT:IFINKEY$=""T
HEN20
93 FORJ=0TO7:BORDER(J,J)-(31-J,1
5-J),255-16*J:FORT=1TO90:NEXTT,J
100 TEXTOFF:PMODE3,1:SCREEN1,1:P
CLS:COLOR6:PRINT@32,"GRAPHICAL M
ENU":COLOR7
110 PRINT:PRINT"(1) Bar Graphs":
PRINT"(2) Pie Graphs":PRINT"(3)
Charts":PRINT"(4) Functions"
120 COLOR8:PRINT:PRINT"PRESS NUM
BER KEY"
130 I$=INKEY$:IFI$=""THEN130
140 I=INSTR("1234",I$):IFI=0THEN
130 ELSE ON I GOTO200,500,800,11
00
200 PRINT@0,"":PCLS:COLOR8:PRINT
"BAR GRAPHS":PRINT:COLOR6:PRINT"
Heading":COLOR7:INPUTH$:COLOR6:P
RINT"Number of items along X-axi
s":COLOR7:INPUTXN:XN=INT(XN):IFX
N<2THEN200
210 COLOR6:PRINT"Lowest point of
 range (along Y)":COLOR7:INPUTY1
:COLOR8
215 COLOR6:PRINT"Highest point o
frange (along Y)":COLOR7:INPUTY2
220 IFY2<=Y1 THENSOUND10,2:GOTO2
10
225 PRINT:DIMA(XN):FORI=1TOXN
230 COLOR8:PRINT"Item #"I;:PRINT
"-Value":COLOR7:INPUTA(I)
235 IFA(I)<Y1 OR A(I)>Y2 THEN230
240 NEXTI
245 XS=216/XN:YS=XS/2:XX=XS:YY=Y
S:IFXX>16THENXX=16:YY=8
250 PCLS:SCREEN1,0
255 FORI=0TOXN-1:Y=170-150*(A(I+
1)-Y1)/(Y2-Y1):X=XS*I+24:COLOR8:
LINE(X,Y)-(X+XS,170),PSET,B
260 LINE(X+XS,170)-(X+XS+XX,170-
YY),PSET:LINE-(X+XS+XX,Y-YY),PSE
T:LINE-(X+XX,Y-YY),PSET:LINE-(X,
Y),PSET
265 LINE(X+XS,Y)-(X+XS+XX,Y-YY),
PSET
270 FORK=2TOXS-1STEP2:COLOR6:LIN
E(X+K,Y+1)-(X+K,169),PSET:NEXTK
275 FORK=2TOXX-1STEP2:COLOR7:LIN
E(X+XS+K,Y-K/2)-(X+XS+K,170-K/2)
,PSET:NEXTK
280 FORK=1TOXS-1STEP2:COLOR8:LIN
```

```
E(X+K,Y-1)-(X+XX+K,Y-YY+1),PSET:
NEXTK
285 NEXTI
290 PRINT@15,Y2;:PRINT@239,Y1;
300 LINE(24,20)-(240,170),PSET,B
:LINE(240,170)-(240+XX,170-YY),P
SET:LINE-(240+XX,20-YY),PSET:LIN
E-(24+XX,20-YY),PSET:LINE-(24,20
),PSET:LINE(240,20)-(240+XX,20-Y
Y),PSET
310 PRINT@4,H$:PRINT@246,"ITEMS"
;:FORI=1TO6:PRINT@264+16*I,MID$("
VALUES",I,1);:NEXTI
320 IFINKEY$=""THEN320
330 FORI=1TO24:PSCR(0,0)-(31,23)
,U:PSCR(0,24)-(31,47),D:NEXTI
340 TEXTOFF:RUN100
500 PCLS:PMODE3,1:COLOR8:PRINT@3
,"PIE GRAPHS":PRINT
510 COLOR6:PRINT"Input Heading":
COLOR7:INPUTH$:PRINT
520 COLOR6:PRINT"How many entrie
s";:COLOR7:INPUTXN:XN=INT(XN):IF
XN<2THEN520
525 II=XN:A=0:DIMA(XN):DIMA$(XN)
:PRINT:FORI=1TOXN
527 COLOR6:PRINT"Entry #"I;"name
":COLOR7:INPUTA$(I)
530 COLOR8:PRINT"Entry #"I;"%":C
OLOR7:INPUTA(I):IFA(I)>100THEN53
0
535 A=A+A(I):IFA>100THENII=I-1:A
$(I-1)="Other":I=XN+1:GOTO540
540 NEXTI
550 PMODE4,1:PCLS1:SCREEN1,1:CIR
CLE(160,96),72,0:PI=3.14159265
555 LINE(160,96)-(232,96),PRESET
:A=0:FORI=1TOII:A=A+A(I):B=A-A(I
)/2:LINE(160+72*COS(-2*PI*A/100)
,96+72*SIN(-2*PI*A/100))-(160,96
),PRESET
560 PRINT@32*I-32,A$(I);:LINE(4+
8*LEN(A$(I)),12*I-6)-(45*COS(-2*
PI*B/100)+160,96+45*SIN(-2*PI*B/
100)),PRESET
570 NEXTI
575 IFA<100THENB=A+2:PRINT@32*I-
32,"Other":PAINT(45*COS(-2*PI*B/
100)+160,96+45*SIN(-2*PI*B/100))
,0,0:LINE(44,12*I-6)-(45*COS(-2*
PI*B/100)+160,96+45*SIN(-2*PI*B/
100)),PRESET
580 IFINKEY$=""THEN580
590 FORJ=1TO24:PSCR(0,0)-(31,23)
,U:PSCR(0,24)-(31,47),D:NEXTJ
595 RUN100
800 PMODE4,1:PCLS:SCREEN1,0
810 PRINT@12,"CHARTS":PRINT
820 PRINT"In this example we hav
e taken  the interest rates of
a number  of countries, expresse
d as DATA statements and  transc
ribed theminto charts."
825 PRINT:PRINT"  HIT ANY KEY TO
 SEE THE CHART"
830 IFINKEY$=""THEN830
835 DATA FRANCE,15,16,16,15.5,15
.5,14.5
836 DATA BRITAIN,13,15,14,13,11.
5,11
837 DATA USA,13,15.5,14,13,10.5,
11.5
838 DATA GERMANY,10,9.5,9,9.5,8.
5,8
839 DATA JAPAN,7.5,8,7.5,8.5,8.5
,7.5
840 DATA SWITZ.,5,5.5,5,5,4.5,4.
5,4
845 PCLS1:SCREEN1,0
846 LINE(16,0)-(256,100),PRESET,
B
847 PRINT@0,"20";:PRINT@449,"0";
850 FORK=1TO6:READA$,I:FORJ=1TO5
:READD:LINE(16+48*J,100-10*D)-(-
32+48*J,100-I*10),PRESET:I=D:NEX
TJ,K
855 PRINT@58,"FRANCE";:PRINT@152
,"BRITAIN";:PRINT@177,"U.S.A";
856 PRINT@202,"GERMANY";:PRINT@3
02,"JAPAN";:PRINT@362,"SWITZERLA
ND";
857 PRINT@486,"1 9 8 2  Interest
 Rates";
860 IFINKEY$=""THEN860
870 FORJ=1TO16:PSCR(0,0)-(15,47)
,L:PSCR(16,0)-(31,47),R:NEXTJ:RU
N100
1100 PMODE4,1:PCLS1:SCREEN1,1
1110 PRINT"Enter the function of
 x":INPUT F$
1120 INPUT"Minimum value of x";X
1
1130 INPUT"Maximum value of x";X
2
1135 IFX2<=X1 THENSOUND1,10:GOTO
1130
1140 INPUT "Minimum value of y";
Y1
1150 INPUT"Maximum value of y";Y
2
1155 IFY2<=Y1 THENSOUND1,10:GOTO
1150
1160 PMODE4,1:PCLS1:SCREEN1,1
1170 X=X1:LINE(16,20)-(256,180),
PRESET,B:."Y="+F$:VV=160*(Y-Y2)/
(Y1-Y2)+20:IFVV>180THENVV=180ELS
EIFVV<20THENVV=20
1175 PRINT@8,F$;:PRINT@496,"x";:
PRINT@480,X1;:PRINT@508,MID$(STR
$(X2),2,3);:PRINT@32,Y2;:PRINT@4
48,Y1;
1180 FORH=18TO256STEP2:X=X1+(H-1
6)*(X2-X1)/240
1190 ."Y="+F$:V=160*(Y-Y2)/(Y1-Y
2)+20
1200 IFV>180THENV=180ELSEIFV<20T
HENV=20
1210 LINE(H-2,VV)-(H,V),PRESET:V
V=V
1220 NEXTH
1230 IFINKEY$=""THEN1230
1240 FORJ=0TO15:PSCR(0,0)-(15,47
),R:PSCR(16,0)-(31,47),L:NEXTJ
1250 RUN100
```

CoCo Cat

LIFE WITH FORTH

by John Redmond

I've concluded that my past articles on Forth have been a bit on the ponderous side - no fun. Therefore, this month, a game! It is John Conway's famous Game of Life. The aim is to populate the world in just the right way. Each creature is placed in a cell of an array. In our case, this array has the dimensions 64 x 32, to match the low-res mapping of the cells. If a creature is in a cell with more than three neighbours, it smothers: if it has less than two, it dies of loneliness. Furthermore, if an empty cell has three neighbours, a creature will be born into that cell in the next world.

In coding this game, which turned out to be very easy, I've tried to show how much fun it can be to use Forth. The words have been named to give a Genesis flavour (if you're a pessimist, you might like to change them to get an Apocalyptic feeling). As always, with a Forth program, the highest-level word, in this case LIFE, should tell you a lot about the function of the program. When LIFE is invoked,

an EDICT is issued,
CREATION occurs
a cycle of JUDGEMENT and RENEWAL is repeated
until REST is obtained.

Next, we look at EDICT which, quite reasonably, spells out the rules. CREATION is your chance to play God and design a world to your liking, but (beware) you will be subjected to JUDGEMENT. This word surveys the OLDWORLD and, on the basis of where HAPPINESS is found, decides which creatures will inhabit the NEWWORLD when RENEWAL occurs. And so it continues until you (God) want a REST.

For the most part, I have kept the coding at the simple, if whimsical, level. The two arrays OLDWORLD and NEWWORLD are accessed in the simplest of ways, using OFFSET and the base address. This can be done much more elegantly, using a specially defined ARRAY word, but we can talk about that some other time.

The words used in CREATION and RESPOND are simple, but useful general tools for interactive graphics (a Forthcoming Rainbow mini-series). Try using them for your own applications.

During the 1970s, structured programming was all the rage but, using standard Pascal for instance, it can become an awful nuisance. It can become necessary to dream up all sorts of Boolean variables, like continuing, valid, complete, etc., so that the program knows whether to continue at certain points. One of the better aspects of C is the ability to make unstructured exits, with or without a returned value. Look at the word RESPOND, which decides what to do in response to a pressed key. There are six valid inputs and, for structured logic, this would require six levels of if .. else .. if .. else if .. etc.

In any language, this sort of conditional nesting is unreadable and error-prone. So, if a match is found in RESPOND, the stack is cleaned up, the appropriate action taken and an unstructured EXIT is performed. It's easier to read - and it's easier to add to or subtract from the list. Forth has MUCH better ways of making multiple decisions, but they will have to wait for another article. If you're impatient, The keyboard interpreter of th A*FORTH editor uses a decision array to make sense of more than 30 choices. Another way is with a CASE word. The standard IF .. ELSE .. way is the worst way, apart from spaghetti code.

The up-and-coming trendy way of making complex decisions is by means of decision tables. The word HAPPY is a very simple example. It has to decide whether a cell in the OLDWORLD is happy on the basis of its number of nearest neighbours. The rules are:

if the cell is occupied (STATUS=1)
and it has 2 or 3 neighbours, then
the cell LIVES in the NEWWORLD;
if the cell is empty (STATUS=0) and
it has 3 neighbours, then it LIVES
in the NEWWORLD;
otherwise the cell will be empty in the NEWWORLD.

The structured logic is terrible. Look at NEIGHBOURS. Using nested loops, it count the occupied cells in a 3 x 3 matrix centred on the cell in question. The count is sometimes inaccurate. If the cell is occupied (STATUS=1), NEIGHBOURS returns the true count + 1 (the maximum count is therefore 9). Now HAPPINESS is a 2 x 9 array. The first nine elements are valid for STATUS=0 (i.e., the offset is 9 x 0); the elements for STATUS=1 have an offset of (9 x 1= 9). All HAPPY then needs to do is add the further offset (number of neighbours) and return the byte value (1= happy and 0= unhappy). JUDGEMENT then takes the appropriate action. Look at how simple and fast the code is for HAPPY and note that we can introduce an entirely different set of rules for HAPPINESS simply by changing the pattern of 0 and 1 values in its array. Forth at its best and another chance to play God.

This version of Life was dashed off pretty quickly and it can be made to go significantly faster. (How? Where is all the time taken up?) And it wastes a lot of memory for the arrays. You see, memory has been set aside for two worlds. Each is a byte array, yet only one bit in each byte has been used (bit 0). It is a simple step to use a bit array (which would take only one-eighth of the space). Think about how you would do this. The words that have to be changed are the array access words, INSERT, ZAP, STATUS, etc., and the words they invoke. Another, more complicated, approach is to use a single array which would have different (non-zero) values which code the fates of the different cells in the next world.

To show the syntactic flexibility of Forth, I've put in some quite unnecessary fun words like TRANSFORM and DARKEN. It's quite a language.

Each cycle of the Game of Life takes about 21 seconds. As I said, there is scope for improvement but I'm sure that it is much faster than Basic (does anyone have a timing for arrays of the same size?).

```
\ THE GAME OF LIFE
\ WRITTEN BY JOHN REDMOND
\ IN A*FORTH 23/11/85

HEX
88 CONSTANT CURSOR
400 CONSTANT SBASE
200 CONSTANT SSIZE
40 CONSTANT ARRAYWIDTH
ARRAYWIDTH 1- CONSTANT XMAX
20 CONSTANT ARRAYHEIGHT
ARRAYHEIGHT 1- CONSTANT YMAX
ARRAYWIDTH ARRAYHEIGHT *
CONSTANT WORLDSIZE

7000 CONSTANT OLDWORLD
7800 CONSTANT NEWWORLD

VARIABLE NEAR
VARIABLE (MARG)
VARIABLE COLOUR 80 COLOUR !
CREATE POINTER 4 ALLOT

: LINE ( #) 0F MIN 0 MAX
20 * SBASE + CURSOR ! ;

: MARGIN ( #) (MARG) ! ;

: COLUMNS ( #)
CURSOR @ FFE0 AND + CURSOR ! ;
```

```
: MLINE LINE (MARG) @
COLUMNS ;
  •.
: WAIT INKEY DROP ;

: OFFSET ( X,Y--ADDR)
ARRAYWIDTH * + ;

: OLDSPOT ( X,Y--ADDR)
OFFSET OLDWORLD + ;

HERE 8 C, 4 C, 2 C, 1 C,

: MASK LITERAL + C@ ;

: MAP ( X,Y--BYTE,ADDR) DUP 2/
5 << ROT DUP 2/ ROT + SBASE +
>R ( ADDRESS) 1 AND SWAP
( SAVE XOFFSET) 1 AND 2* +
( ADD TO YOFFSET) MASK R> ;

: (PIXEL) ( BYTE,ADDR) DUP C@
ROT OR SWAP C! ;

: PIXEL ( X,Y) MAP (PIXEL) ;

: (-PIXEL) ( BYTE,ADDR) DUP C@
ROT FF XOR AND SWAP C! ;

: -PIXEL ( X,Y) MAP (-PIXEL) ;

: STATUS OLDSPOT C@ ;

: ?REMOVE ( X,Y) 2DUP STATUS
NOT IF -PIXEL ELSE 2DROP
THEN ;

: COORDS POINTER 2@ 2DUP
?REMOVE ;

: (UP) 1- 0 MAX ;

: (DOWN) 1+ YMAX MIN ;

: (LT) 1- 0 MAX ;

: (LEFT) SWAP (LT) SWAP ;

: (RT) 1+ XMAX MIN ;

: (RIGHT) SWAP (RT) SWAP ;

: CHANGEPOINTER ( X,Y)
2DUP POINTER 2! PIXEL ;

: UP COORDS (UP)
CHANGEPOINTER ;

: DOWN COORDS (DOWN)
CHANGEPOINTER ;

: LEFT COORDS (LEFT)
CHANGEPOINTER ;

: RIGHT COORDS (RIGHT)
CHANGEPOINTER ;

: INSERT ( X,Y) OLDSPOT 1 SWAP
C! ;

: ZAP ( X,Y) OLDSPOT 0 SWAP
C! ;

: KEEP POINTER 2@ INSERT ;

: REMOVE POINTER 2@ ZAP ;

CREATE HAPPINESS 12 ALLOT
HAPPINESS 12 0 FILL
1 HAPPINESS 3 + C!
1 HAPPINESS 0C + C!
1 HAPPINESS 0D + C!

: CHANGECOLOUR COLOUR @ 10 +
FF AND 80 MAX COLOUR !-;

: WALL SBASE SSIZE ROT FILL ;

: TRANSFORM WORLDSIZE CMOVE ;

: DARKEN WORLDSIZE 0 FILL ;

: COLOURED COLOUR @ ;

: DAWN COLOURED WALL YMAX 0 DO
XMAX 0 DO I J STATUS IF I J
PIXEL THEN LOOP LOOP ;

: RENEWAL NEWWORLD OLDWORLD
TRANSFORM NEWWORLD DARKEN
DAWN CHANGECOLOUR ;

: REST INKEY 3 = ;

: HAPPY ( #,X,Y--T/F) STATUS 9
* + HAPPINESS + C@ ;

: LIVES ( X,Y) OFFSET NEWWORLD
+ 1 SWAP C! ;

: NEIGHBOURS ( X,Y--#) 0 NEAR
! DUP (DOWN) 1+ SWAP (UP)
POINTER 2! DUP (RT) 1+ SWAP
(LT) DO POINTER 2@ DO J I
STATUS NEAR +! LOOP LOOP NEAR
@ ;

: JUDGEMENT YMAX 0 DO XMAX 0
DO I J NEIGHBOURS I J HAPPY IF
I J LIVES THEN LOOP LOOP ;

: RESPOND ( CHAR) DUP ASCII K
= IF DROP KEEP EXIT THEN DUP
ASCII R = IF DROP REMOVE EXIT
THEN DUP ASCII @= IF DROP UP
EXIT THEN DUP 0A = IF DROP
DOWN EXIT THEN DUP 9 = IF DROP
RIGHT EXIT THEN 8 = IF LEFT
THEN ;

: CREATION 80 COLOUR !
COLOURED WALL 20 10 2DUP
POINTER 2! PIXEL OLDWORLD
DARKEN NEWWORLD DARKEN BEGIN
INKEY DUP 3 - WHILE RESPOND
REPEAT DROP ;

: EDICT PAGE 7 MARGIN
1 MLINE ." LIFE WITH A*FORTH"
2 MLINE ." BY JOHN REDMOND."
4 MLINE ." MOVE THE CURSOR"
5 MLINE ." WITH THE ARROW KEYS;"
6 MLINE ." KEEP OR REMOVE A"
7 MLINE ." CELL WITH K OR R &"
8 MLINE ." STOP WITH <BREAK>"
0A MLINE ." AFTER EACH ROUND,"
0B MLINE ." YOU CAN ABORT WITH"
0C MLINE ." <BREAK> OR GO ON"
0D MLINE ." WITH ANOTHER KEY."
WAIT ;

: LIFE
EDICT
CREATION
BEGIN JUDGEMENT RENEWAL
REST UNTIL ;

\ ROUTINES FOR KEEPING AND
\ DISPLAYING TIME IN SECONDS.
\ THEY USE THE 50 HZ TIMER
\ OF EXTENDED BASIC.
\ BY JOHN REDMOND 9/11/85

HEX
112 CONSTANT (TIME)

: TIME ( -- INTEGER)
( USED TO INITIALIZE TIMER)
(TIME) ! ;

: .TIMER (TIME) @ 0 2DUP D+
( CONVERT TO LONG INTEGER)
( AND DOUBLE IT TO GIVE THE)
( TIME IN 1/100THS OF SECS.)
<# # # ASCII . HOLD #S #>
TYPE SPACE ." SECONDS" ;

\ EXAMPLE OF USE:
\ 0 TIME (other code) .TIMER

DECIMAL

: WTEST 0 DO ( test word)
LOOP ;
: TESTS ( NUMBER) 0 TIME 0 DO
1000 WTEST LOOP .TIMER ;

\ THE test word CAN BE ANY
\ WORD THAT YOU WISH TO TIME.
\ NOTE, THOUGH, THAT THE STACK
\ ALIGNMENT MUST BE MAINTAINED,
\ OTHERWISE STACK OVERFLOW OR
\ UNDERFLOW WILL OCCUR.

\ THEN TYPE IN:
\ (e.g.) 1000 TESTS <CR>

\ SIMPLE FORTH WORDS
\ FOR INSERTION OF ACCURATE
\ TIMING LOOPS INTO PROGRAMS
\ BY JOHN REDMOND, 9/11/85

DECIMAL
: SECONDS ( #)
0 DO
15789 0 DO LOOP
LOOP ;

\ THE VALUE OF 15789 IS QUITE
\ ACCURATE FOR MY COCO.  YOURS
\ MAY NEED A SLIGHTLY DIFFERENT
\ VALUE.

: MINUTES ( #)
0 DO
60 SECONDS
LOOP ;

: HOURS ( #) 0 DO 60 MINUTES
LOOP ;

\ USAGE: (E.G.)
\ 1 HOURS 7 MINUTES 20 SECONDS
\ (THIS WILL INTRODUCE THE
\ APPROPRIATE WAIT LOOP.)
```

INPUT / OUTPUT DRIVER ROUTINE

by Geoff Fiala

(To utilise this program you will require one of our CoCoConnections - the unit which allows you to interface your computer to the real world.
The model railway project is progressing nicely - further news and hints next month! G.)

This software routine provides the following features:
a) allows the user to specify the initializing of each of the 4 user ports 1A,1B,2A and 2B as an input or output port
b) For any port programmed (initialized) as an output, the user can set an output condition on any port line without affecting the output condition of the other port lines.
c) For any port programmed (initialized) as an input, the user can check the input status of any port line.

The programme is written in the form of a demonstration programme using BASIC LANGUAGE subroutines to perform the functions outlined above. This allows the user to adapt these for their own particular programme requirement. In doing this all that is required in most cases would be a changing of the keyboard input statements to BASIC assignment statements eg.

line 30---- PRINT "ENTER I/O MODULE BASE ADDRESS :"
line 31---- INPUT ADR

could be replaced by the following:

line 30---- ADR = 65408

The features of the programme are best demonstrated by connecting the LED Display Board supplied with the COCO Connection to one of the Ports and connecting the Push Button Switch to function as an input source. This should be done before loading the software

USING THE PROGRAMME

a) When the prompt "ENTER I/O MODULE BASE ADDRESS" is displayed, enter 65408. This is the default base memory address setting of the CoCo Connection. If you are using a different base address then enter it instead.
b) Next the programme requests the configuration for each input port. Enter "1" if you want the port to be configured as inputs or "0" if you want it configured as outputs. All 4 ports are to be configured.
c) After configuring all the ports the next prompt asks whether you want to set an output condition or check for an input condition. Enter "1" to check for an input condition or "0" to set an output condition.
d) Next you are asked to " ENTER INPUT/OUTPUT STRING COMMAND" as follows:
i) to set outputs: Enter a 4 character string as follows eg

```
2BL1L

PORT NO 1A,1B,2A,2B--111----OUTPUT LINE CONDITION
                      I     L= logic LOW
PORT LINE NO. 1-8-----I     H= logic HIGH
```

The above string specifies that Port 2B, line 1 is to be set "LOW". If the LED dispaly board is wired to this port,the LED connected to line 1 will be turned "ON". Similarly if the next Output Command String is:

"2B8L"

then the LED connected to line 8 on Port 2B will also be turned "ON".

ii) To check for an input : Enter a 3 character string as follows eg:

```
2A6

PORT NO. 1A,1B,2A,2B--11---PORT LINE NO. 1-8
```

The above string specifies a check for an input on Port 2A, Line 6. If the Push Button Switch was connected to this Port Line and and is held down to simulate an input and the above command string is entered , then an "INPUT DETECTED" message will be displayed.

Similarly if the next Input Command string is :

"1A1"

then PORT line 1 on Input Port 1A will be checked for an Input.

No error checking has been included to check for the correct input of the Command String. Therefore if you enter an incorrect string, just re-enter the correct string on the next command.

PROGRAMME STRUCTURE

As was mentioned previously the programme has been written using subroutines for the various functions. These are briefly described below:

Lines 300-340
The initialization routine for Ports designated as Inputs

Lines 400-450
The initialization routine for Ports designated as Outputs

These two routines are general purpose subroutines and require that the Variable "A" = the memory address of the Port to be initialized . This is set-up in lines 32-80

Lines 3000-3030
These determine the Port memory address form the entered command string.

Lines 3100-3230

This subroutine further decodes the command string A$ to determine the Port line number and Output condition. It also works out the correct mask byte pattern to mask out the other port lines and only changes the line specified . As the position of the port lines for ports 1A and 1B are the reverse of Ports 2A and 2B two different Mask byte patterns are required and these are calculated as well.

Lines 4000-4090

This subroutine contains the Input check routine , and decodes the command string A$ to find the Port and Port line number to be checked. It also works the required Mask byte pattern such that an input condition for the line specified will only be checked. All other port lines are masked or blocked out.

The numeric variable "IP" will equal zero if an input condition (logic LOW) was detected., otherwise "IP" will be a non-zero value.

Therefore on return from this subroutine line 180 checks the value of "IP" for a zero value.

Using the above routines and changing the keyboard input statements to BASIC assignment statements, one could write a programme to check certain inputs for a "ON" or "OFF" condition and depending on this turn "ON" audible alarms or buzzers. This could form the basis of a burgular alarm system using the CoCo Connection and the Color Computer.

The Listing:

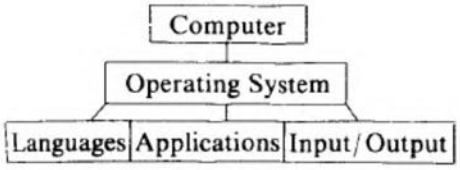
```
 1 CLS
 2 PRINT"INPUT/OUTPUT DRIVER FOR
 THE COCO CONNECTION.
 3 REM WRITTEN BY GEOFF. FIALA --
 -REV 01-10/12/85
 5 CLS
 10 PRINT "GENERAL PURPOSE INITIA
 LIZATION ROUTINE"
 20 PRINT
 30 PRINT "ENTER I/O MODULE BASE
 ADDRESS :"
 31 INPUT ADR
 32 REM CONFIGURE PORT 1A
 35 A$="1A":A=ADR:GOSUB 200
 40 REM CONFIGURE PORT 1B
 45 A$="1B":A=ADR+2: GOSUB 200
 50 REM CONFIGURE PORT 2A
 60 A$="2A":A=ADR+4: GOSUB 200
 70 REM CONFIGURE PORT 2B
 80 A$="2B":A=ADR+6: GOSUB 200
 90 CLS: PRINT" ALL PORTS INITIAL
 IZED"
 95 A=ADR
 100 PRINT "ENTER CONDITION TO BE
  SET/CHECKED"
 105 TIMER=0
 110 PRINT "I=INPUT"
 120 PRINT "O=OUTPUT"
 130 INPUT"ENTER 'I' OR 'O' :";B$
 140 PRINT "ENTER INPUT/OUTPUT ST
 RING COMMAND":PRINT:INPUT A$
 150 IF B$="I" THEN 170 ELSE IF B
 $="O" THEN 160 ELSE GOTO 100
 160 GOSUB 3000:REM GET PORT ADDR
 ESS
 162 GOSUB 3100:REM SET OUTPUT CO
 NDTION
 164 PRINT "OUTPUT CONDTION SET"
 165 GOTO 190
 170 GOSUB 4000:REM CHECK INPUT C
 ONDITION
 180 IF IP=0 THEN PRINT"***INPUT
 DETECTED***" ELSE PRINT "***NO I
 NPUT DETECTED***"
 190 SOUND 200,5:GOTO 100
 200 CLS:PRINT @192,"ENTER INITIA
 LIZATION FOR PORT ";A$
 210 PRINT "I = INPUTS"
 220 PRINT "O = OUTPUTS"
 230 PRINT
 240 PRINT "ENTER LETTER (I OR O)
  :":INPUT B$
 250 IF B$="I" THEN 300 ELSE IF B
 $="O" THEN 400
 260 CLS:GOTO 200 :REM NO MATCH
 300 REM INPUT INITIALIZATION ROU
 TINE
 310 REM A=PORT ADDRESS
 320 REM ALSO INITIALIZES BUFFERS
  AS INPUTS IF INSTALLED.
 330 POKE A+1,0:POKE A,0:POKE A+1
 ,60
 340 RETURN
 400 REM OUTPUT INITIALIZATION RO
 UTINE.
 410 REM INITIALIZES BUFFERS AS O
 UTPUTS IF INSTALLED.
 420 REM A=PORT ADDRESS.
 430 REM ALL OUTPUTS WILL BE PROG
 RAMMED TO A LOGIC "1" AFTER INIT
 IALIZATION
 440 POKE A+1,4:POKE A,255:POKE A
 +1,0:POKE A,255:POKE A+1,52
 450 RETURN
 3000 REM FIND PORT  ADDRESS
 3010 AD$=LEFT$(A$,2)
 3020 IF AD$="1A" THEN ADR=A ELSE
  IF AD$="1B" THEN ADR=A+2 ELSE I
 F AD$="2A" THEN ADR=A+4 ELSE IF
 AD$="2B" THEN ADR=A+6
 3030 RETURN
 3100 REM GET OUTPUT LINE NUMBER
 AND ADJUST
 3120 N=ASC(MID$(A$,3,1))-49
 3130 REM GET OUTPUT LINE SETTING
 3140 IF RIGHT$(A$,1)="H" THEN 31
 90
 3150 REM SET BIT PATTERN TO TURN
  PORT LINE "ON" (LOW)
 3151 IF LEFT$(A$,1)="1" THEN 316
 0
 3152 REM ADJUST BIT PATTERN FOR
 PORT 2
 3153 V =255-INT(2^(7-N))
 3154 GOTO 3170
 3160 V=255-INT(2^N)
 3170 POKE ADR,(PEEK(ADR) AND V)
 3180 RETURN
 3190 REM SET BIT PATTERN TO TURN
  PORT LINE "OFF" (HIGH)
 3200 IF LEFT$(A$,1)="1" THEN 321
 0
 3201 REM ADJUST BIT PATTERN FOR
 PORT 2
 3202 V =INT(2^(7-N))
 3203 GOTO 3220
 3210 V =INT(2^N)
 3220 POKE ADR,(PEEK(ADR) OR V):R
 EM SET OUTPUT LINE
 3230 RETURN
 4000 REM FIND PORT ADDRESS
 4010 GOSUB 3000
 4020 REM GET INPUT PORT LINE NUM
 BER AND ADJUST
 4030 N=ASC(MID$(A$,3,1))-49
 4040 REM SET BIT PATTERN MASK FO
 R CHECKING INPUT LINE
 4050 IF LEFT$(A$,1)="1" THEN 406
 0
 4051 REM ADJUST BIT PATTERN FOR
 PORT 2
 4052 V=INT(2^(7-N))
 4053 GOTO 4070
 4060 V=INT(2^N)
 4070 REM CHECK INPUT LINE
 4072 IP=PEEK(ADR)
 4074 IF IP <> 255 THEN 4080 ELSE
  GOTO 4090:REM NO INPUT
 4075 IF PEEK(ADR) <> 255 THEN 40
 80 ELSE IP=255
 4080 IP=IP AND V
 4090 RETURN
```

Getting Started With The OS-9 Operating System

By Bruce Warner

Before diving into the considerable technical information required to operate under OS-9, I want to explain that I will give only the information you *need* at any given time. It's more important to know that the Kernel plays a vital part in managing the OS-9 operating system than to read 24 pages of what the Kernel does and how it manages to do what it does. For beginners, I like to keep things simple by speaking in broad terms, by saying, for instance, "the Kernel supervises the Operating System."

First off, OS-9 is an *operating system*, not a programming language, or even a program (not in the sense that it allows you to write a letter or calculate a row of figures). An operating system is what comes between the hardware and the software. It manages the software and directs the output to the hardware. That's about as simple as it can be stated. This diagram should help you see the relationship between the computer, its operating system and the applications for which it may be used:



The operating system manages program execution and computer control of hardware. By this definition, there even must be an operating system while operating under Disk Extended BASIC, and there is. The difference is that Disk Extended BASIC comes with a "transparent" operating system. It is wired into the BASIC ROM, leaving you with what appears to be a computer without an operating system. Do not let the appearance fool you. Disk Extended BASIC is a DOS (Disk Operating System).

Booting OS-9

Let's do something with OS-9. First, we'll make OS-9 control your Color Computer. To accomplish this feat, you must change from one operating system to another. This requires a machine language program to turn off the ROM for the 64K RAM mode, load the OS-9 operating system and execute the OS-9 Kernel. This is done on the Color Computer by one of two methods; the method used depends entirely on which version of Disk Extended BASIC you have. The two methods for booting are as follows:

1) For Disk Extended BASIC 1.0, place the boot disk in Drive 0 and type RUN"*". You are prompted to select one of two options. You can either boot OS-9 or test the speed of your disk drives. Type B for boot and wait for the Color Computer to tell you when to change disks and enter the OS-9 system disk.

2) If you have Disk Extended BASIC 1.1 or newer, the procedure is considerably easier. Put your system disk in Drive 0 and simply type in DOS.

Regardless of the method used, the computer sends the disk drive's read head to Track 34 to start reading any program loaded on that track. This program executes the procedures we talked about earlier, causing OS-9 to take over your Color Computer.

If you have DOS 1.0 and want to upgrade, it's a simple and inexpensive process. You do *not* have to buy a whole new disk controller. All you need is one of the DOS ROM chips, available from a number of suppliers for $39 or less.

If you decide it's time to upgrade, open your ROM pack (make sure the screw located under the controller label is removed before opening the case). Note the location of the notch on the Disk ROM chip before removing it. Remove the chip and insert the replacement chip with the new one (make sure the notch is in the same location as the one you took out). Be careful not to bend any of the pins on the chip when inserting it. Close the case and replace the screw. Job complete.

What Time is it?

When booting OS-9 you are prompted to enter the time. OS-9 is very dependent on time and uses a clock for a number of functions. The format should be entered as requested. Spaces can take the place of punctuation marks, but all other spacing is required.

The future validity of the filing system depends on your entries. If you have a real-time clock (like the one available on the PBJ PC-Pak), you'll be able to add the time setting to the start-up procedure file by entering the year in the start-up file.

If you haven't already done so, get out the *Getting Started With OS-9* booklet and make a backup copy of the OS-9 System disk. Your system master should only be used to make copies for personal use.

After entering the time, you are operating under OS-9. If you have never operated under an advanced operating environment, get ready for a unique experience.

Unlike Disk Extended BASIC, OS-9 has two directories it works from at all times — the data directory and the execution directory. When you boot OS-9, the data directory will be the ROOT directory on the OS-9 system disk. The execution directory will be contained within the CMDS directory on the same disk.

Just to make sure the screens we are viewing are set up for the same display, type in and enter `tmode -upc`. This command changes the display on the terminal (TV or monitor) from all uppercase to an upper- and lowercase display. Now your display will look the same as the examples.

To get a look at the current data directory, try typing the DIR command you may have used in Disk Extended BASIC and see what happens. After hearing your disk drive churn a bit, you should see a directory that looks like this:

```
DIRECTORY OF .   23:22:14
OS9Boot   CMDS       SYS
DEFS      startup
```

You can easily see that this is a directory, but OS-9 doesn't stop there. It continues by telling you it is a listing of the current data directory (indicated by the period). The time on your system should be different.

This looks a lot different from the directory you may be used to under Disk Extended BASIC, FLEX or StarDOS. This is your OS-9 system disk with all its ROOT directory files. Directories are special files designed to help organize a disk that may contain 100 or more programs and files. Of course, you can also incorporate files that stand on their own, like the start-up file, but that would fill up the screen in no time and waste one of the major features of an OS-9 system.

A Closer Look at the System Disk

There are five files on the OS-9 system master. They include the OS 9Boot file (this is the file we used to boot OS-9 earlier), a directory of all the system's commands available, a directory of system files (things like passwords and message of the day), a directory of established procedures used by OS-9 assembly language programs and a start-up file.

As I said earlier, OS-9 has two default directories. As soon as you have booted OS-9, the system locates the directory on Drive 0 named CMDS and makes it the current execution directory.

Why Won't It Run?

At almost every meeting of the Northern Virginia Color Computer Club, we can expect one question to come up. What I am about to say is the answer to that question. This may be the most important thing you will ever learn about OS-9. Please pay *very* close attention. OS-9 has an established. . . logical. . . order for locating programs and procedures to be executed. The order in which a procedure is searched is as follows:

1) If a complete pathlist is defined, the program/procedure is called from that pathlist and immediately executed (a complete pathlist would resemble /d0/cmds/dir).

2) If a complete pathlist is not defined, OS-9 looks for the program in memory. If the program/procedure called is located in memory, it is executed immediately.

3) The next place OS-9 looks for a program is in the current execution directory. This has been a source of confusion for a great number of people. Because of this, trying to run BASIC09 or C under OS-9 has resulted in creating more errors than programs because you have not copied BASIC09 or C from its master disk into the current execution directory. This is in large part because of the documentation you have received. If you want to find all of the programs in the current execution directory, type `dir x` and you should see a directory that looks like the following example.

```
Directory of . 06:21:22
asm         attr        backup
binex       build       cmp
cobbler     copy        date
dcheck      debug       del
deldir      dir         display
dsave       dump        echo
edit        exbin       FORMAT
free        ident       link
list        load        login
makdir      mdir        merge
mfree       os9gen      printerr
procs       pwd         pxd
rename      save        setime
shell       sleep       tee
tmode       tsmon       unlink
verify      xmode
```

This directory shows all of the procedures available from OS-9.

4) Finally, OS-9 will look for the program as a listing of batched procedures in the current data directory. A procedure is a line you type from the OS-9 prompt. Any number of procedures can be put in a procedure file. We'll go into procedure files a little later.

A Note on Control Keys

There are a number of standards that have been set in the computer industry, some of which are missing on the Color Computer. The most noteworthy is the lack of two keys, the CONTROL and ESC keys. Both of these are available in OS-9.

Since OS-9 has no need for the CLEAR key, it takes the place of the CONTROL key, and ESC is accomplished by holding down the CONTROL (CLEAR) key and pressing the BREAK key. If you ever want to get out of executing a program, the ESC is your "ESCape."

Ready to Climb a Tree?

Ready for a little tree climbing? Of course, I mean climbing the OS-9 operating system of tree directories. From here on we will discuss the directory system: how to get into them, how to get out of them, how to create them, how to tell where you are in them and how to delete them.

Just to make sure we're in the right mode, type in the command to change the terminal to the upper- and lowercase mode. If you don't remember how to change to the upper- and lowercase mode, try typing in the following command and see what happens when you enter the dir command: `TMODE -UPC`. The directory should look something like the following example.

```
Directory of .   22:53:47
OS9Boot   CMDS       SYS
DEFS      startup
```

What's in a File?

Before we go too much further, let's look at how you can tell the difference between directories and files in a *well-organized* OS-9 system. Notice that the start-up file is displayed in all lowercase, the OS9Boot file is in upper- and lowercase, and the remainder of the files is displayed in all uppercase.

The TMODE command alters the mode of the device it is directed to change (in this case the display on your TV or monitor) and the -upc means to turn off the all uppercase display. Now there is an upper- and lowercase display, and your Color Computer will

display the directories with some degree of logic, provided the rules of file naming are followed.

Here are the rules: 1) Name all directories with all uppercase characters; 2) Name other files with any combination of upper- and lowercase characters.

This is a marked improvement over a number of other operating systems (especially a certain three-letter company that claims to be the leader in the computer industry). Both FLEX/Star-DOS and MS-DOS do not permit the use of lowercase characters in filenames! FLEX/Star-DOS does not permit the use of directories, either.

Another item worth noting is that filenames in OS-9 are not limited to the traditional eight characters found in FLEX, Star-DOS, MS-DOS or Disk Extended BASIC. With OS-9 you are "limited" to names that are one to 29 characters long. Just imagine the luxury of a filename that tells exactly what's in the file!

If these rules are followed, you'll know what to expect from any filename simply by looking at its name.

Even More than You Want to Know

OK, so you're not all that organized yet. Neither are too many of the rest of us (there is a naming mistake on your original disk if you look in the CMDS directory). Therefore, OS-9 was developed with an extension of the directory command to allow a second look at any directory and find out what is in it. This time type in the following line: `dir e`. Now you should see a directory that looks very much like this one:

```
Directory of .  20:02:29

CREATED ON    OWNER   NAME
  ATTR       START      SIZE
============================
83/06/02 1921     0  OS9Boot
------wr         A       3032
83/06/02 1956     0  CMDS
d-ewrewr        3C        620

83/06/02 2002     0  SYS
d-ewrewr       164         A0
83/06/02 2002     0  DEFS
d-ewrewr       17F         C0
```

The 'e' tells OS-9 you want to know *everything* about the directory, and it tells you everything! For thoroughness, we'll look at the information for the CMDS directory. It was created on June 2, 1983 at 7:56 p.m. The owner of the file is the *super user*. The file is named CMDS. Then, we see a crazy section of the line, d-ewrewr, which we'll cover in the next paragraph. The START is the physical sector on the disk where the beginning of the file is located, and the SIZE is the number of sectors used by the file on the disk.

The eight characters used to identify the file attributes are code letters. Each column must contain either a hyphen (-), for not selected, or one specific letter. The meaning of each letter is as follows:

Column	Letter	Meaning
1	D	File is a directory of other files
2	S	File may be shared by more than one user
3	E	File may be executed by anyone
4	W	File may be written to by anyone
5	R	File may be read by anyone
6	E	File may be exected by owner and super user
7	W	File may be written to by owner and super user
8	R	File may be read by owner and super user

Value in Directories

Now that you have discovered how to identify a file, what makes these directories so valuable? Let's look at a good example of a disk used by a writer.

This example is typical of many writers in the computer industry. Many of us write for a number of publications. Because of this we need a system that is well-organized.

Making a Directory

My ROOT directory contains a number of files not found on the original OS-9 disk. In addition, I have included the following:

```
FOR_RAINBOW
FOR_SOFTNEWS
FOR_NAVY
RGS_MANUALS
PERSONAL_FILES
```

To create these files on my disk, I use the MAKDIR command. Starting from the top, I have typed the following lines. (To create the "underline character," as in FOR_RAINBOW, press the CLEAR and minus sign (-) keys simultaneously.)

```
makdir FOR_RAINBOW
makdir FOR_SOFTNEWS
makdir FOR_NAVY
makdir RGS_MANUALS
makdir PERSONAL_FILES
```

Now my directory looks like this:

```
 Directory of .  12:50:55
OS9Boot    CMDS       SYS
DEFS       startup    FOR_RAINBOW
FOR_SOFTNEWS          FOR_NAVY
RGS_MANUALS           PERSONAL_FILES
NVCCC_NEWSLETTER
```

Before you start wondering how I can put so much on one disk, I've added a five-meg hard disk drive to my 'E' board CoCo since I last wrote for RAINBOW.

Going out on a Limb

Since this is an article for RAINBOW, let's look inside the FOR_RAINBOW directory. To get inside it, we must first change our data directory. This is done by using the CHD command:

```
chd for_rainbow
```

Notice the directory name does not have to be typed in all uppercase after it has been created. This saves a lot of time and avoids a lot of mistakes.

The directory now looks like this:

```
 Directory of .  13:00:45
COVER_LETTERS          ARTICLES
REVIEWS     CONTACTS
```

Notice that these still are all directories (all capital/uppercase letters). So we'll change again. This time, change to the articles directory; type `chd articles`. This directory looks a little more useful:

```
 Directory of .  13:10:15
OS9_intro OS9_directories
SCRATCH03
```

This directory also shows one of the ways in which OS-9 uses various files. The SCRATCH03 files is not a permanent file. It is created by *DynaStar* to hold the file until a session is completed. When finished, the old file is deleted and the scratch file becomes the new file. The final product is named the same as the old file.

Now that these directories have been created, and we are somewhere down in the holes of a directory, how do we find out where we are? Let OS-9 tell us! Try typing in the following:

```
pwd
```

The command stands for "print working directory." It gives a listing of the current working directory; mine looks like this:

```
/H0/FOR_RAINBOW/ARTICLES
```

It says that I am using device H0 (the leading slash means it is a device) in the FOR_RAINBOW directory and further in the subdirectory ARTICLES.

Retracing our Steps

When an article is finished, you should always have a cover letter to go with it, so you'll want to back out of the current directory. OS-9 allows you to do this by using shorthand. The period (.) indicates one directory level. One period is the current level, two is one level higher, three is another level higher, and so on. There is no reasonable limit to the number of periods used. Now type:

```
chd . .
```

This goes back to the FOR_RAINBOW directory, which looks like this:

```
Directory of .  13:45:18
COVER_LETTERS        ARTICLES
REVIEWS     CONTACTS
```

A third period will take you up to the ROOT directory (one level higher).

Even with five-meg of online storage, everything can't be kept on the system at one time. To help solve the problem of storage, copy old files over to another disk for historical storage and delete them from the working disk.

Suppose you decide to delete a directory that's no longer needed. You can start by deleting every file in the directory, then delete the directory, or simply delete the directory. This is done with the DELDIR command followed by the name of the directory to be deleted. As an example, from inside the FOR_RAINBOW directory, type `del dir contacts`.

After answering the series of questions as prompted, you can again type the directory command and see something like the following.

```
Directory of .  14:00:27
COVER_LETTERS        ARTICLES
REVIEWS
```

With a little foresight, you'll see how a file can easily be organized. If there are several family members who use the computer, directories can be made for each member. If you have a business and also use the computer for personal use, subdirectories can be made for personal and business use.

As promised at the onset of this article, we've covered directories in the OS-9 operating system. With this information, you will be able to get a lot of mileage out of a single disk (and even more from a hard disk drive). With directories, you should now know how to get into them, get out of them, create them, tell where you are in them and delete them.

Enjoy what you have learned about getting started with OS-9. OS-9 can help make the most of your Color Computer. □

NEW OS9

by Bob Rosen

Bob Rosen from Spectrum Projects sent the following information relating to the new OS-9 Version 02.00.00 for CoCo, soon to be released by Tandy.

A. The Standard Boot for 2.00.00

IOMAN	RFB	SCF	PIPEMAN
CLOCK.60HZ	CCDISK	CC10	PIPER
	d0	C032	PIPE
	d1	TERM32	
		PRINTER	
		P	
SysGo		RS232	
Shell		T1	

B. Driver Changes for the new version.

1. CC10/TERM - the keyboard and display driver for the Colour Computer.

a. CC10 has been split into three separate modules, the main module CC10 and two co-modules for output called GRF0 and C032.

i. CC10 handles all keyboard input.

ii. C032 handles output to standard display.

iii. GRF0 handles graphics output. CC10 can still handle all non drawing functions but the GRF0 module is required for these routines to work. GRF0 is not in the standard Boot, but is in the the CMDS directory and should be loaded if needed.

b. Changes to CC10 module.

i. Auto key repeat on all keys!!!!!

ii. Keyboard beep on Control G - "display 07"

iii. '2' key is now ALT key to set high order bit the typed character. On standard display this generates semigraphics characters.

iv. Support for true lower case on machines with the new VDG.

v. Allows allocation of three graphics buffers and selection of which one is the current view buffer. Adds possibility for animation.

c. Changes to graphics routines (GRF0)

i. Erase circle routine.

ii. Flood Fill routine.

iii. Bug fixed to allow drawing in background colour in 2 colour modes.

2. Internal Serial Drivers - PRINTER/P & RS232/T1

a. User selectable baud rate, word length, stop bits and parity from the device descriptor or from the tmode utility after the device is initialised.

i. BAUD byte

a. bits 0-3 = baud rate

b. bits 4 = reserved

c. bits 5-6 = word length 0=8, 1=7 bits.

d. bit 7 = stop bits 0=1, 1=2

ii. TYPE byte

a. bits 0-3 = reserved

b. bits 4-7 = parity

b. Bug fix from 01.01.00 - now no longer need the ACIA cartridge installed for the bit banger port to operate. (There's hope for CoCoLink yet! G.)

c. New utility to help timing problems, TUNEPORT. This utility allows you to fine tune your serial port for the baud rate you wish to use so it will work best with your printer or terminal.

3. External Serial Driver - ACIAPAK/T2

a. Support for changing baud, word length, stop bits and parity after device has been initialised similar to above drivers.

b. Support for auto answer modems. Driver now uses the DCD line to tell if a hangup occurs and all processes started will be killed off by the system so that the next caller will not come in where the previous caller left off. (Again sounds like CoCoLink might benefit - yes? G.)

4. New Drivers added for additional hardware.

a. SSCPAK/SSC - driver for the Speech/sound Cartridge.

b. MODPAK/M1 & M2 - driver for a 300 baud ACIA cartridge addressed at $FF6C and to be used in slot 2 of the Multipak. A standard ACIAPAK (addressed at $FF68) can be modified to work with this driver.

c. C080/TERM80 - CC10 comodule for an 80 column display card.

d. CCHDISK/H0 & H1 - hard disk driver for the WD1000-TBI controller board and a 15 or 35 meg drive. This is the standard drive sold by Tandy for their other systems but a separate interface card is required to interface with the CoCo external bus.

C. System Changes - Kernel, Clock, IOMAN and Sysgo.

1. VIRQs - the virtual interupt. Allows for interrupts to be used by devices which do not generate physical interupts or devices that do generate interupts but are located in a MultiPak slot that does not access the CART line.

2. New IOMAN - now IOMAN gives up a device's static storage after the last path to the device is closed, this will ease the memory fragmentation problems normally found in level 1.

3. Hard Disk Support - On boot SysGo will attempt to change to a device called /H0 and to a commands directory at /H0/CMDS. You cannot boot from the hard disk but you can boot to the bard disk from the floppy. If the /H0 device is not found you will boot to /D0 as usual.

continued on Page 62.....

CORRECTIONS

"Cooking With CoCo" (Febuary 1985, Page 43): Colin Stearman advises us that there is a problem with the FLEXIKEY routine in his program. To correct this bug, change the following lines in the source code:

825 - Replace with these two lines:

```
ABX

LDA     ,X
```

882 - Replace with these two lines:

```
ABX

TST     ,X
```

"Getting On The Right Track" (Sept.1985,Page51): Colin Stearman also informed us there is a minor problem in Listing 2 that causes an 80-track drive to access only 142 granules. Insert the following line to correct the problem:

```
885 IF TRACKS=80 AND VP=142 THEN

 VP=156
```

"If Your Horse Comes In First, You Lose" (December 1985, Page 11): Two readers, Wilf Sloan and Scott Kelly, suggested fixes for a problem in this program that causes incorrect scoring. Change lines 1040 and 1280 to read as follows:

`1040... SC(1)=0...` (change this statement only)

`1280... PK=253 ...` (change this statement only)

continued from Page 22.

```
680 PRINT:PRINT"  YOU CAN TRY TO
 BLOCK THE        OTHER SNAKE WI
TH YOURS."
690 PRINT@422,"(ANY KEY FOR MORE
)";:GOSUB 340
700 IF INKEY$="" THEN 700
710 CLS4:PRINT@32,"  THE JOYSTIC
K CONTROLS THE       DIRECTION O
F YOUR SNAKE."
720 PRINT:PRINT"  WHEN YOU SEE T
HE SNAKE HEADS,   PRESS EITHER f
ire button TO     START THE GAME
."
730 PRINT@358,"DO YOU WANT TO PL
AY";:PRINT@389,"FAST OR SLOW (f
OR s)";:GOSUB 340
740 A$=INKEY$:IF A$="F" THEN S=1
 ELSE IF A$="S" THEN S=50 ELSE G
OTO 740
750 GOTO30
760 '******SNAKE CHASE*******
770 '*****ROBERT E. RICE******
780 '**15910 WOODPOST PLACE**
790 '****TAMPA, FL 33624******
800 '********JUNE 1984*********
```

continued from Page 50.

```
9000 '-OPEN HDR/SYS-
9005 '
9010 F$="HDR/SYS:"+RIGHT$(STR$(H
S),1)
9015 OPEN"D",#1,F$,110
9020 FIELD#1,5 AS H1$,1 AS H2$,1
04 AS H3$
9025 K1=LOF(1):RETURN
9100 '-OPEN HDR/BAK-
9105 '
9110 F$="HDR/BAK:"+RIGHT$(STR$(H
B),1)
9115 OPEN"D",#2,F$,110
9120 FIELD#2,5 AS I1$,1 AS I2$,1
04 AS I3$
9125 K2=LOF(2):RETURN
9150 'HEADER/BAK FIELD
9155 GOSUB9100
9160 FIELD#2,5 AS H1$,1 AS H2$,3
 AS H3$,2 AS H4$,80 AS H5$,5 AS
H6$,5 AS H7$,1 AS H8$,8 AS SP$
9165 RETURN
9200 '-MSG/SYS-
9205 '
9210 F$="MSG/SYS:"+RIGHT$(STR$(M
S),1)
9215 OPEN"D",#1,F$,80
9220 FIELD#1,80 AS MG$
9225 K1=LOF(1):RETURN
9300 '-MSG/BAK-
9305 '
9310 F$="MSG/BAK:"+RIGHT$(STR$(M
B),1)
9315 OPEN"D",#2,F$,80
9320 FIELD#2,80 AS MX$
9325 K2=LOF(2):RETURN
```

continued from Page 57.

D.Utility Additions & Changes.

1. Several utilities have been modified to alter their output format depending upon the size of the screen it is being sent to. If you send a 'dir' to the standard screen and to /T2 you will note a large difference in format. The utilities which adjust the screen size are:

dir	mdir	tmode
login	config	dump
procs	tsmon	xmode

2. INIT2 has been added to the utility set to initialise a device. This is useful in startup to initialise commonly used devices like P.

3. OS9GEN has a new -s option for single drive operation, although you must do lots of swaps!

4. TUNEPORT has been added to adjust your serial port.

5. CONFIG - a new menu driven utility on a separate disk to create custom system disks. Lets you choose between all the possible drivers to create a disk that has on it just what you want and need for your own system.

NOW! is the time to subscribe to Australian RAINBOW

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

RATES

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

BACK ISSUES

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

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

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

AUSTRALIAN RAINBOW

Feb '85-Apr '85 ☐	$3.95
Aug '84-Jan '85 ☐	$3.25
to July '84 ☐	$3.00

DISKS & TAPES

CoCoOz (Aust CoCo on Tape)

☐ Latest $8

☐ 6 months $42

☐ 12 months $75

First Issue Mar '83

or Debit my Credit Card Monthly ☐

Blank tapes 12 for $18 or $1.70 ea ☐

Cassette Cases 10 for $5 ☐

Disks — $3.50 ea ☐ 10 for $29.99 ☐

Rainbow on Tape (program listings) $12 for month of ☐

Please note that RAINBOW on TAPE is issued irregularly

or Annually $144 ☐ or Debit my Credit Card Monthly ☐

BOOKS

Byte
Elementary $5.95 ☐

Help
Medium $9.95 ☐

Facts
Advanced $11.95 ☐

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

PLUS

☐ Best of CoCoOz # 1 (Education)
☐ Best of CoCoOz # 2 (Games)
☐ Part 1 for 16K ECB
☐ Part 2 for 32K ECB
(Some 16k CB on P1)

TAPE $ 10.00, DISK $ 21.95

BULLETIN BOARD

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

☐ VISA ☐ BANKCARD
☐ MASTERCARD

Signature _____

☐ CASH
☐ CHEQUE
☐ MONEY ORDER

Authorised

Amount $ _____

BLOCK CAPITALS PLEASE

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

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

FIRST NAME SECOND NAME

Address

PC

STD Code Local Number

Telephone Number

New Subscription ☐ Renewal ☐

GOLDSOFT

Hardware & Software for your TANDY computer.

HARDWARE

The CoCoConnection: Connect your CoCo to the real world and control robots, models, experiments, burglar alarms, water reticulation systems — most electrical things. Features two MC 6821 PIAs; provides four programmable ports; each port provides eight lines, which can be programmed as an input or output; comes complete with tutorial documentation and software; supplied with LED demonstration unit. Switchable memory addressing allows use with disk controller or other modules via a multipack interface; plugs into Cartridge Slot or Multipack, uses gold plate connectors; a MUST for the hardware designer and debugger!		$206.00
Video-Amp: Connects simply to your CoCo to drive a Colour or Mono monitor.	With instructions With instructions and sound	$25.00 $35.00

SOFTWARE

CoCoOz: The programs you see lis.ed in Australian CoCo Magazine are available on CoCoOz! No laborious typing — just (C)LOAD and Go!	Each Tape Subscription, 6 months 12 months **NEW** for 1986 ONLY Each DISK Subscription on disk, 12 months	$8.00 $42.00 $75.00 $10.95 $102.50
Rainbow on Tape: Australian. The programs you see listed in Australian Rainbow Magazine are available on tape. A boon if you don't understand the language! American. We also supply the programs found in American Rainbow on tape. Please specify either Australian or American.	Each Tape Subscription, 12 months **NEW** for 1986 ONLY Each DISK Subscription on disk, 12 months	$12.00 $144.00 $15.00 $172.00
MiCoOz: The programs in the MiCo section of Australian CoCo Magazine. (For MC 10 computers only) Back issues of CoCoOz and MiCoOz are always available on tape only.	Each Tape Subscription, 12 months	$8.00 $75.00
To be released soon: **GOLDDISK 1000** — programs from 'softgold' for your Tandy 1000 on disk, and, **GOLDDISK 4** — programs from 'softgold' for your Tandy Model 4!		$10.95
The Best of CoCoOz: #1 . . . EDUCATION programs. Fourteen programs for the teacher or parent. #2 . . . Part 1 . . . 16K GAMES. (Mainly ECB). Sixteen programs to keep you on your toes! Part 2 . . . 32K GAMES. Adventures, simulations and arcade games for everyone! #3 . . . UTILITIES. Programs to make your use of the computer easier. Spoolers, Reverse Video, Disk utilities and more. #4 . . . BUSINESS programs. Invoicing, Accounts, Creditors and more.	Per Tape Per Disk #3 on Disk Any two tapes	$10.00 $21.95 $16.00 $17.95
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 Minerva (OTC) and Viatel (Telecom).	Subscription to CoCoLink, 12 months	$29.00
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) See Subscription Page for further details.		
Books: HELP: A quick reference guide for CoCo users. FACTS: THE reference for CoCo machine language programmers. MiCo HELP: A quick reference for owners of MC 10 computers.		$9.95 $11.95 $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.

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

user group CONTACTS

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

ADELAIDE	JOHN HAINES	08 278 3568
ADELAIDE NTH STUN	EISENBERG	08 258 6214
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	DANIEL CLARKSON	067 72 8831
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
BLACKWATER	ANNIE MEIJER	079.82.6931
BLAXLAND	BRUCE SULLIVAN	047 39 3983
BOWEN	TERRY COTTON C/O	077 86 2220
BRASSALL	BOB UNSWORTH	07 201 8659
BRIGHTON	GLENN DAVIES	08 296 7477
BRISBANE:		
EAST	ROB THOMPSON	07 848 5512
SOUTH	PATRICK SIMONIS	07 209 3177
SOUTH WEST	GRAHAM BUTCHER	07 376 3480
WEST	BRIAN DOUGAN	07 300 2072
BROKEN HILL	DEAN PARADICE	080 6701
BUNDABERG	RON SIMPKIN	C/O TANDY
CAIRNS	GLEN HODGES	070 54 6583
CAMDEN	KEVIN WINTERS	046.66.8868
CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 88 9226
CHURCHILL	GEOFF SPOWART	051 22 1389
COFFS HARBOUR	BOB KENNY	066 51 2205
COOMA	ROSS PRATT	0648 23 865
COORANBONG	GEORGE SAVAGE	049 77 1054
DALBY .	ANDREW B. SIMPSON	074.62.3228
DARWIN	BRENTON PRIOR	089.81.7766
DENILIQUIN	WAYNE PATTERSON	058 81 3814
DUBBO	GRAEME CLARKE	068 89 2895
EMERALD	LEIGH EAMES	059 68 3392
FORBES	JOHANNA VAGG	068 52 2943
FORSTER	GARY BAILEY	065 54 5829
GIPPSLAND STH	PAT KERMODE	056 74 4583
GOLD COAST	GRAHAM MORPHETT	075 51 8815
GOSFORD	PETER SEIFERT	043 32 7874
GOULBURN VALLEY	TONY HILLIS	058 59 2251
GRAFTON	PETER LINDSAY	066 42 2583
GREENACRES	BETTY LITTLE	08 261 4883
GUYRA	MICHAEL J. HARTMANN	067 79 7547
HASTINGS	MICHEAL MONCK	059.86.8288
HERVEY BAY	LESLEY HORWOOD	071 22 4989
HOBART	BOB DELBOURGO	002 25 3896
IPSWICH	MILTON ROWE	07 281 4859
JUNEE	PAUL MALONEY	069 24 1868
KALGOORLIE	TERRY BURNETT	090.21.5212
KINGSTON	WIM DE PUIT	002 29 4958
LEETON	CHRIS NAGEL	069 53 2969
LITHGOW	DAVID BERGER	063 52 2282
MACKAY	LEN MALONEY	079 512588
MAFFRA	MAX HUCKERBY	051 45 4315
MAITLAND	LYN DAWSON	049 49 8144
MARYBOROUGH	NORM WINN	071 21 6638
MELBOURNE:		
' DANDENONG	DAVID HORROCKS	03 793 5157
' DONCASTER	JUSTIN LIPTON	03 857 5149
' FRANKSTON	BOB HAYTER	03.783.9748
' NARRE WARREN	LEIGH EAMES	03 704 6688
' NTH EASTERN	KEVIN KAZAZES	03 437 1472
' MELTON	MARIO GERADA	03 743 1323
' RINGWOOD	IVA DAVIES	03 758 4496
' SUNBURY	JACK SMIT	03.744.1355
MILDURA	SCOTT HOWISON	050 23 6816
MORPHETTVALE	KEN RICHARDS	08 384 4583
MOREE	ALF BATE	067 52 2465
MORWELL	GEORGE FRANCIS	051 34 5175
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6288
MUDGEE	BRIAN STONE	063-72-1958
MURGON	PETER ANGEL	071 68 1628
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 2895
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7831
ORANGE	STEVE LOVETT	063.62.4825
' or	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PENRITH	ALEX SCHOFIELD	047 31 5303
PERTH	IAN MACLEOD	09 448 2136
PORT LINCOLN	BILL BOARDMAN	086 82 2385
PORT MacQUARIE	RON LALOR	065 83 8223
PORT NOARLUNGA	ROB DALZELL	08 386 1647
PORT PIRIE	KEVIN GOWAN	086 32 1368
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
SALE	BRYAN McHUGH	051 44 4792
SANDGATE	MARK MIGHELL	07 269 5898
SCARBOROUGH	PETER MAY	07 203 6723
SEACOMBE HTS	GLENN DAVIS	08 296 7477
SHEPPARTON	ROSS FARRAR	058 25 1887
SMYTHESDALE	TONY PATTERSON	053 42 8815
SPRINGWOOD	DAVID SEAMONS	047 51 2187
STURT	MARY DAVIS	08 296 7477
SWAN HILL	BARRIE GERRAND	050.32.2838
SYDNEY:		
' BANKSTOWN	CARL STERN	02 646 3619
' BNKSTWN WEST	ART PITTARD	02 72 2881
' BLACKTOWN	KEITH GALLAGHER	02-627-4627
' CAMPBELLTOWN	LEO GINLEY	02 605 4572
' CHATSWOOD	BILL O'DONNELL	02 411 3336
' or	MARK ROTHWELL	02 817 4627
' CLOYTON HERMAN	FREDRICKSON	02 6236379
' DACEYVILLE	ROBERT MILK	SEEAUSTCOCO
' HILLS DIST	DENIS CONROY	02 671 4865
' HORNSBY	ATHALIE SMART	02 848 8838
' LIVERPOOL	LEONIE DUGGAN	02-607-3791
' MacQUARIEFLDS	KEITH ROACH	02 618 2858
' ROSEVILLE	KEN UZZELL	02 467 1619
' SUTHERLAND	IAN ANNABEL	02 528 3391
' TEENS	ROD HOSKINSON	02 48 5948
'SYDNEY EAST	JACKY COCKINOS	02 344 9111
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
TARA	STEVEN YOUNGBERRY	
TONGALLA	TONY HILLIS	058 59 2251
TOOWOOMBA	GRAHAM BURGESS	076 38 4254
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2864
TRARALGON	MORRIS GRADY	051 66 1331
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
WAGGA WAGGA	CES JENKINSON	069 25 2263
WONTHAGGI	PAT KERMODE	056 74 4583
WYNYARD	ANDREW WYLLIE	004 35 1839
YARRAWONGA	KEN SPONG	057 44 1488

SPECIAL INTEREST GROUPS

' BUSINESS

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

' TDP - TELEWRITER, DYNACALC PROCOLOR

BRISBANE	GEOFF TOLPUTT	07 44 6884

' OS9 GROUPS

BANKSTOWN	CARL STERN	02 646 3619
BRISBANE	JACK FRICKER	07 262 8869
COOMA	FRED BISSELING	0648 23263
KALGOORLIE	TERRY BURNETT	090.21.5212
PENRITH	BOB THOMSON	047 30 2468
SYDNEY EAST	JACKY COCKINOS	02.344.9111
SYDNEY NTH	MARK ROTHWELL	02 817 4627

' MiCo GROUPS

LITHGOW	DAVID BERGER	063 52 2282
PORT LINCOLN	BILL BOARDMAN	086 82 2385
ROCKHAMPTON	TIM SHANK	079 28 1846
SYDNEY	RAJA VIJAY	02 519 4186
WARRNAMBOOL	GARY FURR	055 62 7448

' TANDY 1000 / MS DOS

BRISBANE	BRIAN DOUGAN	07 300 2072
MELBOURNE	TONY LLOYD	03 580 8878
SYDNEY	ROGER RUTHEN	047.39.3983

' FORTH

BRISBANE	JOHN POXON	07 208 7828
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
SYDNEY	JOHN REDMOND	02 85 3751

' ROBOTICS

BOWEN	TONY EVANS	077 86 2220
GOLD COAST	GRAHAM MORPHETT	075 51 8815
TAMWORTH	ROBERT WEBB	067 65 7256
WAGGA WAGGA	CES JENKINSON	069 25 2263

AUSTRALIAN COCO / softgold THIS MONTH

SPECIAL EDUCATION ISSUE

* CLUB ROOM
* CASINO
* MC-10
* BATCH FILES

POSTAGE PAID AUSTRALIA