Serving the CoCo Community for

The

RAINBOW®

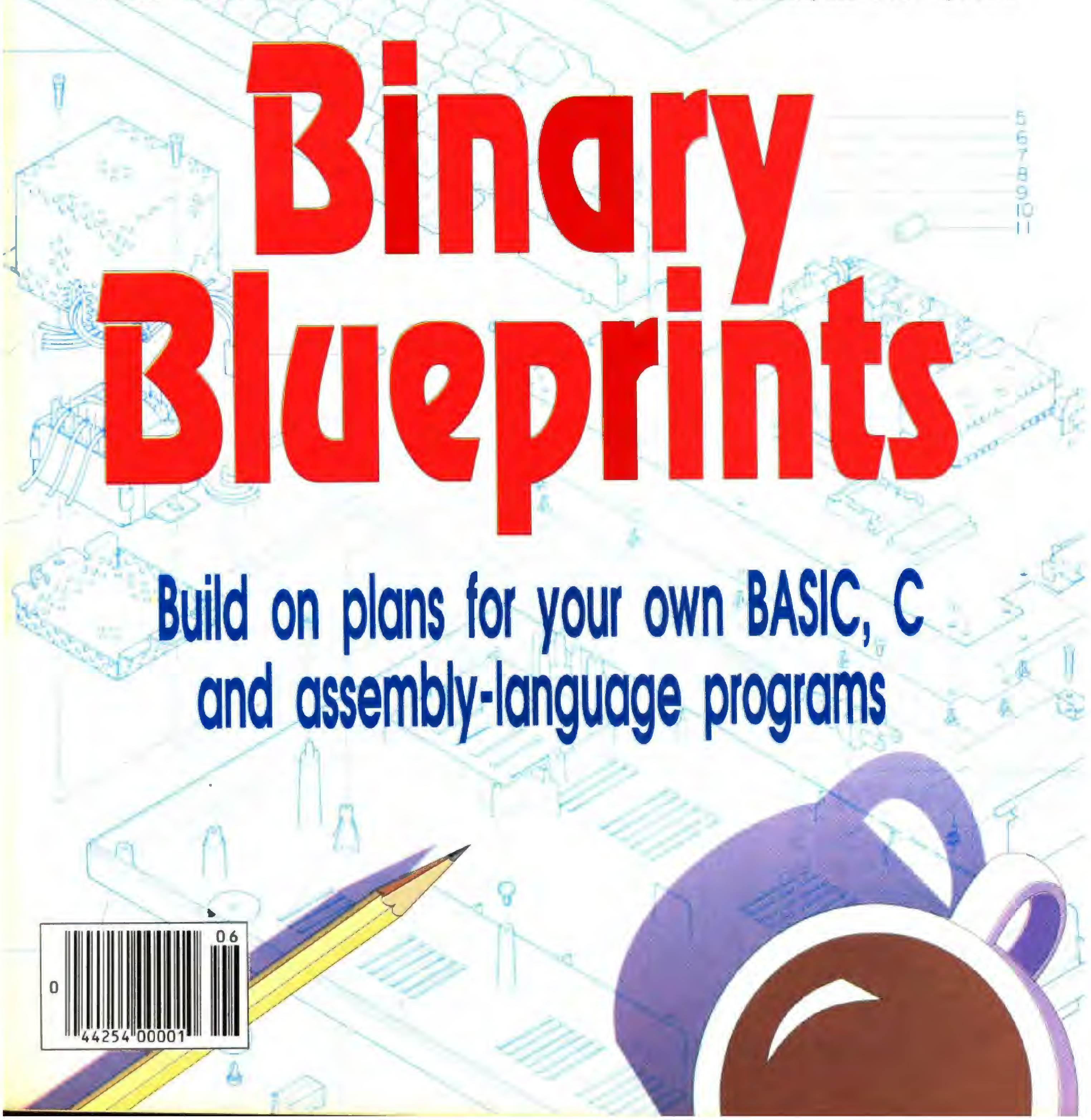
10 YEARS

THE COLOR COMPUTER MONTHLY MAGAZINE

June 1991 Vol. X No.10 Canada $4.95 U.S. $3.95

Binary Blueprints

Build on plans for your own BASIC, C and assembly-language programs



From Computer Plus to YOU . . .

PLUS *after* **PLUS** *after* **PLUS**

Tandy 1100 FD $599*
Tandy 102 32K $310*
Tandy WP-2 $279

Tandy 2500 XL/2 $1099
Tandy 4020 SX $1499

Tandy 1000 RL $499
Tandy 1000 TL/3 $699*

DMP-134 $219*

Color Computer Disk Drive
Drive 0 $239 Drive 1 $149

TandyFax 1010 $619

BIG SAVINGS ON A FULL COMPLEMENT OF RADIO SHACK COMPUTER PRODUCTS

COMPUTERS	
Tandy 1000 RLX HD 1 Drive 1Meg RAM	999.00
Tandy 1500 HD 1 Drive 640K	1499.00
Tandy 2810 HD 1 Drive 1 Meg RAM	2049.00
Tandy 1000 RL HD-1 Drive-512K	829.00
Tandy 1100 FD 1 Drive 640K	599.00 *
PRINTERS	
Tandy DMP-107 120 CPS	219.00
Tandy DMP-302 270 CPS	469.00
Tandy DMP-202 180 CPS	299.00
Tandy DMP-442 300 CPS	359.00 *
Tandy LP-950 Laser Printer	1299.00
Tandy DMP-240 192 CPS 8 color	415.00
Panasonic KXP 1180 192 CPS	189.00 *
Panasonic KXP 1191 240 CPS	259.00 *
Panasonic KXP 1124 192 CPS	299.00 *
Okidata 320 300 CPS	369.00
Okidata 390 270 CPS 24 Wire HD	515.00
OKI Laser 400 4PPM	699.00 *
MODEMS	
Tandy DCM-6	52.00
Tandy DCM-7	85.00
Cardinal 1200 Baud External	99.00
Cardinal 2400 Baud	129.00

COLOR COMPUTER MISC.	
Tandy Drive Controller	89.00 *
Extended Basic Rom Kit (28 pin)	14.95
64K Ram Upgrade Kit (2 or 8 chip)	39.00
Tandy Deluxe Keyboard Kit	24.95
HI-RES Joystick Interface	8.95
Color Computer Deluxe Mouse	44.00
Multi Pak Pal Chip for COCO 3	14.95
COCO 3 Service Manual	29.95
Serial to Parallel Converter	59.95
Tandy Deluxe Joystick	19.95
Magnavox 8135 RGB Monitor	299.00
Magnavox Green or Amber Monitor	99.00
Tandy CM-8 RGB Monitor	249.00
Tandy Pistol Grip Joystick	17.95 *
PBJ OK COCO 3 Upgrade Board	29.95
PBJ 512K COCO 3 Upgrade	89.00
Tandy OK COCO 3 Upgrade Board	39.95
Tandy 512K COCO 3 Upgrade	99.00 *

COLOR COMPUTER SOFTWARE

	TAPE	DISK
The Wild West (COCO 3)		25.95
Worlds of Flight	34.95	34.95
Mustang P-51 Flight Simul.	34.95	34.95
Flight 16 Flight Simul.	34.95	34.95
COCO Util II by Mark Data		39.95
COCO Max III by Colorware		79.95
Max 10 by Colorware		79.95
AutoTerm by PXE Computing	29.95	39.95
TW-80 by Spectrum (COCO 3)		39.95
TeleWriter 64	49.95	59.95
TeleWriter 128		79.95
Elite Word 80		79.95
Elite Calc 3.0		69.95
CoCo 3 512K Super Ram Disk		19.95
Home Publishing by Tandy (CoCo 3)		35.95
Sub Battle Sim. by Epyx (CoCo 3)		26 95
Thexder by Sierra (CoCo 3)		22.45
Kings Quest III by Sierra (CoCo 3)		31.45
Flight Sim. II by SubLogic (CoCo 3)		31.45
OS-9 Level II by Tandy		71.95
OS-9 Development System		89.95
Multi-View by Tandy		44.95
VIP Writer (disk only)		69.95
VIP Integrated Library (disk)		149.95

Prices are subject to change without notice. Please call for shipping charges. Prices in our retail store may be higher. Send for complete catalog.

***Sale prices through 6-10-91**

CALL TOLL FREE 1-800-343-8124

- *LOWEST POSSIBLE PRICES*
- *BEST POSSIBLE WARRANTY*
- *KNOWLEDGEABLE SALES STAFF*
- *TIMELY DELIVERY*
- *SHOPPING CONVENIENCE*

VISA AMERICAN EXPRESS MasterCard

computer plus

P.O. Box 1094
480 King Street
Littleton, MA 01460

SINCE 1973

IN MASSACHUSETTS CALL (508) 486-3193

Serving the CoCo Community for 10 years

RAINBOW

Table of Contents

June 1991
Vol. X No. 11

The cassette tape/disk symbols beside features and columns indicate that the program listings with those articles are on this month's RAINBOW ON TAPE and RAINBOW ON DISK. Those with only the disk symbol are not available on RAINBOW ON TAPE. For details, check the RAINBOW ON TAPE and RAINBOW ON DISK ad.

THE RAINBOW is published every month of the year by FALSOFT, Inc., The Falsoft Building, 9509 U.S. Highway 42, P.O. Box 385, Prospect, KY 40059, phone (502) 228-4492. **THE RAINBOW**, **RAINBOWfest** and **THE RAINBOW** and **RAINBOWfest** logotypes are registered trademarks of FALSOFT, Inc. ● Second class postage paid Prospect, KY and additional offices. USPS N. 705-050 (ISSN No. 0746-4797). POSTMASTER: Send address changes to **THE RAINBOW**, P.O. Box 385, Prospect, KY 40059. Authorized as second class postage paid from Hamilton, Ontario by Canada Post, Ottawa, Ontario, Canada. ● Entire contents copyright by FALSOFT, Inc., 1991. **THE RAINBOW** is intended for the private use and pleasure of its subscribers and purchasers and reproduction by any means is prohibited. Use of information herein is for the single end use of purchasers and any other use is expressly prohibited. All programs herein are distributed in an "as is" basis, without warranty of any kind whatsoever. ● Tandy, Color BASIC, Extended Color BASIC and Program Pak are registered trademarks of the Tandy Corp. ● Subscriptions to **THE RAINBOW** are $31 per year in the United States. Canadian rates are U.S. $38. Surface mail to other countries is U.S. $68, air mail U.S. $103. All subscriptions begin with next available issue. ● Limited back issues are available. Please see notice for issues that are in print and their costs. Payment accepted by VISA, MasterCard, American Express, cash, check or money order in U.S. currency only. Full refund after mailing of one issue. A refund of 10/12ths the subscription amount after two issues are mailed. No refund after mailing of three or more magazines.

THE RAINBOW

Editor and Publisher
Lawrence C. Falk

Managing Editor Cray Augsburg
Associate Editor Sue Fomby
Copy Editor Rob Moore
Submissions/Reviews Editor Tony Olive
Technical Editor Greg Law
Technical Assistants Ed Ellers, Gregory Shultz
Editorial Assistant Julie Hutchinson
Contributing Editors
William Barden, Jr.
Tony DiStefano, Martin Goodman, M.D.
Eddie Kuns
Art Director Heidi Nelson
Designers Sharon Adams, Teri Kays, Debbee Diamond
Consulting Editors Judi Hutchinson, Laurie D. Falk
Typesetter Debbee Diamond

Falsoft, Inc.

President Lawrence C. Falk
General Manager Peggy Lowry Daniels
Asst. General Mgr. for Finance Donna Shuck
Admin. Asst. to the Publisher Kim Thompson
Editorial Director John Crawley
Director of Creative Services O'Neil Arnold
Chief Bookkeeper Diane Moore
Dealer Accounts Judy Quashnock
Asst. Gen. Manager For Administration Sandy Apple
Corporate Business Technical Director Calvin Shields
Customer Service Manager Beverly Bearden
Customer Service Representative Patricia Eaton
Chief of Printing Services Melba Smith
Chief of Prepress and Dispatch Tim Whelan
Business Assistant Wendy Falk Barsky
Chief of Building Security and Maintenance Lawrence Johnson
Asst. General Manager for Advertising, Development and Production Ira Barsky
Advertising Representatives Belinda Kirby, Kim Lewis
Advertising Assistant Carolyn Fenwick
(502)228-4492

For RAINBOW Advertising and Marketing Office Information, see Page 81

Letters to the RAINBOW

Thanks!

Editor:

I wrote you concerning my search for a disk-based version of *Cyrus* or a way to transfer it from the ROM pack to disk. I never expected my letter to get published, but to my surprise, I was wrong. It was published in the March 1991 issue and I received letters with offers of help from everywhere.

Thank you for helping me by publishing my letter. There are a lot of really nice people in the CoCo Community who need to know how important and helpful their responses are. I'm relatively new to the CoCo and this helped me decide to subscribe to THE RAINBOW. To those of you who have not yet subscribed, go for it!

James Gunter
Ruston, Louisiana

Pardon Our Omission

Last month (May 1991 issue) Rick Benton posed a question regarding file transfers between different operating systems. In answering Rick Benton's letter, we covered Disk BASIC/MS-DOS transfers. However, we neglected to mention Granite Computer Systems File Transfer Utilities, *which runs under OS-9 and allows file transfers between OS-9, MS-DOS and Flex.*

Drive Alignment

Editor:

I enjoy the articles in THE RAINBOW even when I don't understand them. I just keep trying. I am retired and want to study and learn more from my back issues of THE RAINBOW. I am interested in the OS-9 material and notice it goes back many issues.

Now that I have two Tandy 1000 computers, I like the CoCo even more. With the CoCo I can tell the machine what I want it to do. With the Tandy 1000, I have to give precise instructions to get it to do what I want.

I also have a question for your experts. My 26-1161A and 26-3029 disk drives do not work together any more. When I format a disk on either one of the drives, I must use the disk on the same drive on which it was formatted. This tells me the tracking is not working the same on both of them. I am looking for the best way to get them working together again.

Robert Welsh
3250 N. Cascade
Colorado Springs, CO 80907

If one drive cannot read a disk formatted on the other drive, one or both drives may need to be realigned. To determine which needs to be aligned, format a disk on both drives and then format a disk on a drive that is known to be good. If either of your drives is good, it should be able to read the disk formatted on the known drive and vice versa. When you determine which drive(s) need to be aligned, get an estimate from the computer stores in your area. While aligning a drive is relatively simple, you may find the cost of getting the drive aligned very close to that of buying a new drive, especially considering many mail-order vendors sell 40-track, double-sided drives for $50 to $60.

Cub Scout Software

Editor:

I am a Cub Scout leader and wonder if there is any available software, with Cub Scout or Boy Scout items, that I can use to print banners or award certificates. I use a DMP-105 printer with a CoCo 3. Any help you can offer is appreciated.

Jeffrey Mongold
190 Old Corry Fld. Road, Apt. 2004
Pensacola, FL 32507

We know of no programs with scouting-specific symbols. However, there are several fine graphics programs available on the market. Many of these programs support clip art, which would give you the ability to add your own symbols. See the advertisements for Zebra Systems, Colorware, T&D Software and Microcom Software.

Planting Easter Eggs

Editor:

Since there is no medium for hints, tips and Easter eggs for programs since THE RAINBOW Scoreboard vanished, I want to let our customers know about an undocumented feature of the OS-9 version of *Kyum-Gai: to be Ninja*. If you type `ninja` (the normal start-up file) with a number parameter indicating the starting level, you can skip to any level in the game, up to Level 8. This feature was included by Kevin Darling as a debugging tool, and when he asked if we should remove it, I thought it would be kind of fun to leave it in. Therefore, this feature is for those of you who want to see the next level, but just can't get there, and those who want to start the battle with the Giant at the end with a full regiment of lives. Note that this feature can only be used with the OS-9 version. Thanks to the CoCo Community whose support made this project possible.

Glen R. Dahlgren
President
Sundog Systems

Looking for an Assembler

Editor:

Where can I get a good editor/assembler? I like to write assembly-language programs, and my T&D assembler just doesn't cut it any more. I have a CoCo 3 with a tape and a disk drive.

Ryan Boughter
RD #2, Box 554
Barto, PA 19504-9447

You can purchase Disk EDTASM+ *or the* EDTASM+ *ROM pack from Tandy's Express Order (EOS) service (call 800-321-3133). When we checked, EOS indicated they had limited quantities of both in stock.*

Slick Printer Ribbons

Editor:

If you want to get more use out of your printer ribbon, but cannot find a re-inking kit, gently pry open the cartridge and give the ribbon two or three squirts of WD-40. Put the ribbon away for a while.

The oil breaks most of the ink loose from the unusable portions of the ribbon and allows it to spread to the usable portions.

It is not like having a new or re-inked ribbon, but it is good for routine [draft] printing.

MMC Daniel Statham
FPO Seattle

Day Dreaming

Editor:

I am a longtime subscriber to your magazine. I think you do a great job of bringing us a quality product. I have a few ideas that you might be able to use.

First, I would like to see a hardware project for a remote keyboard. I know that past issues have shown how to extend the keyboard with ribbon cable, but I would like to see an issue dealing with extending the CoCo's keyboard using a five-wire cable communicating with the computer via asynchronous or synchronous mode. Howard Medical has such a package, but it looks like an IBM-style keyboard is required. A blank panel could be installed where the CoCo keyboard once resided, and all kinds of goodies could be installed on it — High Resolution Joystick Interface with a switch for low resolution, on/off switch, reset switch, power on/off light.

It would be great if this could be accomplished with the existing polling used by Microsoft BASIC and OS-9 software. If not, can you explain why?

Next, I would like to see a hardware article on interfacing a hand scanner to the CoCo. I'm dreaming, but it would be nice.

I would also like to see a software article on object-oriented programming (OOPS) and an article to add object-oriented structures to existing languages such as C, BASIC09 and OS-9 Pascal.

Cecil Ellis
709 Woodlawn Street
Fort Walton Beach, FL 32548

You can purchase a six-foot keyboard cable from Microcom to extend your keyboard, or an IBM-style keyboard and adapter from Frank Hogg Laboratories. The CoCo is designed to use a "dumb" keyboard, which is nothing more than a series of switches wired in a certain matrix and connected to the computer in "parallel." IBM-style keyboards on the other hand use a sophisticated keyboard-controller chip that converts the matrix into a set of scan codes that is transmitted serially to the computer. A standard CoCo keyboard will not work with the currently available IBM-keyboard adapter.

"High Scores" for VIP

Editor:

I imagine one of the most-used programs on any CoCo, second only to games, is the word processor. I would like to suggest a column similar to "Scoreboard" for games, but this could be called "Utility Help." I imagine a lot of people would like help using their word processor or database program. I have used *VIP Writer* for over five years and have managed to print labels, the Canadian Postal Code, lined forms, and numerous small graphics — all on a word processor. Sample questions include: How do you print in two columns with *VIP Writer*? How do you use the 'format commands' in ...? How do you edit a BASIC program with a word processor? These questions, and others, could be answered by the readers to help others.

John Coldwell
Prince Rupert, BC
Canada

Pie in the Sky

Editor:

I'm looking for a pie-chart program that can print a decent graphics facsimile of the chart itself. I tried a program in THE RAINBOW, as well as one from T&D Software. Both produced insufficient graphics images. I'm an avid user of *Max-10*. Please help me if you can.

Pastor James Dale Altom
3213 Harvard
Collinsville, IL 62234

Math Coprocessors

Editor:

I have a 512K CoCo 3 with one double- and one single-sided drive, but it is presently packed away to make room for our new baby. I recently adopted a program from one of my astronomy magazines (*Sky and Telescope*) to make a star-map database and display program. In the same magazine were advertisements for IBM, Apple and Commodore programs. Most programs require a math coprocessor, 512K and VGA. What is a math coprocessor and can one be built for the CoCo 3 (either onboard or plugged into the ROM cartridge slot)?

I know I could probably just use the RS-232 port and another CoCo, but I was thinking of something a little like a coprocessor. This would free the CoCo for other tasks while something else did high-speed number crunching and then pass the answers back to the CoCo for graphics display or animation. This would greatly enhance the graphics qualities of the CoCo 3.

Mike Miador
1009 Good Bar
Nashville, TN 37217

A math coprocessor is a special microprocessor designed to perform floating-point calculations. It works in tandem with the primary microprocessor, such as a Motorola 68000 or an Intel 8086, and often

CORRECTIONS

"The Assembly Line" (May 1991, Page 56): Due to a production error, the last line is missing from Listing 1 on Page 57. Add the following line:

```
130 DATA 2,3,80,10,9,0,14,3
```

The program as it appears on the May 1991 issue of RAINBOW ON TAPE/DISK is correct.

requires bus-arbitration signals, such as Bus Request and Bus Acknowledge, to prevent the two processors from using the bus simultaneously.

Serial/Parallel

Editor:

I hope you can help me. I need a cable to connect my CoCo 3 computer to a Toshiba (IBM-compatible) printer. On the CoCo 3 end I need a 4-pin DIN plug. On the other end I need a 36-pin parallel printer connector. Is it possible to order such a cable?

Bette Foss
1736 Serena Drive, E.
Jacksonville, FL 32225

You need a serial/parallel converter to connect that Toshiba printer to your CoCo. See the ads for Owl-Ware, Dayton Associates, Frank Hogg Laboratories and Microcom Software.

All the Right Questions

Editor:

I just received my very first issue of THE RAINBOW. It makes me happy to see that the CoCo has a lot of support. Keep up the good work.

I own a 512K CoCo 3, a CM-8 monitor, an FD-501 dual-drive system, and a cassette recorder. As far as I know, the CoCo 3 with a CM-8 monitor will not display CoCo 2 games in color. But, by accident, I managed to do it.

If you put the *Pitfall II* disk in Drive 1 and the *Ghana Bwana* disk in Drive 0 and enter the following commands:

```
DRIVE 1
DIR
DOS
```

Drive 1 loads the booting program for *Pitfall II*. Press ENTER and the computer switches to Drive 0 and loads *Ghana Bwana* in color.

I would like to know why? Also, what can I do to get all of my CoCo 2 games (including program paks and tapes and disks) displayed in color?

My Drive 0 is a double-sided drive. Why won't it format the back of my disks? What can I do to use the back of my disks?

I also need a Multi-Pak Interface. From whom can I get one? And what does the PAL chip do to the interface?

I read your article on monitors, and I was a little unhappy when I found out Magnavox makes a better monitor than the CM-8 from Tandy. Does the Magnavox ICM135 display CoCo 2 games in color? Is the 80-column display on the Magnavox sharper than the CM-8? Is .42mm so sharp the display difference is noticed right away? I know I have asked a lot of questions, but I need the answers from somebody. I am very dissatisfied with Radio Shack customer service. They sell the computers, but know nothing about them.

Also, if I hook up the CoCo 3 to a stereo from the audio to auxiliary, can it damage the computer or the stereo?

Raul Torres
7011 Baywood Court
Tampa, FL 33615

Phew! Anything else? Well, let's see . . .

The `DOS` *command loads the data stored on Track 34 of the disk in the last drive accessed — in your example above, you could leave out the* `DRIVE 1` *command and start by entering* `DIR 1`. *If the first two characters of the loaded data are OS,* `DOS` *executes the data as a program. In the case of Pitfall II, the boot program is hard-coded for Drive 0, so it switches back to Drive 0.*

Steve Bjork, author of Ghana Bwana, Super Pitfall II, Desert Rider *and* One on One, *provided a new boot loader for these games in the May 1991 issue (Page 74). This loader allows these games to display in real colors on an RGB monitor. Also see "Artifact Colors on CoCo 3's RGB" (February 1988, Page 114) for more information. A correction for that article appears in the April 1988 issue.*

If you have Disk BASIC 2.0 (check the startup screen), enter `POKE &HD7AC, &H41` *and* `POKE &HD7AD, &H42` *to use double-sided drives. If you have Disk BASIC 2.1, enter* `POKE &HD89F, &H41` *and* `POKE &HD8A0, &H42`. *The original values for these locations are $04 and $40, respectively, regardless of your Disk BASIC version. These pokes assume your drive cable has all teeth intact in the drive-end connectors. Also, the locations are returned to their original values the second you press Reset.*

The Slot Pak III, a multi-slot cartridge device, is available from Frank Hogg Laboratories. The PAL upgrade for Radio Shack Multi-Paks is intended to eliminate problems caused by address ghosting.

The Magnavox ICM135 does support composite video so, yes, it will display artifact colors used by CoCo 2 software. As it uses a .42 dot pitch (as opposed to the CM-8's .52), the image is much sharper and is immediately noticeable.

The Audio output on the back of the CoCo is a "line-level" signal. You should be able to connect it to a "line" input (AUX, TAPE IN, etc.) on the back of stereo receiver without cause for concern.

Craps, MON!

Editor:

The March 1991 issue of THE RAINBOW has a neat little game program called *Craps*. It has a serious flaw for users of *ADOS-3*.

The variable `MON` is a resident macro in the *ADOS-3* system, and its inclusion as a variable makes the program completely unworkable. Changing the variable to `MN` solves the problem and makes for a very nice little program.

My congratulations to Doug Fingliss. Not many 9-year-olds could design a program like this. I can't, and I'm a great-grandfather!

John Norris
Severna Park, Maryland

Thanks for the tip, John.

Note Writer

Editor:

I liked the *Note Writer* program written by John Musumeci and featured in the February 1991 issue of THE RAINBOW. However, I would prefer an 80-column program suitable for letter writing. Perhaps Mr. Musumeci could write such a program for your magazine.

Also, the keyboard input is so slow with this program I found myself dropping letters. This could be improved by speeding up the computer with the `POKE 65497, 0` command.

I am writing this letter on a modified version of Mr. Musumeci's program, but frankly I don't understand parts of it. I hope you will forward my suggestions to Mr. Musumeci and that he will write a neat 80-column word processor program.

Jim Benson
Chicago, Illinois

It's interesting we should receive your letter at the same time we receive an updated version of Note Writer *from Mr. Musumeci, but that is what happened. Look on Page 16 and you'll find a version that still uses the 40-column screen, but it allows for two side-by-side screens so you effectively get 80 columns on the printout.*

THE RAINBOW welcomes letters to the editor. Mail should be addressed to: Letters to Rainbow, The Falsoft Building, P.O. Box 385, Prospect, KY 40059. Letters should include the writer's full name and address. Letters may be edited for purposes of clarity or to conserve space.

Letters to the editor may also be sent to us through our Delphi CoCo SIG. From the CoCo SIG> prompt, type RAI to take you into the Rainbow Magazine Services area of the SIG. At the RAINBOW> prompt, type LET to reach the LETTERS> prompt and then select Letters for Publication. Be sure to include your complete name and address.

DELPHI'S 20/20 ADVANTAGE, VISIBLY SUPERIOR

D E L P H I	$20
G E N I E	$124
C O M P U S E R V E	$120-$256

COST OF 20 EVENING HOURS OF DOWNLOADS AND CHAT

On DELPHI's 20/20 Advantage Plan, you get 20 hours of connect time each month for only $20. Additional time is only 2 cents/minute. No other full-featured online service even comes close!

And those are 20 hours of real online fun and productivity. Download files, chat with friends, send electronic mail, play multi-player games, and make travel reservations.

Join DELPHI now for only $39.95 and your first 20 hours are free. With your CoCo and modem:

- Dial 1-800-365-4636
- At *Username:* type **JOINDELPHI**
- At *Password:* type **COCO20**

DELPHI

THE WORLD'S PREMIER ONLINE INFORMATION SERVICE

800-544-4005 • 617-491-3393

Already a DELPHI member? Type **GO USING ADVANTAGE**

Print#-2

Binary Blueprints

This is one of my favorite issues of THE RAINBOW because it happens to involve a subject that, I think, goes to the heart of what the Color Computer is all about to a great many of us.

I *almost* put "— programming" at the end of the sentence above, but decided not to because I know many of you are "turned off" by programming. You shouldn't be, because while programming can be complex, of course, it can also be so simple it can solve a whole range of small problems in a jiffy.

Those of you who have read this space for a long while know that the main thing I like about my CoCo is that it can be just about anything I want it to be: A sophisticated typewriter one minute, a link to other computer users the next; an accountant now, a moment later an entertainment source. The possibilities are endless.

All of this is, naturally, accomplished by programming our CoCo. Sometimes it gets a bit complicated, but it does not need to be. This month's theme, with the catchy title Binary Blueprints, simply suggests that you use programming to help solve a few problems for yourself.

Despite all the spreadsheets, word processors, drawing programs, communications programs, desktop publishers and games available for the CoCo, one of the things I like the most is its ability to solve those little problems for me simply and easily. Just the other day, when our friends at the Postal Service raised their rates astronomically, I was greeted with the news of how much additional cost would be involved in mailing this copy of THE RAINBOW to you. I decided I wanted to know the percentage difference.

Well, how do you do that?

(Longtime readers of this column will also know I have supreme difficulty adding two numbers together and getting the same answer twice — any knowledge of formulas disappeared from my head the day I said my last, fond, goodbye to Mrs. Maxwell, my high school algebra teacher.)

I fiddled with the problem on my calculator over and over again. Since I did not know what the answer was, I decided to fiddle with numbers I knew the answer for, then apply them to this problem. It took about 20 minutes, but I finally got it.

This is useful, I thought, but I sure don't want to go through this again. I'll write a program and save it to disk.

The formula for a percent change (which I happen to know is called "Delta percent" with Delta meaning change. I recall this from my NASA-reporting days — Delta-*v* is the change in velocity, but don't ask me how to calculate it) is to subtract the original number from the new number and divide that answer by the old number.

Mrs. Maxwell would have been proud of me if I wrote it as:

(New - Original) / Original

We can write a program in BASIC right on the CoCo to do this same formula:

```
150 C=((N - O) / O)
```

Well, it *really* is as simple as that. Of course we need to get the numbers into and out of the program, but that is a simple matter of adding statements like these:

```
110 INPUT "Original Number: "; O
120 INPUT "New Number: "; N
160 PRINT C
```

Well, what about proper decimal points? Let's just change Line 160 to:

```
160 PRINT C * 100
```

I saved this little gem with the name PCT.BAS and now all I have to do is run it and have this thorny little problem solved in seconds. It is a good example of what the CoCo can do to solve problems for you.

You can fancy this up some if you want, but I am trying to keep it simple. My point is that it *is* pretty simple in the long run. You can make use of a lot of these things, too. And make your CoCo even *more* useful.

* * * * *

Since we were, just a moment ago, on the subject of postage increases, I need to mention here that while we do not plan any increase in United States-addressed subscription prices to THE RAINBOW, part of the process we went through to analyze these costs has prompted another change in the way we distribute our magazine.

We will gradually be cutting back in the non-subscription distribution of THE RAINBOW — that is, those copies we sell to stores for resale at retail. The reason for this is simple: We have never made a penny on "newsstand sales" because of the discounts and shipping costs involved. Historically, however, we have sold magazines this way because many of our distributors were near a Radio Shack store. Someone could buy a CoCo and stop in to pick up a copy of THE RAINBOW on their way home. We hoped they would subscribe.

With the CoCo no longer generally available in Radio Shack stores, this approach no longer makes a lot of sense to us. We will continue to sell to newsstands where there has been an historic demand for THE RAINBOW, and in these locations the magazine will still be available. But let me urge you to subscribe rather than buy from a newsstand.

The obvious advantage of subscribing is to ensure that you do not miss an issue. And there is the convenience factor as well. That we make a little money on a subscription may sway you as well — as long as someone is going to make a profit, better us than some distributor or trucking company.

I know this will not affect a great many of you. But for those of you who it does affect, we welcome your subscription!

— Lonnie Falk

The Rainbow Bookshelf

The Rainbow Book of Simulations, 20 award-winning entries from THE RAINBOW's first Simulations contest.

The Second Rainbow Book of Simulations, the 16 Winners from our second Simulations contest.

The Complete Rainbow Guide to OS-9, authors Dale Puckett and Peter Dibble demonstrate OS-9's multitasking and multiuser features.

The Complete Rainbow Guide to OS-9 Level II Vol.I: A Beginners Guide to Windows. Puckett and Dibble uncover the mysteries of the new windowing environment.

The First Rainbow Book of Adventures, contains 14 winning programs from our first Adventure contest.

The Second Rainbow Book of Adventures, featuring 24 of the most challenging Adventure games ever compiled.

The Third Rainbow Book of Adventures, the Excitement continues with 19 new Adventures.

The Fourth Book of Adventures, fourteen fascinating new Adventures from the winners of our fourth Adventure competition.

A Full Turn of the Screw

The Rainbow Introductory Guide to Statistics, Dr. Michael Plog and Dr. Norman Stenzel give a solid introduction to the realm of statistical processes.

Name ______________________________

Address ______________________________

City ______________________________

State ________________ Zip ____________

❑ Payment Enclosed, or ❑ Charge to:
❑ VISA ❑ MasterCard ❑ American Express

Account Number ______________________________

Card Exp. Date ______ Signature ______________________

❑ **The Rainbow Book of Simulations** (first)	~~$ 9.95~~	$ 3.50	______
❑ Rainbow Simulations Tape (first)	~~$ 9.95~~	$ 3.50	______
❑ First Simulations Package	~~$19.90~~	$ 6.00	______
❑ **The Second Rainbow Book of Simulations**	~~$ 9.95~~	$ 4.95	______
❑ Second Rainbow Simulations Tape	~~$ 9.95~~	$ 4.95	______
❑ Second Rainbow Simulations Disk	~~$10.95~~	$ 4.95	______
❑ Second Simulations Package with Tape	~~$19.90~~	$ 8.95	______
❑ Second Simulations Package with Disk	~~$20.90~~	$ 8 95	______
❑ **The Complete Rainbow Guide to OS-9**	~~$19.95~~	$12.95	______
❑ Rainbow Guide to OS-9 Disk Set (2 disks)	~~$31.00~~	$19.95	______
❑ Rainbow Guide to OS-9 Package	~~$47.95~~	$29.95	______
❑ **The Windows & Applications Disk for** The Complete Rainbow Guide to OS-9 Level II, Vol. I		$19.95	______
❑ **The Rainbow Book of Adventures** (first)	~~$ 7.95~~	$ 2.00	______
❑ Rainbow Adventures Tape (first)	~~$ 7.95~~	$ 2.00	______
❑ First Adventure Package	~~$15.90~~	$ 3.50	______
❑ **The Second Rainbow Book of Adventures**	~~$13.95~~	$ 6.95	______
❑ Second Rainbow Adventures Tape	~~$13.95~~	$ 6.95	______
❑ Second Adventure Package	~~$27.90~~	$11.95	______

❑ **The Third Rainbow Book of Adventures**	~~$11.95~~	$ 6.95	______
❑ Third Adventures Tape	~~$ 9.95~~	$ 6.95	______
❑ Third Adventures Disk Set (2 disks)	~~$14.95~~	$ 7.95	______
❑ Third Adventure Package with Tape	~~$21.90~~	$11.95	______
❑ Third Adventure Package with Disk	~~$26.90~~	$12.95	______
❑ **The Fourth Rainbow Book of Adventures**	~~$10.95~~	$ 7.95	______
❑ Fourth Adventures Tape	~~$ 9.95~~	$ 6.95	______
❑ Fourth Adventures Disk	~~$14.95~~	$11.95	______
❑ Fourth Adventure Package with Tape	~~$20.90~~	$13.90	______
❑ Fourth Adventure Package with Disk	~~$25.90~~	$18.90	______
❑ **A Full Turn of the Screw**		$19.95	______
❑ **Introductory Guide to Statistics**	~~$ 6.95~~	$ 2.95	______
❑ Guide to Statistics Tape or Disk (indicate choice)	~~$ 5.95~~	$ 2.95	______
❑ Guide to Statistics Package (indicate choice of tape or disk)	~~$12.90~~	$ 4.95	______
Add $2 per book Shipping and Handling in U.S.			______
Outside U.S., add $4 per book			______
Kentucky residents add 6% sales tax			______
Canadian residents add 7% GST			______
(Allow 6 to 8 weeks for delivery)		Total	______

Mail to: Rainbow Bookshelf, The Falsoft Building, P.O. Box 385, Prospect, KY 40059. To order by phone (*credit card orders only*) call (800) 847-0309, 9 a.m. to 5 p.m. EST. For other inquiries call (502) 228-4492.

Please note: The tapes and disks offered by The Rainbow Bookshelf are not stand-alone products. That is, they are intended to be an adjunct and complement to the books. Even if you buy the tape or disk, you will still need the appropriate book for loading and operating instructions. OS-9 is a registered trademark of the Microware Systems Corporation.

Programming techniques — understanding the misunderstood

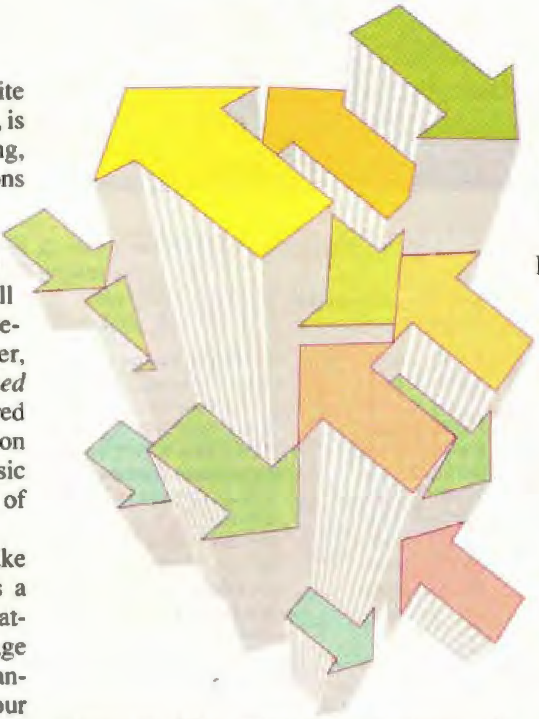
Organized Chaos

by Greg Law

Structured programming, despite what you may have been told, is not a set of all-encompassing, complex rules and regulations enforced by some almighty programming police. I'm sure you've heard the term many times in recent years, but what exactly is this thing we call *structured programming*? Quoting a precise definition is very difficult. However, we know that structured implies *organized* or *orderly*, so we might say that structured programming is an orderly progression toward the creation of a program. In a basic sense, structured programming is more of an ideology than anything else.

Before going any farther, I want to make it clear that this article isn't meant as a comparison of languages. Nor is it an attempt to persuade your use of one language over another. The use of a particular language for any given task is left to your discretion. This article is about structured programming and structured programming techniques. However, modern languages currently available provide modular structure and lend themselves to structured programming. But structured programming is not impossible with non-modular languages such as Color BASIC. You may find some of the objectives of structured programming more difficult to achieve with non-modular languages, but they are not entirely impossible. It is true that Color BASIC programs aren't as readable as those provided with modular languages, but don't let that stand in your way.

In addition to being OS-9 Online SIGop, Greg Law enjoys programming on all types of computers and has worked on systems ranging from the CoCo to the Burroughs B6700 super mainframe. He lives in Louisville, Kentucky.



Programming Objectives

There are several objectives of structured programming that make the extra effort worth every minute. The most important objectives include:

- Improve program readability
- Improve program efficiency
- Improve program reliability
- Minimize program complexity
- Simplify program maintenance
- Increase programmer productivity
- Provide a disciplined programming methodology

You may notice that these objectives aren't trivial, but they are very real and easily obtained. Let's look at each of these objectives a little closer. As we do so, keep in mind that while I have listed certain "rules" under each of the objective headings, most of them apply under other headings as well.

Program Readability

Several steps must be taken to enhance

the reader's comprehension of a program. First and foremost, the correspondence between the source code and the execution process must be trivial. The term *execution process* (the process through which the program executes) implies *the general flow of the program*. In other words, with a well-written program, you can follow the flow of the program by examining the source code as it executes. As the program runs, you are able to determine the section of code the program is currently executing. Perhaps the best way to illustrate how readability is affected by the language and techniques used is to show some examples.

If you were to put a group of people in a room and asked them to explain structured programming, you would undoubtedly hear many proclaim "No GOTOs!" In all honesty, there are some instances where the use of GOTO may make the program easier to read and less complex. However, the need to use GOTO in a modular language is extremely rare. Modular languages, such as Pascal, C and structured dialects of BASIC (such as BASIC09), provide many constructs that make the use of GOTO nearly obsolete. But GOTO may be required occasionally in a BASIC program since most interpreted versions of BASIC do not provide enough constructs, such as DO, WHILE, UNTIL and CASE, to avoid the use of GOTO.

While the standard version of BASIC is very popular and easy to learn, it is a non-modular language with an extremely limited set of control constructs. A control construct is a set of statements that perform a conditional test (e.g. IF/THEN, WHILE/DO and CASE). A frustrating aspect of unstructured dialects of BASIC is that they require multiple IF statements be placed on a single line. This also places a limitation on the number of tasks that can be performed within an IF statement and makes the listing very difficult to read. In BASIC a typical IF statement might be written as follows:

```
100 IF A=B THEN IF C=D THEN GOSU
B 2000:GOSUB 3000 ELSE GOSUB 500
0:GOSUB 6000 ELSE GOSUB 1000
```

In BASIC09 the same compound statement might be written as follows:

```
IF a=b THEN
  IF c=d THEN
    RUN SubTotals
    RUN Totals
  ELSE
    RUN GetChecks
    RUN GetDeposits
  ENDIF
ELSE
  RUN PrintStatement
ENDIF
```

The same compound statement can be written in C as follows:

```
if(a == b) {
  if(c == d) {
    SubTotals();
    Totals();
  } else {
    GetChecks();
    GetDeposits();
  }
} else
  PrintStatement();
```

The modular forms are much easier to read than the non-modular forms. This limitation also leads to the use of GOTO where it would otherwise be unnecessary. One of the most common uses is within an IF statement:

```
100 I$=INKEY$:IF I$="" THEN 100
```

The use of GOTO in this example is required because no other control constructs are provided to perform the test in a more logical manner. In the realm of modular languages, a WHILE loop could be used for the same purpose:

```
WHILE I$=""
  I$=INKEY$
ENDWHILE
```

While this solution is much better, it may or may not work in this specific example. The problem with the WHILE loop in this context is that the comparison is performed before the operation. If I$ contains a string before the comparison, the operation is never performed. This requires adding the statement I$="" immediately before the loop is encountered. A better solution is a DO/WHILE loop, which performs the comparison at the bottom of the loop:

```
DO
  I$=INKEY$
WHILE I$=""
```

In this case, the operation is performed before the comparison and yields the proper result. Consider an actual routine taken from a listing published in THE RAINBOW:

```
760 LN=0:OPEN "I",#1,NA$:IF YN$<
>"" THEN 790
770 GOSUB 250:LOCATE 0,23:ATTR 1
,2:INPUT " Print blank lines (y/
N)";YN$:ATTR 1,0:CLS:IF YN$<>"y"
 AND YN$<>"n" THEN YN$="n"
780 GOSUB 250:LOCATE 0,23:ATTR 1
,2:INPUT " Filter ANSI commands
(y/N)";AN$:ATTR 1,0:CLS:IF AN$<>
"y" AND AN$<>"n" THEN AN$="n"
790 H=0
```

Trying to determine the true meaning of the above routine can be difficult at best, even though it is relatively short. If you are very familiar with BASIC, you may be able to understand the routine over a period of time. The routine is more clear when written one statement-per-line:

```
760 LN=0
761 OPEN "I",#1,NA$
762 IF YN$<>"" THEN 790
770 GOSUB 250
771 LOCATE 0,23
772 ATTR 1,2
773 INPUT " Print blank lines (y
/N)";YN$
774 ATTR 1,0
775 CLS
776 IF YN$<>"y" AND YN$<>"n" THE
N YN$="n"
780 GOSUB 250
781 LOCATE 0,23
782 ATTR 1,2
783 INPUT " Filter ANSI commands
(y/N)";AN$
784 ATTR 1,0
785 CLS
786 IF AN$<>"y" AND AN$<>"n" THE
N AN$="n"
790 H=0
```

Even though the routine is more clear, the true meaning is still obscured because the IF statement is limited to one line. Because of this limitation, the IF statement in Line 762 had to be reversed with a branch to Line 790 if the condition is false. If BASIC supported an ENDIF statement, an IF statement could span multiple lines, resulting in a listing that's easier to read. Of course, you could also add your own indentation to make the flow of the listing easier to follow:

```
760 LN=0
761 OPEN "I",#1,NA$
762 IF YN$="" THEN
770   GOSUB 250
771   LOCATE 0,23
772   ATTR 1,2
773   INPUT " Print blank lines
(y/N)";YN$
774   ATTR 1,0
775   CLS
776   IF YN$<>"y" AND YN$<>"n" T
HEN
777     YN$="n"
778   ENDIF
780   GOSUB 250
781   LOCATE 0,23
782   ATTR 1,2
783   INPUT " Filter ANSI comman
ds (y/N)";AN$
784   ATTR 1,0
```

```
785   CLS
786   IF AN$<>"y" AND AN$<>"n" T
HEN
787     AN$="n"
788   ENDIF
789 ENDIF
790 H=0
```

The listing is easier to read, but it's still not obviously clear. Notice how the line numbers hinder the overall clarity. And what exactly are `LN`, `YN$`, `AN$`, and `H`? Given only this section of the program, it appears that `YN$` is used for a Yes/No response and `AN$` is used for an Answer response. It could also be that `YN$` and `AN$` contain the string "y" if blank lines are printed and ANSI commands are filtered. A definitive answer cannot be determined from this section of the listing. Unfortunately, there is not enough information in this section of the listing to even guess what `LN` and `H` are used for. The modular version of the listing with descriptive variable names and without line numbers is the easiest to read (see Figure 1).

```
LinesInFile:=0
OPEN "I",#1,Filename

IF YesNo="" THEN
    GOSUB ClearMenu
    LOCATE 0,23
    ATTR 1,2
    INPUT " Print blank lines (y/N)";YesNo
    ATTR 1,0
    CLS

    IF YesNo<>"y" AND YesNo<>"n" THEN
        YesNo:="n"
    ENDIF

    GOSUB ClearMenu
    LOCATE 0,23
    ATTR 1,2
    INPUT " Filter ANSI commands (y/N)";ANSI_Filter
    ATTR 1,0
    CLS

    IF ANSI_Filter<>"y" AND ANSI_Filter<>"n" THEN
        ANSI_Filter:="n"
    ENDIF
ENDIF

LineNumber:=0
```

Figure 1: Sample Modular Listing

By now you probably have the impression that BASIC is the worst language known to man. Actually, nothing could be further from the truth. BASIC is by far the easiest programming language to learn. It is also fairly powerful.

Improve Efficiency

Care must be taken to ensure a program is efficient with respect to memory requirements and execution time. This doesn't mean you should spend all of your time and energy squeezing every single byte and clock cycle out of the program. The best time to optimize the program is during the planning stages. Choose your variables and routines well at the start of the program. Refine the program during its development to ensure maximum efficiency.

Improve Reliability

The use of local variables and localized control flow are strongly recommended. If two or more modules use the same variables, it is easy to forget that fact and accidentally modify the contents of a variable that is currently being used in another module. *Local variables* are variables that are used in only one module. Consider the following example:

```
SUB Module1
  DIM Count:INTEGER
  FOR Count:=1 TO 5
    PRINT COUNT
    CALL Module2
  NEXT Count
ENDSUB

SUB Module2
  DIM Count:INTEGER
  FOR Count:=1 TO 5
    PRINT COUNT;
  NEXT Count
  PRINT
ENDSUB
```

The variable `Count` is a local variable in both `Module1` and `Module2`. Even though the name of the variable is the same, neither module knows the other is using a variable with the same name. If `Module1` were executed, you would see the following printed on the screen:

```
1
1 2 3 4 5
2
1 2 3 4 5
3
1 2 3 4 5
4
1 2 3 4 5
5
1 2 3 4 5
```

One problem with non-modular languages is that all variables are *global*. That is, each subroutine in a BASIC program can access any variable used by any other subroutine. Consider a similar program written in Disk BASIC:

```
10 FOR C=1 TO 5
20 PRINT C
30 GOSUB 60
40 NEXT C
50 END
60 FOR C=1 TO 5
70 PRINT C;
80 NEXT C
90 PRINT
100 RETURN
```

In this case, the output of the program is entirely different:

```
1
1 2 3 4 5
```

Because the value of `C` is 5 when the subroutine at Line 60 returns, the `FOR` loop in Lines 10 through 40 assume the loop has been completed. The implication for the BASIC programmer (and those using global variables in structured languages) is to be more careful in the use of variables.

Localized control flow means that an `IF` statement, a `FOR` loop or a `GOTO` must be contained within the module for which it is intended. By far the most difficult problem facing BASIC programmers in the realm of localized control flow is the use of `ON ERR GOTO` and `ON BRK GOTO`. Often these two statements are used at the beginning of a program to trap errors and prevent program execution from being abruptly aborted. Although these two statements help make the resulting program more resilient to errors, the flow of the program can suddenly shift from one end of the program to the other without notice.

Minimize Complexity

In a structured program, the overall complexity is dramatically reduced. This means the program is broken down into logical components, and these components are broken down into subcomponents. This process continues until all of the modules are very simple to write and comprehend.

The flow of a well-written program is a natural progression from beginning to end, without program control jumping all over the place. Trying to follow a complex program that isn't well planned can be extremely difficult. In fact, it can be a total nightmare.

Simplify Maintenance

A well-written program is structured in such a way as to ensure easy maintenance, with potential problem areas easy to find and correct. A structured program is also constructed in a way that requires almost no debugging. How, you may ask, can you possibly develop a program that doesn't require debugging? First of all, thoroughly test each module during the development stages rather than waiting until the entire program is written. As an example, let's assume you have written and individually tested Modules B and C for a Super Duper Spreadsheet. When you finish Module A and the program is tied together, the data coming out of Module C isn't correct. But you proved during the development cycle that both Modules B and C handle data as designed. Since Modules B and C are known to be correct, there is no need to even check them. You can immediately surmise the data coming from Module A is incorrect.

Imagine the scenario had you waited until the program was completely written before testing. All it takes is one bad module to make an entire chain look defective. But if each module is thoroughly tested as it is developed, large sections of code can immediately be removed from suspicion. Ask yourself one very important question: Would you rather spend days, or possibly weeks, debugging a large, complex program or spend a few hours testing a small, simple module?

Increase Productivity

Obviously, the process of structured programming forces the programmer to think about every minute detail before writing the program. Instead of jumping feet first into a project, a systematic breakdown of the whole project into its individual components is more beneficial. If you were asked to write a full-featured desktop publishing system, you'd probably cringe at the complexity of the task. On the other hand, if you were asked to write a graphics input routine or a printer dump routine, you should be able to handle the task without much problem. If a task appears complex, stop and think about how you can break it down before attempting to write it.

With each program you write, you might ask yourself the following questions (write your answers on paper): What do I want the program to accomplish? What are the logical steps necessary to achieve that goal? What data structures are the most efficient to achieve the goals of the program? What level of user will be running the program? While this isn't a definitive list of all the questions you should ask, it provides a good start. Each answer often leads to other questions. You may want to pretend you are answering questions from someone with absolutely no idea of what you are talking about. Get a friend to play along. The more work you do in the planning stages, the less work you will do when you actually start writing the program — your productivity will increase.

Disciplined Methodology

This objective refers to the approach you take when actually writing or entering program code. There are four basic approaches

---

Awards, Certificates, Licenses, and Diplomas

First Prize™ features an extensive collection of templates with superbly crafted art and titles. After selecting a template, you choose a border, fill in the date and signature lines, and type in whatever you want to say in the main text area. Many titles can also be edited.

BOWLING AWARD

Presented by the members of the Eastdale Bowling League to

FRED WATSON

for having bowled a perfect game during the Eastdale Fall 1990 Championship Games sponsored by Olympic Hardware

December 1, 1990

[illegible]

First Prize Includes Nine Template Disks, Program Disk, Labeled Disk-Storage Box, 32 Page User's Manual, and 64 Page Template Directory. Price $49.95

Program requires a mouse or joystick, 64K CoCo II or 3, disk drive and compatible printer. Printer support is mostly similar to our Label Designer, but please specify your printer(s) when ordering.

Ordering Instructions: All orders add $3.00 Shipping & Handling. UPS COD (cash only) add $4.00 more. VISA/MC Accepted. PA residents add sales tax.

Zebra Systems, Inc., 121 S. Burrowes Street, State College, PA 16801 (814) 237-2652

NEW! **Deception Path** NEW!

Reach your goal in the maze before the pathways shift. Block your opponent! Requires logic skills. 1-4 players.

128K CoCo3 One Disk Drive Joystick $19.95

NEW! **Soviet Bloc** NEW!

Falling shapes drop faster and faster as you struggle to fit them together in this challenging game.

128K CoCo3 One Disk Drive Joystick $19.95

Solitaire Games

Classic Solitaire (Klondike, Canfield and Pyramid) $14.95

La Belle Lucie $14.95.

Both for the CoCo3 with 1 disk drive and 128K.

Tazman (CoCo3)(requires OS9/II & 512K)...............$24.95
(A game of Space Commerce)
Armchair Admiral (CoCo3).....................................$14.95
(Battleship - play friends or computer!)
Riddle of the Ring (CoCo3/CoCo2) (Text Adventure)..$10.00
Games Pack I (CoCo3/CoCo2)..............................$10.00
(Hangman, Concentration, Mastermind)
Zenix (CoCo3) (Arcade)..$29.95
Crystal City (CoCo3) (Arcade)............................. $34.95

software submissions invited

Checks, Money Orders, MasterCard & Visa $2.50 S&H. COD additional $2.00. WA addresses add 8.2% sales tax.

Eversoft Games, Ltd
P.O. Box 3354
Arlington, Wa 98223

30 day money back guarantee
(206) 653-5263
10am - 6pm PST

you can use to write better-structured programs. We'll look at each of these, and then we'll examine how they are used together.

— Modular Programming

The first step toward writing structured programs — often termed modular programming — is a process known as *divide and conquer*. In this process, you break down the program into its individual components, one step at a time. For example, when writing an adventure game, you don't start programming until you've defined the individual rooms and the mapping from one room to another. In a sense, you can say that a room in an adventure game is like a small module in a complex program, and the mapping defines the flow of the program from one module to another.

The primary goal of the divide and conquer approach is to divide a complex task into a series of smaller, easier tasks that are individual pieces of the puzzle. A large application, for example, might be broken down into Screen I/O and File I/O sections, among others. Further subdivision of Screen I/O may lead you to the routines for direct positioning of the cursor, enabling and disabling video attributes, inserting and deleting lines and characters, etc. If the terminal doesn't provide support for delete line, further subdivision may lead to successive calls to delete characters or to rewriting portions of the screen to simulate the delete line function.

```
/*
**      The functions in this file perform the routine I/O operations
**     for handling the OS-9 file structure. This includes reading and
**     writing a logical sector, converting a logical sector number to
**     the physical track, sector and head number, reading and updating
**     a specified file descriptor, and managing the sector allocation
**      table.
**
**      Structures used:
**          File_Descriptor    A pointer to a file descriptor template
**
**      Global variables:
**           Track                 Physical track number
**           Sector                Physical sector number
**           Head                  Physical head number
**
**      Notes, Bugs, Quirks
**           At the present time, these routines assume the cluster
**          allocation size is one sector. This is fine for floppy and
**           hard drives but should be altered to maintain maximum
**           compatibility and flexibility.
**
**      Revision History:
**           02/15/91 Updated Convert_Segment_List()
**           02/09/91 Updated Read_OS9_Directory()
**           02/07/91 Updated Read_OS9_Directory()
*/
```

Figure 3: Example of Overview Documentation

— Stepwise Refinement

Stepwise refinement is a process in which each module is initially expressed in an English-like language known as pseudocode. The initial draft may, indeed, be written in standard English. For example, one step may be written as "Enter the student's grade and store it in the database." Each successive step in this process further refines the draft from this pseudo-code into the finished source code that is fed through an assembler, interpreter or compiler.

The objective of stepwise refinement is to begin the programming task in everyday language. It is much easier to express a program in English than in an actual programming language. Once each task has been expressed in English, the next step translates the English language to a closer form of the desired programming language. This process continues until the entire program has been translated. The number of steps required depends on the complexity and may be anywhere from one to 1,000 (perhaps even more) steps. The goal isn't to use the shortest number of steps, but to make the transition to the target language as easy as possible.

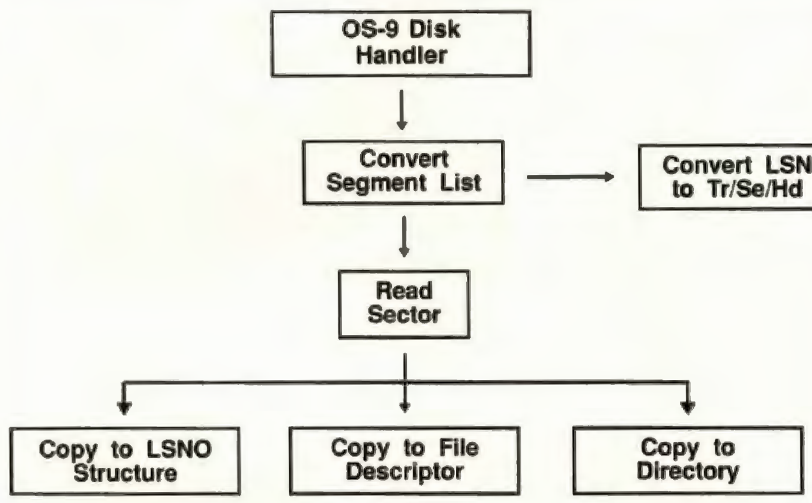


Figure 2

— Top-Down Programming

As its name implies, *top-down programming* is a method in which you develop a program starting at the top and working your way to the bottom. The program is initially broken down into a hierarchy of layers with the entry point of the program in the top layer and the supporting routines in successive lower routines. Figure 2 shows a partial flow chart of one of my latest projects — each box represents one function. In this method, the primary module shown at the top of the chart is written and debugged first, and it is followed by each of the supporting modules in their order of appearance in the lower levels.

— Bottom-Up Programming

Bottom-up programming is the opposite of top-down programming. In this method, you write and debug the modules at the bottom of the hierarchy and progress toward the top. A dummy driver is usually written to simulate the module(s) in the next-higher level to test the modules in the level being written.

— Top-Down Versus Bottom-Up

Top-down programming is the preferred

choice over bottom-up programming because it is the most direct route toward a hierarchical program. The pitfall with bottom-up programming is that it requires you to build the program in reverse order. Subroutines that are composed of several blocks are built from the bottom, which requires writing several disposable drivers to test each block. These disposable drivers aren't needed when building a subroutine from the top. Bottom-up programming also requires writing disposable drivers to simulate modules in higher layers, further adding to the amount of time spent testing and writing redundant code.

— Combined Techniques

Situations often arise where the use of strict top-down or bottom-up programming isn't practical. For example, you may have several modules that are used in several parts of the program. It may be easier to first write and test these common modules. It may also be easier to first write and test individual chains of the program. For example, if you are writing a word processor it may be easier and faster to write and test all of the screen-handling modules before working on the routines that depend on these functions. This also allows you to see how these routines work on the screen instead of writing the internal modules first and having to wait to see if they work properly when you do write the screen-handling routines. The advantage in this case is that you won't have to backtrack through previously written modules to prove their accuracy. Without first writing the screen-handling routines, you may assume that certain values that would normally be handed to the screen-handling routines are correct. By actually seeing the results on the screen, such as by watching the cursor moving or the ruler line getting updated, you have achieved direct feedback to prove the accuracy of each routine without worrying about assumptions.

Documentation

When most people think of documentation, they think of the manual that accompanies a product. But documentation goes much deeper than this. While the manual included with a product is certainly important, there are so many beliefs and approaches that we could (and might) print an article dealing only with that topic. In this article, documentation refers to the way a programmer *comments* his work.

Documentation is an integral part of all programs. It is written at three levels:

- Overview
- Organization
- Instruction

```
/*
**     Read an OS-9 directory using the File Descriptor. The directory
**     structure is allocated into a static area of memory. The amount
**    of memory allocated is based on the file's size and rounded to an
**     even multiple of 256 bytes for safety. This area of memory is
**     freed at the beginning of the routine and reallocated upon each
**     call. Note that any "empty" slots in the directory on disk are
**     skipped. In other words, the directory structure is packed on
**     the fly. This is done to prevent massive calculation later.
**
**     Parameters:
**          Drive           Drive number (0 or 1)
**          *File_Des       A pointer to the file descriptor
**          *NumEntries     A pointer to the number of entries read
**
**     Returns:
**          *Directory      A pointer to the static directory structure
**          *NumEntries     The number of directory entries read
**
**     Revision History:
**          02/09/91        Replaced 24-bit integer conversion with
**                          a call to l3tol().
**          02/07/89        Replaced the string copy loop with a call
**                          to strhcpy().
*/
```

Figure 4: Sample Organization Documentation

Overview documentation introduces the program to the reader. If you must choose between the three levels of documentation, overview documentation is the most fundamental and, therefore, the most essential. It is also the easiest to provide and the most stable of the three types of documentation. The basic overview is included at the beginning of the source code and is brief and general. This overview basically describes the function of the program and its primary goal. It also describes the major variables and structures used along with any data files that may be used.

If a program is split into several source files, each file contains an overview description of the file's basic functions. As shown in Figure 3, I usually include an outline of the revision history in the overview documentation. Although the revision history isn't required here, it allows the user to quickly glance at the top of the file to determine which functions have been modified and the date of each modification.

Imagine picking up two disks containing the source code to your latest project. Both files look remarkably similar. You now have the difficult task of determining the disk that contains the latest version of the source code. A cursory glance at the revision history in the overview documentation at the beginning of the file can spell the difference between disaster and relief. If you choose not to include the revision history in the overview documentation, at least include the last modification date.

Organization documentation defines the name and description of each function used in the program. Typically, the organization documentation is defined immediately prior to the function itself and explains what the function does — not how it works. Also included here is a revision history with the dates and descriptions of each modification. If more than one person is working on the same source file, it is a good idea to initial each modification as well. An example of organization documentation is shown in Figure 4.

Program instruction documentation explains what a specific line or block of source code does or how it works. Program instruction documentation is used sparingly. Adding comments to the source code such as "Increments J by one" is both useless and redundant. Don't include program instruction documentation for blocks of source code with obvious meanings.

Summary

I have only skimmed the surface of structured programming. Topics that I haven't covered include diagramming techniques, the various types of charts and diagrams, analysis and design, as well as many others. But I have given you the basic information you need to begin methodical programming practices. If you want to learn more about structured programming, look in your local library for one of the many books on the subject. Look primarily under the category Programming, Structured. You may also find similar books under the guise of structured techniques. Happy hunting, and may you find many pleasures in your programming efforts.

Note Writer 2 x 40

by John Musumeci

Have you ever wanted to print a note or letter without hassle? *NoteRite 2x40* is similar to an earlier program of mine that was featured in the February 1991 issue of THE RAINBOW. This new version has many enhancements.

NoteRite 2x40 prints an 80-column letter — it now has two side-by-side, 40-column screens. There is automatic switching between the left screen and the right screen. You can see either screen by pressing CTRL.

To view the menu press CLEAR. At the menu you can save the letter, print, continue writing once the first part of the letter is printed, write a new letter, load text, quit or return to the writing mode.

To run the program, enter `LOAD "NOTE2X40"` followed by `RUN`. Then you can put the paper in your printer and start writing. Screen movement is accomplished with the arrow keys. The space bar allows you to rewrite or delete any part of the text. Use the shifted-arrow keys to move to the screen borders. When you near the end of a line, a tone reminds you that upon completion of the next word, the cursor switches to the next line after pressing the space bar. To add a few small words to the end of a line without beginning a new line, use the right arrow for movement.

There is a high-speed poke in Line 155, but it is only evident when in the Write mode. Line 20 sets the baud rate for your printer and can be changed — or eliminated — as needed. If you write a letter that is longer than one full screen, continue writing (after saving and printing) and the text falls into place. When writing a long letter, I suggest using similar filenames, but adding new digits. For example, `JOHNDOE1` and `JOHNDOE2`.

The Listing: `NOTE2X4Ø`

```
1 'NOTE RIGHT 2X4Ø
2 'WRITTEN BY JOHN MUSUMECI
3 'COPYRIGHT 1991 FALSOFT, INC.
1Ø CLEAR 3ØØØ
2Ø POKE15Ø,7:'*BAUD RATE*
3Ø DIMB$(22)
4Ø WIDTH 4Ø:T=Ø
5Ø CLS5:LOCATE5,1Ø:PRINT"ADJUST
PAPER TO PERFORATION THEN      P
RESS <ANY KEY> WHEN READY."
55 ON BRK GOTO 99Ø
6Ø A$=INKEY$:IF A$="" THEN 6Ø
8Ø CLS:FOR A=1 TO 22:B$(A)=STRIN
G$(8Ø,32):NEXT A
9Ø AA$=STRING$(13,45)+"*LEFT SCR
EEN*"+STRING$(14,45)
1ØØ MID$(B$(22),1,4Ø)=AA$
11Ø AB$=STRING$(13,45)+"*RIGHT S
CREEN*"+STRING$(13,45)
12Ø MID$(B$(22),41,4Ø)=AB$
13Ø LOCATEØ,21:PRINT AA$;
14Ø LOCATE8,22:PRINT"PRESS <CLEA
R> FOR OPTIONS"
15Ø LOCATE8,23:PRINT"  PRESS <BR
EAK> TO QUIT          ";
155 POKE65497,Ø
16Ø X=Ø:Y=Ø:A=1:Z=1
17Ø LOCATEX,Y
18Ø A$=INKEY$:IF A$="" THEN 18Ø
2ØØ IF A$=CHR$(12) THEN 41Ø
21Ø IF A$=CHR$(13) AND Z>4Ø THEN
 59Ø ELSE IF A$=CHR$(13) AND Z<4
1 THEN 61Ø
22Ø IF A$=CHR$(8) AND Z=41 THEN
63Ø ELSE IF A$=CHR$(8) AND X=Ø T
HEN 17Ø ELSE IF A$=CHR$(8) THEN
X=X-1:Z=Z-1:GOTO 17Ø
23Ø IF A$=CHR$(9) AND Z=8Ø THEN
59Ø ELSE IF A$=CHR$(9) THEN 64Ø
24Ø IF A$=CHR$(1Ø) THEN Y=Y+1:A=
A+1:GOSUB 73Ø:GOTO 17Ø
25Ø IF A$=CHR$(94) THEN Y=Y-1:A=
A-1:IF Y<Ø THEN Y=Ø:A=1:GOTO 17Ø
 ELSE 17Ø
26Ø IF A$=CHR$(21) AND Z>4Ø THEN
 X=Ø:Z=41:GOTO 17Ø ELSE IF A$=CH
R$(21) AND Z<41 THEN X=Ø:Z=1:GOT
O 17Ø
27Ø IF A$=CHR$(93) AND Z>4Ø THEN
 X=39:Z=8Ø:GOTO 17Ø ELSE IF A$=C
HR$(93) AND Z<41 THEN X=39:Z=4Ø:
 GOTO 17Ø
28Ø IF A$=CHR$(91) THEN Y=2Ø:A=2
1:GOTO 17Ø
29Ø IF A$=CHR$(95) THEN Y=Ø:A=1:
GOTO 17Ø
3ØØ IF A$=CHR$(189) AND Z>4Ø THE
N 68Ø ELSE IF A$=CHR$(189) AND Z
<41 THEN 7ØØ
31Ø GOSUB 73Ø
32Ø MID$(B$(A),Z,1)=A$
33Ø IF Z<41 THEN S=1 ELSE IF Z>4
Ø THEN S=41
34Ø LOCATEØ,Y:PRINT MID$(B$(A),S
,4Ø);
35Ø X=X+1:Z=Z+1
36Ø IF Z=41 THEN 72Ø
37Ø IF Z=81 THEN 59Ø
38Ø IF Z=69 THEN SOUND2ØØ,2
39Ø IF Z>68 AND A$=CHR$(32) THEN
 59Ø
4ØØ GOTO 17Ø
41Ø LOCATEØ,21:PRINT STRING$(15,
42)+" OPTIONS "+STRING$(16,42);
413 LOCATE8,22:PRINT"  L-LOAD. S
-SAVE. P-PRINTER."
416 LOCATE8,23:PRINT"R-RETURN. C
-CONT. N-NEW. Q-QUIT";:LOCATEX,Y
418 POKE65496,Ø
42Ø A$=INKEY$:IF A$="" THEN 42Ø
43Ø IF A$="L" OR A$="l" THEN 81Ø
44Ø IF A$="S" OR A$="s" THEN 9ØØ
45Ø IF A$="P" OR A$="p" THEN 54Ø
46Ø IF A$="R" OR A$="r" THEN 89Ø
47Ø IF A$="C" OR A$="c" THEN 8Ø
48Ø IF A$="N" OR A$="n" THEN 1ØØ
Ø
49Ø IF A$="Q" OR A$="q" THEN 99Ø
53Ø GOTO 41Ø
54Ø CLS:LOCATE7,2:PRINT"P R I N
T I N G . . ."
55Ø IF T=Ø THEN FOR X=1 TO 12:PR
INT#-2:NEXT X:T=1:AA=Ø:GOTO 57Ø
56Ø IF AA=1 THEN FOR X=1 TO 24:P
RINT#-2:NEXT X:AA=Ø ELSE AA=1
57Ø FOR X=1 TO 21:PRINT#-2,B$(X)
;:NEXT X
58Ø GOTO 89Ø
59Ø Y=Y+1:A=A+1:Z=1:X=Ø
6ØØ GOSUB 73Ø:GOSUB 78Ø:GOTO 17Ø
61Ø Y=Y+1:X=Ø:Z=1:A=A+1
62Ø GOSUB 73Ø:GOTO 17Ø
63Ø GOSUB 78Ø:Z=Z-1:X=39:GOTO 17
Ø
64Ø IF X=39 THEN 65Ø ELSE 67Ø
65Ø GOSUB 75Ø
66Ø X=Ø:Z=41:GOTO 17Ø
67Ø X=X+1:Z=Z+1:GOTO 17Ø
68Ø GOSUB 78Ø
69Ø Z=Z-4Ø:GOTO 17Ø
7ØØ GOSUB 75Ø
71Ø Z=Z+4Ø:GOTO 17Ø
72Ø GOSUB 75Ø:X=Ø:GOTO 17Ø
73Ø IF Y>2Ø THEN Y=2Ø:A=21
74Ø RETURN
75Ø LOCATEØ,Ø:FOR C=1 TO 22
76Ø PRINT MID$(B$(C),41,4Ø);
77Ø NEXT C:RETURN
78Ø LOCATEØ,Ø:FOR C=1 TO 22
79Ø PRINT MID$(B$(C),1,4Ø);
8ØØ NEXT C:RETURN
81Ø CLS
815 PRINT:PRINT"USE UPPER-CASE L
ETTERS"
82Ø LOCATE1Ø,4:LINEINPUT"NAME OF
 FILE: ";F$
83Ø F$=F$+"/TXT"
84Ø CLS:LOCATE7,2:PRINT"L O A D
I N G"
85Ø OPEN "I",#1,F$
```

One-stop CoCo shopping? One choice.

Alpha. Burke & Burke. Danosoft. Disto. Howard Medical. Oblique Triad. Performance Peripherals. SRB. Sundog. Zebra. And MANY more.

The CoCo is our ONLY business. We'll be here when you need us.

Hardware.

Best prices. Full warranty.
(180 days or mfgr. warranty)

CoCoPRO! RS-232 PAK & KIT

A clever hardware serial port created by Marty Goodman. FULLY-compatible with the Slotpak-3, Multipak, or Y-cable. Buy this re-engineered Tandy DC Modem pak complete, or purchase the kit and convert your own pak!

Pak- $44.95 Kit- $26.95

•*Howard Medical*

Slotpak-3 Multipak replacement.	$84.95
IBM Kybd. interface (Puppo)	$79.95

•*Disto*

SALE!

Super Controller II (no-halt)	$99
4-in-1 board	$99
SC2 / 4-in-1 COMBO	$189
One-meg upgrade	$159
Super Controller I (4-DOS)	$89
3-in-1 board	$65

Magnavox 1CM135 RGB Monitor
RGB, CGA, Composite. 2-yr. warr. **$285** ($10 ship US)

AMT Star 2400-baud modem
Hayes-compat, 5-yr. warranty. **$95**

360k Floppy Disk Drive System
Complete, double-sided HH drive. **$119** ($8 ship US)

Adaptec 4000a hard disk board
Rare SCSI board, use with Disto. **$89**
SALE!

512k Upgrade w/Lightning software
The BEST! 120ns DRAM, PP board **$74**

Software.

Best prices, best service. Everyday. Why shop elsewhere?

Alpha Software (OS-9)

SALE!

Data Windows 1.1	$54.95	Data Merger	$12.95
Presto-Partner	$26.95	Level II Tools	$22.95
Disk Mgr. Tree	$26.95	The Zapper	$17.95
Multi-Menu	$17.95	OS9 Lv2 BBS	$26.95

TOOLS II

Introducing a NEW set of 22 OS-9 lv 2 tools to make your computing life easier! Features Window Utilities (such as a screen-saver, a global search & replace w/wildcards, a go command to easily change directories, and more), Process Scheduling utils such as alarm & demon (a sophisticated background task mgr., with scheduling), Script File utils such as ask and recobbler, I/O Port utils including netlink and an autodialer for voice calls (stop redialing!), and handy Calculation Utilities. You'll wonder how you've done without these!

All 22 utilities can be yours for only **$34.95**

1/$10 3/$25 $7/40 12/$55!!

Disk Utility 2.1a	Incredible disk & file-handling tool! For CC 1, 2, 3.
Font Bonanza	Replace the "plain" CoCo-3 screen font!
EZ-Base	User-friendly database with many features. CC 1, 2, 3.
Fast-Dupe 2	Makes disk backups MUCH faster! CC 1, 2, 3.
Multipak Crak	Moves many ROMpaks to disk using MPI. CC 1, 2, 3.
CoCo-3 Utilities	Several useful utils, including CC2-CC3 conversion
CC3 Screen Dump	Dump HSCREEN & PMODE to RS or Epson-comp.
Telepatch-3	Patch and enhance Telewriter-64. For CC 2 or 3.
Tape/Disk Utility	Transfer programs to & from, even a disk at a time!
CoCo Checker	Test drives, controller, ports, chips, more! CC2, part CC3
Blackjack Royale	Hi-res simulation & card-counting tutor! CC 1, 2, 3
Adv. Generator	Create your own 100% ML adventure games! CC 2, 3

COCOPRO!
1334 BYRON YPSILANTI MI 48198

Include $4 S&H on all U.S. orders, 10% CDN. COD $4 additional. VISA/MC add 5%. No P.O. Boxes, please, as we usually ship UPS.

VISA/MC ORDERS 1.800.937.7746
ALL OTHER CALLS 1.313.481.3283
1-9 PM EST MON-SAT

For a 12-mo. subscription to our catalog of used soft- & hardware at INCREDIBLE prices (20-90% off), send $15

```
86Ø FOR A=1 TO 21
87Ø LINEINPUT #1,B$(A)
88Ø NEXT A:CLOSE #1
89Ø LOCATEØ,Ø:FOR C=1 TO 22:PRIN
T MID$(B$(C),1,4Ø);:NEXT C:GOTO
14Ø
9ØØ CLS:PRINT:PRINT"USE UPPER-CA
SE LETTERS":LOCATE1Ø,4:LINEINPUT
"NAME OF FILE: ";F$
91Ø F$=F$+"/TXT"
92Ø CLS:LOCATE7,2:PRINT"S A V I
N G"
93Ø OPEN "O",#1,F$
94Ø FOR C=1 TO 21
95Ø PRINT #1,B$(C)
96Ø NEXT C
97Ø CLOSE #1
98Ø GOTO 89Ø
99Ø POKE65496,Ø:WIDTH32:END
1ØØØ RUN
```

Utility
CoCo 3

Hi-Res Art Pie
by Keiran Kenny

When you are writing graphics programs for HSCREEN4, special consideration must be given to setting screen coordinates. Another effect you might use to spruce up your programs is artifact color patterns. To demonstrate these concepts, HIRESPIE draws a circle at 320,96 with a radius of 170. On HSCREEN2 or a PMODE screen, the equivalent radius would be 85.

Lines 120-140 draw radials at 30 degree intervals to produce a pie with 12 slices. The vertical coordinate for the end of each radial is multiplied by a factor, CR, which is 192/640*1.65 as established in Line 40. The 1.65 should really be 1.7 (1+SIN(45)), but 1.65 keeps the ends of the radials neatly within the circle. In lines 180 and 200, CR is also applied to the vertical coordinate of the HPAINT and HPRINT statements.

Lines 160-220 paint each slice of the pie in a Hi-Res artifact color pattern numbered for your guidance. The statement, POKE &HE79B,196 gains access to the artifact colors and POKE &HE79C,CL establishes the number of the pattern.

Normally, these pokes should be followed by POKE &HE79B,212 and POKE &HE79C,181 to restore normal operating conditions (as in Line 190). If the program in which you want to apply artifact colors includes an ON BRK GOTO statement (and it should), these pokes should also be included in the GOTO address line, as in Line 250.

If you see a color pattern you want to use in your own program, press the space bar to hold the pie on the screen. Note the number and dominant color. Press any key to continue the display. Otherwise, the program continues to display the color patterns 12 at a time, and it repeats after all 255 have been displayed.

The patterns appear best on a black background, but also try HCLS2 and HCLS3 in lines 60 and 240.

The Listing: HIRESPIE

```
Ø 'A HI-RES ART PIE
1 'WRITTEN BY KEIRAN KENNY
2 'COPYRIGHT (C) MARCH 1991
3 'BY FALSOFT, INC.
4 'RAINBOW MAGAZINE
1Ø RGB
2Ø POKE65497,Ø
3Ø ONBRKGOTO25Ø
4Ø PALETTEØ,Ø
5Ø CR=192/64Ø*1.65
6Ø HSCREEN4
7Ø HPRINT(1,Ø),"HSCREEN4"
8Ø HPRINT(1,1),"ARTIFACT"
9Ø HPRINT(2,2),"COLOR"
1ØØ HPRINT(1,3),"PATTERNS"
11Ø HCIRCLE(32Ø,96),17Ø
12Ø FORA=ØTO36ØSTEP3Ø:Z=A/57.295
77951
13Ø HLINE(32Ø,96)-(32Ø+17Ø*COS(Z
),96-17Ø*CR*SIN(Z)),PSET
14Ø NEXT
15Ø FORA=15TO345STEP3Ø:Z=A/57.29
577951
16Ø CL=CL+1:IFCL>255THENCL=1
17Ø POKE&HE79B,196:POKE&HE79C,CL
18Ø HPAINT(32Ø+85*COS(Z),96-85*C
R*SIN(Z)),,1
19Ø POKE&HE79B,212:POKE&HE79C,18
1
2ØØ H=INT((3Ø5+19Ø*COS(Z))/8):V=
INT((96-19Ø*CR*SIN(Z))/8)
21Ø HPRINT(H,V),STR$(CL)
22Ø NEXT
23Ø FORD=1TO1ØØØ:NEXT:IFINKEY$=C
HR$(32)THENEXEC44539:K$=INKEY$:G
OTO24ØELSEFORD=1TO15ØØ:NEXT
24Ø HCLS:GOTO7Ø
25Ø POKE65496,Ø:POKE&HE79B,212:P
OKE&HE79C,181:RGB:CLS:END
```

Game
16K Extended

Car Quest
by Donald A. Turowski

Any parents who have ever taken a trip with their children will appreciate *Car Quest*. *Car Quest* provides a way for your children to pass the time until reaching your destination without asking "when are we going to get there?" after only 30 minutes (or less) on the road.

Parents have used the license plate game (finding license plates on cars from various states and writing them down) as a source of travel amusement for many years. But, as you may have discovered, children quickly tire of writing down the name of the state. So, let the CoCo print a checklist and make the game easier for the child — and the parent!

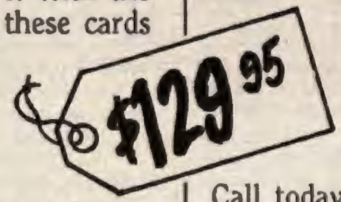
Car Quest is designed to run on a 16K Color Computer with Extended BASIC. A printer with compressed print capability is also needed. The codes for compressed print are located in Line 90 and can be changed to fit your specific printer. The codes in this program are for a Gemini 10X. Also, the printer speed is set for 9600 baud in Line 6. You can change this to POKE 150,87 for 600 baud or any other appropriate baud as desired. Finally, an option for additional copies appears after the checklist is printed. I hope *Car Quest* can make your next trip a little less strenuous.

Turn your Tomcat™ into a TIGER!

The power of the 68000, even from BASIC!

The TIGER is a 68000 co-processor for the TOMCAT TC9 that runs at the blistering speed of 10 Mhz! That's over 5 times faster than the CoCo3! Some functions can be speeded up by as much as 8 times with the TIGER!

Now use all your existing RSDOS software and hardware AND have the power of the 68000 in the same cabinet. Amaze your friends with the speedup possible with the TIGER. If and when you want you can add OS9/Level II and speed that up by a factor of 2 or 3! You could also add OS9/68000 to the TIGER without sacrificing ANY of your existing software OR hardware. OS9/68000 runs on the TIGER with your drives etc. that are running from your TOMCAT TC9! NO extra hardware needed. Later, if you want, you can further improve the performance of the TIGER by adding modules (cards) to the TIGER on the K-Bus! Already there are more than 20 different cards available including memory cards, 2 and 4 port serial cards, 4 port parallel cards, SCSI and floppy controller cards, DMA (Direct Memory Access) cards, and more! With the TIGER you can use these cards even from BASIC! Just 'CALL THE TIGER'



You don't have to spend THOUSANDS to have the power of the 68000! As a matter of fact the TIGER introductory price is ONLY $129.95!

That's right! ONLY $129.95!

Now with the TIGER, FHL, long the leader in Color Computer hardware and software, brings the power of the 68000 within reach of Color Computer users without requiring that you abandon your existing hardware OR software. AND you get the ability to run ALL the software that will be available someday for the more expensive OS9/68000 computers.

So, why spend THOUSANDS today for a computer with little software when for JUST $129.95 you can have the SAME power, run the SAME software, for one TENTH the cost!

The TIGER is in stock for immediate delivery!

Call today for our complete catalog and newsletter with more information on the TIGER and the TOMCAT TC9, It's FREE!

* The TIGER requires the TOMCAT TC9 to run.

ORDERING INFORMATION

VISA and M/C, check and C.O.D. Contential U.S. software shipping add $5.00 Ground - $8.00 Two Day Air. Hardware add $15 ground - $27 Two Day Air. Please call for Next Day Air costs and C.O.D. Foreign add 10% Shipping (Minimum $5 USD). NY residents please add 7% sales tax.

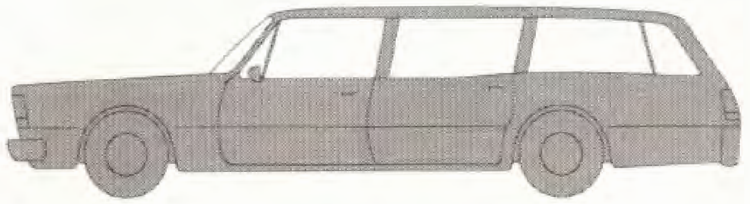
FRANK HOGG LABORATORY

Since 1976
204 Windemere Road
Syracuse, NY 13205
FAX 315/469-8537

Call 315/469-7364

The Listing: CARQUEST

```
1 'CAR QUEST
2 'WRITTEN BY DONALD TUROWSKI
3 'COPYRIGHT 1991 FALSOFT, INC.
6 POKE 150,1
10 CLEAR200:DIM ST$(55)
20 CLS(RND(8)):PRINT@32*6+12,"CA
RQUEST";:PRINT@32*11,"programmed
 by: D.A.TUROWSKI,1987";:SCREEN
0,1
30 GOSUB 260
40 CLS:PRINT"THIS PROGRAM WILL P
RINT A CHECK-LIST THAT CAN BE US
ED AS A GAME OF FINDING LICENSE
PLATES OF    CARS TO PASS YOUR T
IME ON A     TRIP WITH CHILDREN
(OR WIVES)."
50 PRINT:PRINT"THIS PROGRAM IS D
ESIGNED FOR USEWITH A GEMINI-10X
 PRINTER BUT   CAN BE USED WITH
THE PROPER     CODES WITH ANY PR
INTER THAT     HAS CONDENSED PRI
NT CAPABILITY.":SCREEN 0,1
60 GOSUB 260
70 PRINT@32*13,"PRESS ANY KEY TO
 CONTINUE.......";:SCREEN 0,1:EX
EC44539
75 CLS(RND(8)):PRINT@32*6,"stand
 by --printing in progress!";
80 REM ROUTINE TO PRODUCE CHECK
LIST ON PRINTER
90 PRINT#-2,CHR$(27)+CHR$(15)
100 FOR S=1 TO 52
110 READ ST$(S)
130 NEXT S
140 PRINT#-2,TAB(62);"CARQUEST"
150 PRINT#-2:PRINT#-2,"Find the
license plates of the cars from
the list printed below.":PRINT#-
2,"Check off on the dotted line
as you find each one.  GOOD LUCK
!!!!":PRINT#-2
160 FOR S=1 TO 52 STEP 2
170 PRINT#-2,ST$(S);TAB(25);"...
.......       ";TAB(64);ST$(S+1);T
AB(90);"..........       ":PRINT#-
2
180 NEXT S
185 PRINT#-2,TAB(64);"Total numb
er of license plates found ---->
..........":FOR P=1 TO 6:PRINT#-
2:NEXT P
186 CLS:PRINT@32*8+8,"ANOTHER CO
PY (Y/N)";:INPUT RR$
187 IF LEFT$(RR$,1)="Y" OR LEFT$
(RR$,1)="y" THEN RESTORE:GOTO 75
190 END
200 DATA ALABAMA,ALASKA,ARIZONA,
ARKANSAS,CALIFORNIA,COLORADO,CON
NECTICUT,DELAWARE,DISTRICT OF CO
LUMBIA
210 DATA FLORIDA,GEORGIA,HAWAII,
IDAHO,ILLINOIS,INDIANA,IOWA,KANS
AS,KENTUCKY
220 DATA LOUISIANA,MAINE,MARYLAN
D,MASSACHUSETTS,MICHIGAN,MINNESO
TA,MISSISSIPPI,MISSOURI
230 DATA MONTANA,NEBRASKA,NEVADA
,NEW HAMPSHIRE,NEW JERSEY,NEW ME
XICO,NEW YORK,NORTH CAROLINA
240 DATA NORTH DAKOTA,OHIO,OKLAH
OMA,OREGON,PENNSYLVANIA,RHODE IS
LAND,SOUTH CAROLINA,SOUTH DAKOTA
,TENNESSEE
250 DATA TEXAS,UTAH,VERMONT,VIRG
INIA,WASHINGTON,WEST VIRGINIA,WI
SCONSIN,WYOMING,CANADA,END
260 FOR XX=1 TO 1000:NEXT XX:SCR
EEN 0,1
270 RETURN
```



Game
16K Extended

Slot Machine
by Neil Johnson

You no longer have to go to Las Vegas to play the slot machines. With *Slot Machine*, Las Vegas comes to you. Just type this program into your CoCo and win your fortune without leaving home.

You begin play with $20. You can bet any, or all, of this amount at any one time. Wheels turn and, within a few seconds, show four colors. If none of the four colors match, you lose your bet. If two colors match, you win an amount equal to your bet. Three matching colors bring a profit of twice your bet. And if all four colors match, win three times the amount of your bet. Play continues until you run out of money.

Good luck!

The Listing: SLOT

```
1 'SLOT MACHINE
2 'WRITTEN BY NEIL JOHNSON
3 'COPYRIGHT (C) MARCH 1991
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 REM**SLOT MACHINE**
20 REM**BY NEIL JOHNSON**
30 REM**JANUARY 19,1987**
40 CLS
50 PRINT @ 234, STRING$(11,128)
60 PRINT @ 266, CHR$(128)
70 PRINT @ 276, CHR$(128)
80 FOR I=1 TO 6
90 PRINT @ 266+I*32, STRING$(11,
128);
100 NEXT I
110 M=20
120 PRINT @ 10, "SLOT MACHINE"
130 PRINT @ 38, "YOU HAVE";M;"DO
LLARS."
140 INPUT "     BET HOW MANY DOL
LARS"; C$
150 C=VAL(C$)
160 IF C<1 OR C>M THEN GOTO 120
170 FOR I=1 TO 7
180 T(I)=0
190 NEXT I
200 FOR I=1 TO 20
210 SOUND 95,1
220 FOR J=1 TO 4
230 R(J)=(RND(7)*16)+143
240 IF I=20 THEN T((R(J)-143)/16
)=T((R(J)-143)/16)+1
250 PRINT @ (266+J*2), CHR$(R(J)
);
260 NEXT J
270 NEXT I
280 SOUND 200,3
290 FOR I=1 TO 7
300 IF T(I)>=T(1) AND T(I)>=T(2)
 AND T(I)>=T(3) AND T(I)>=T(4) A
ND T(I)>=T(5) AND T(I)>=T(6) AND
 T(I)>=T(7) THEN W=T(I) ELSE NEX
T I
310 W=(W-1)*C
320 PRINT @ 103,"YOU WIN";W;"DOL
LARS!";
330 M=M-C+W*2
340 IF M<1 THEN PRINT @ 137, "YO
U ARE BROKE.":END
350 PRINT @ 130,"PRESS ANY KEY T
O PLAY AGAIN."
360 EXEC 44539
370 FOR I=0 TO 128 STEP 32
380 PRINT @ I
390 NEXT I
400 GOTO 120
```

CBASIC Editor/Compiler
The ULTIMATE Color Computer Basic Compiler!!!

If you want to write fast machine language programs but you don't want to spend the next few years trying to write them in Assembly Language , then CBASIC is the answer!!!
CBASIC is the only fully integrated Basic Compiler and program editing system available for the Color Computer. It will allow you to take full advantage of all the capabilities available in your color computer without having to spend years trying to learn assembly language programming. CBASIC allows you to create, edit and convert programs from a language you are already familiar with Extended Disk Color Basic, into fast efficient machine language programs easily and quickly.

"The most complete Editor/Compiler I have seen for the CoCo... "-The RAINBOW March 1986

CBASIC is a powerful tool for the Beginner as well as the Advanced Basic or M.L. programmer. CBASIC features well over 150 compiled Basic Commands and Functions that fully support Disk Sequential and Direct access files, Tape, Printer and Screen I/O. CBASIC supports ALL the High and Low Resolution Graphics, Sound, Play and String Operations available in Enhanced Color Basic, including Graphics H/GET, H/PUT, PLAY and H/DRAW, all with 99% syntax compatibility.

Specify Coco 1, 2 or 3 Disk $149.00

DataPack III Plus V1.1

AUTOPILOT and AUTO-LOG Processors
X-MODEM DIRECT DISK FILE TRANSFER
VT-100 & VT-52 TERMINAL EMULATION

- No lost data even at 2400 Baud on the Serial I/O port.
- 8 Selectable Display Formats, 32/40/64/80 columns
- ASCII & BINARY disk file transfer via XMODEM.
- Directly record receive data (Data Logging).
- VT-100 emulation for VAX, UNIX and other systems.
- VT-100/52 cursor keys, position, PF & Alt. Kbd. keys.
- Programmable Word Length, Parity, Stop Bits
- Complete Full and Half Duplex operation,
- Send full 128 character set from Keyboard .
- Complete Editor, Insert, Delete, Change or Add .
- 9 Variable length, Programmable Macro Key buffers.
- Programmable Printer rates from 110 to 9600 Baud.
- Send Files from the Buffer, Macro Key Buffers or Disk.
- Display on Screen or Print the contents of the Buffer.
- Freeze Display & Review information On line .
- Built in Command Menu (Help) Display.

Supports: Modem-Pak & Deluxe Pak or Serial Port.
Specify Coco 1, 2, 3 Disk $49.95

EDT/ASM III

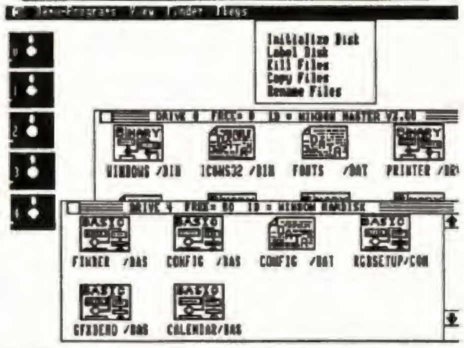
EDT/ASM III is a Disk based co-resident Text Editor & Assembler. It is designed to take advantage of the new features of the COCO 3. It has 8 Display formats from 32/40/64/80 columns. The disk also contains a free standing ML Debug Monitor.
EDT/ASM III has the most powerful, easy to use Text Editor available in and Editor/Assembler package for the Color Computer.
* Local and Global string search and/or replace.
* Full Screen line editing.
* Load and Save standard ASCII formatted files.
* Block Moce & Copy, Insert, Delete, Overtype.
* Create and Edit files larger than memory.

The Assembler features include:
* Conditional If/Then/Else assembly.
* Disk Library files up to 9 levels deep.
* Supports standard Motorola directives.
* Allows multiple values in FCB/FDB directives.
* Allows assembly from Buffer, Disk or both.

Specify Coco 1, 2 or 3 Disk $59.95

CER COMP

Window Master V3.0



The Ultimate User Friendly Point & Click operating System for your Coco 3. Simple enough even for children to use, just point and click to run programs, select files, do disk or file maintenance or almost any task you currently do by typing commands. You also get things like a print spooler, Programmable Function Keys, a Buffered Keyboard, Ramdisk, Serial I/O port and Deluxe Pak support along with Windows, Icons, Buttons, Pull Down Menus, Edit Fields and Mouse functions all in one program. It has multiple fonts in 54 possible sizes and styles, Enhanced Basic Editing and much much more. It add over 50 commands and Functions to Basic to fully support the Point & Click System without OS9.

$49.95

"..It offers so many features that it is probably underpriced. I recommend this software to all CoCo3 owners." -The Rainbow Feburary 1989

It is completely compatible with existing Basic programs and takes absolutely no memory away from basic. It requires 1 Disk Drive, R.S. hires interface & Joystick or Mouse.

Includes 128 & 512K Versions for only $69.95

LOOK!

"The Source"

The SOURCE will allow you to easily and quickly Disassemble machine language programs directly from disk and generate beautiful, Assembler Source Code. And " The Source" has all the features and functions you are looking for in a Disassembler
* Automatic Label generation.
* Allows specifying FCB, FCC and FDB areas.
* Disassembles programs directly from Disk.
* Automatically locates addresses.
* Output listing to the Printer, Screen or both.
* Generates Assembler source directly to disk.
* Built in Hex/ASCII dump/display.
* 8 Selectable Display formats 32/40/60/80.
* Built in Disk Directory and Kill file commands.
* Menu display with single key commands.
* Written in Ultra fast machine language.

Specify Coco 1, 2 or 3 Disk $49.95

Window - Ware

<u>Window Writer</u>- A point & click Word Processor, features both Mouse & Keyboard type editing, proportional printer support, powerful formatting capability, works with any printer. On screen Italic, bold etc. WYSIWYG
"Window Writer is a powerful word processor that is fun to use, very user friendly....... The Enviornment compares favorably to that of Microsoft Windows " - RAINBOW Dec. 1989
Requires Window Master & 512K - $59.95
<u>Window Writer/W</u>- same as above for non Window Master owners, 512K & Disk $79.95
<u>Window Basic Compiler</u>-similar to CBASIC only it compiles all the Window Basic Statements to create super fast Machine Language programs & Desk Accessories. $99.00
<u>Window Edt/Asm</u>- A full featured Editor & Assembler for Window Master. $49.95
<u>Font/Icon Editors</u>- A utility disk with the Font & Icon Editors to create or edit your own, includes Basic & M.L. versions $19.95
<u>Advanced Programmers Guide</u>- A guide for Basic & M.L. programmers on interfacing to Window Masters complete system including System Calls, Memory Map, Interrupt handling & Memory Access. $24.95
<u>The Memory game</u>- A point & Click concentration game, lots of fun. $19.95
<u>Desk Accessory Pak</u>- Installs up to 7 resident D.A. programs, including: Font & Icon Editors, Function Keys, Terminal program, Graphics Editor & Calendar and more. $39.95

CoCo III Tool Kit

Disk Commands	***File Commands***
Backup, Initialize, Directory, Verify, Compare, Search, Edit, Erase, Speed Test, Step Rate Test, Gran Table Analysis & Repair	***Arcive, Copy, Kill, Rename, Erase, View, Edit, Print, Compare, Salvage, Search, Verify, Test Arcive, Un-Arcive, Xmodem Send/Rec.***

Coco Tools is a comprehensive set of disk utilities, providing the most complete set of functions available for the standard R.S. DOS disk system. Comparable in scope and functionality to that of the famous utility available for MS-DOS computers "PC-TOOLS"!
Coco Tools is also the most Comprehensive Disk Repair program available, It Automatically diagnosis and repairs file allocation errors, locates corrupt directory information and cross linked files. It provides fast and easy recovery of deleted files without the drudgery normally associated with killed file recovery. For more info see our other ad in this magazine.

Requires a 128K , Disk and 80 col. display $49.95

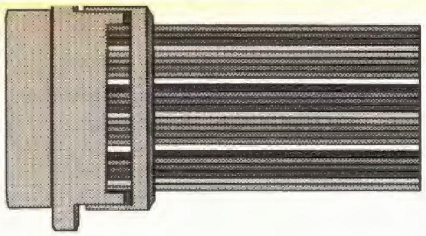
Window Mania

Savings over 50 %

Window Master and Writer, DA Pak,Font/Icon Editors & Memory Game.... $119.95
Window Edt/Asm, Window Basic Compiler and Adv. Programmers guide.....$99.95

VISA, MASTERCARD or COD
Call Toll Free 1-800-383-8529
(Monday thru Saturday, 8am to 5pm PST).
To order by mail, send check or money order for the amount of the program plus $4.00 for shipping to:

Cer-Comp Ltd.
5566 Ricochet Avenue
Las Vegas, NV 89110
702-452-0632



Delphi Bureau

Online Efficiency

by Eddie Kuns
OS-9 SIG Database Manager

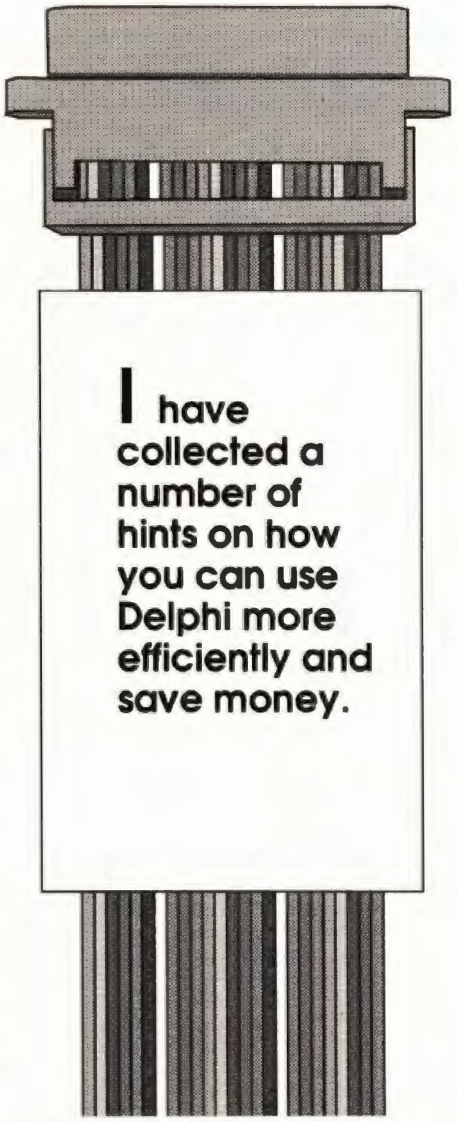
I have collected a number of hints on how you can use Delphi more efficiently and save money. One of the most important ways to budget yourself, of course, is to keep track of the time you spend on Delphi. Unfortunately, I am unaware of any CoCo terminal program that does this for you. However, Delphi keeps track of your total online time.

To look at your Delphi usage history, enter `go using usage`. Next, enter the name of the Delphi account for which you want information (your main account or an associate account). Then press ENTER to get a list of the current month's Delphi usage. You may also enter a range of dates such as `2/1/91, 2/28/91` to see the February usage information. Following a (possibly long) list of Delphi transactions is an estimated total time for the period you requested.

You, too, can keep track of your total time on Delphi. Some terminal programs keep track of elapsed online time. Also, each time you logoff Delphi, it reports the total time you spent online in that session. This should allow you to keep a running total of each month's usage.

The `/timeout` command (and you thought I had covered them all!) may also help you save money. This command sets the length of time in minutes that Delphi waits for a response before automatically logging you off. For example, if you enter the command `/timeout 1`, Delphi signs you off after one minute of idle time. The default timeout is 15 minutes. To make a changed timeout setting permanent (along with any other settings you may have changed), use the `/save` command.

Eddie Kuns is pursuing a PhD in physics at Rutgers University. He lives in Aurora, Illinois, and works as a programmer and researcher at Fermilab. Eddie is co-manager of the CoCo SIG; his username is EDDIEKUNS.



I have collected a number of hints on how you can use Delphi more efficiently and save money.

One night I wanted to download an especially long file, but didn't want to stay awake for the hour I estimated it would take. First, I set my timeout to one minute and then set `/busy`. (I set `/busy` to be polite and avoid someone sending me a message thinking I would see it!) Finally I started downloading the file and went to bed. One minute after the download finished, I was automatically logged off Delphi!

Forum Hints

Some folks have really automated their online time by defining a macro that logs them onto Delphi, captures all new Forum messages to disk, and then logs them off! To read all new forum messages nonstop, use the `read new ns` command. The `ns` option, of course, is short for nonstop. Once you've captured them, you can read the Forum messages at your leisure and edit replies with your favorite editor, all offline. Then, you can post your replies the next time you log onto Delphi.

One easy way to post these replies is to first upload them to your Workspace (using the hints below). Then, in Forum, use the `reply` *number filename* command, where *number* is the message number you are replying to and *filename* is the name you gave the text file in your Workspace. Another way to post these replies is to enter `reply` *number* and, then, ASCII upload the text of the reply.

Workspace Hints

When uploading to your Workspace, you can "batch" upload. This means that you can upload several files in one shot without having to start each upload by hand. To do this, you must have either Ymodem Batch, Zmodem or Kermit, although Kermit is quite slow.

Be careful how much of your Workspace you use. You have only 50 free blocks, or 25K, for storage space. Delphi charges you for disk space used in excess of this amount, taking into account the length of time the space is used. The longer you use disk space, the more you are charged. All mail you receive is stored in your Workspace in files ending in `.MAI`, so it is also important to keep your mailbox clean — delete messages you don't want to keep. However, it is best not to delete files ending in `.MAI`. Mail periodically deletes unused files (and deleted mail messages). If you upload files to your Workspace before submitting them to the databases, you can delete them immediately after finishing the submission. If you send someone a file in mail, you can delete the file from your Workspace immediately after sending the message.

Don't worry about the size of the `MAIL.MAI` file in your Workspace. Even though a mail message is deleted, the space it occupies in the `MAIL.MAI` file isn't removed immediately. Delphi is aware of this and charges you only for the actual space used by the `MAIL.MAI` file.

When you are in Workspace, you can use the `dir/grand` command to show a grand total of disk usage. Another useful command is `dir/exc=.mai` to show all files except those ending in `.MAI`.

Database Hints

When downloading, you want to use the most efficient protocol available. For most people this is either Ymodem or Ymodem Batch. Ymodem sends larger chunks of data than Xmodem before waiting for a response and, thus, can be as much as twice as efficient as Xmodem depending on your modem speed and how you connect to Delphi. In general, the higher the modem speed, the greater the difference. Not all terminal programs support Ymodem, however, so you may want to change to terminal programs that do. Xmodem is usually more efficient than Kermit. If you have access to Zmodem, it is the most efficient of the bunch!

Although you currently cannot truly "batch" downloads from the databases, Ymodem Batch may save you time by automatically sending the filename. This may allow you to start a download faster. This is probably more a matter of convenience than real savings, though.

When submitting files to the OS-9 and CoCo SIG databases, please assign a download name for each file. While this is mainly an aid to those downloading with Ymodem Batch, Zmodem or Kermit (these protocols send the filename along with certain other file information before sending the file), it is also helpful to those using Xmodem and Ymodem, as well as other protocols, to assign an appropriate filename.

If you are submitting a number of related files, it is better to collect them into one group. This makes searching for the files and downloading them somewhat easier since they are all collected in one area. It also saves a little bit of work when submitting — you only have to assign one group name and write one description.

All new uploads to the OS-9 SIG are automatically moved into the New Uploads database for at least one month. When submitting a group of files, however, set the topic you would like the group posted to when moved *from* the New Uploads data-

CoCo III Tool Kit

Disk Commands
Backup, Initialize, Directory, Verify, Compare, Search, Edit, Erase, Speed Test, Step Rate Test, Gran Table Analysis & Repair

File Commands
Arcive, Copy, Kill, Rename, Erase, View, Edit, Print, Compare, Salvage, Search, Verify, Test Arcive, Un-Arcive, Xmodem Send/Rec.

Coco Tools is a comprehensive set of disk utilities, providing the most complete set of functions available for the standard R.S. DOS disk system. Comparable in scope and functionality to that of the famous utility available for MS-DOS computers "PC-TOOLS"!

Coco Tools provides fast and easy operation of standard DOS commands like Copy, Rename, Kill, Disk Initialize and Directory thru a consistent and easy to use interface. It provides easy visual selection of files, so multiple file operations can be carried out with very few keystrokes. Coco Tools is also the most Comprehensive Disk Repair program available, it Automatically diagnosis and repairs file allocation errors, locates corrupt directory information and cross linked files. It provides fast and easy recovery of deleted files without the drudgery normally associated with killed file recovery. It also does lightning fast disk I/O for fast Backup and Copying at speeds you thought only higher priced computers could perform. Coco Tools has multi-file Arciving, Un-Arciving and Test Arciving functions plus the ability to extract any selected group or individual files from a compressed file with easy visual file selection. And allows customizing of display colors, printer/Serial baud rate, Drive step rates and Directory sorting preference. Coco Tools requires a 128K CoCo III, 1 Disk drive and an 80 column display $49.95

To order by VISA, MASTERCARD or COD
Call Toll Free 1-800-383-8529
(Monday thru Saturday, 8am to 5pm PST).
To order by mail, send check or money order for the amount of the program plus $4.00 for shipping and handling to :
Cer-Comp Ltd.
5566 Ricochet Avenue
Las Vegas, NV 89110
702-452-0632

Here at MusicWare, we're prepared to show that our products really perform as advertised, so we're proud to announce a new audio tape!

One For Me

Combining his talents as arranger, song writer, bass guitarist, and jazz musician, Val Burke has come up with a great new album of music. Recorded almost entirely with the aid of CoCo MIDI Pro, this tape is a fabulous example of what a professional can do with the program. Catchy tunes, foot stomping rhythms, and a pleasant variety from slow blues to reggae and latin rhythms are all here. Professionally recorded to preserve the high quality of the master, this is a tape that you'll want to listen to over and over again! Order now while the introductory price is only $7.95

CoCo MIDI Pro is on sale for one more month for only $100, including the MIDI interface box. $50.00 for just the program.

Call or write to get a copy of our free newsletter describing our other products, including exciting new IBM PC releases!

Ordering information: payment by check, money order, or credit card (Visa or Mastercard) is welcome. Please add $3.00 for shipping on all orders.

MusicWare Hamlet Rt. 1261, Seaside, OR 97138
1-800-321-1490 (orders only, please!)
1-503-738-0119

base. It is still in New Uploads for one month, but this assists the database manager in moving the group to the appropriate database topic when the month is over.

Database Information

Several updates were posted in January. Among the updates are *MVCheck, Version 2.2*, and an update to *JBudget*. Also posted was a much faster version of *GDMap*. This program by Mark W. Farrell graphically shows you how full and fragmented your disks are. In the OS-9 Utilities database, Marie-Louis Marcoux submitted an update to the popular disk-zapping program *dEd*. This update aids editing a disk's granule allocation table. Kelly Thompson's *KFormat* utility helps those about to back up a hard drive to multiple floppies. He also posted *The Extractor, Version 2.0*, to better automate the process of dearchiving files.

Dave Philipsen has released Version 2.1 of the popular terminal program *Supercomm*, which supports Ymodem Batch, in the OS-9 Telcom database.

Plenty of information was posted in the CoCo SIG in January about the ZIP code bars you sometimes see on letters. Dan Monday posted two programs based on this information to generate these barcodes on envelopes. Frances Calcraft uploaded information in Utilities & Applications about the MNSS blood group system. Charles Gibson contributed Version 1.1 of an index to RAINBOW files.

Brian Wright submitted two games written by Brian O'Neill. Joe Sannucci released the latest version of *Dungeon Depths* for the CoCo 3, a game including digitized sound! He also contributed blank forms to help you keep track of Delphi online time (neatly fitting into this month's topic).

In Soapbox, Marty Goodman uploaded several articles about the gulf war, which is thankfully over as I write this in early March. Brian Wright submitted the entire text of Dr. Martin Luther King, Jr.'s "I Have a Dream" speech. ❑

Database Report

OS-9 SIG

Applications:
MVCHECK VERSION 2.2
KEITHBAUER Keith Bauer
JBUDGET ARCHIVE UPDATE
JIMBM Jim Manning
FAST GRAPHIC DISK BITMAP DISPLAY
XLIONX Mark W. Farrell
TICTACTOE FOR LEVEL 2 BASIC09
DKINDBERG Darren Kindberg
SUPERIKE - ICON EDITOR
BRIANPAQ Brian Paquette

Utilities:
SCREEN DUMP UTILITY
MARLOU Marie-Louis Marcoux
KFORMAT - FORMAT MULTIPLE DISKS
KMTHOMPSON Kelly Thompson
DEDPLUS CORRECTION
MARLOU Marie-Louis Marcoux
THE EXTRACTOR V2.0!
KMTHOMPSON Kelly Thompson
CLEANUP.PAK
BOBKEMPER Robert Kemper
TIME SHARING UTILITIES
BRUCEISTED Bruce Isted

Device Drivers:
HOW OS9 CAN USE COCO3'S TIMER
PAULSENIURA Paul Seniura
ELIMINATOR SOFTWARE AND MANUAL
BRUCEISTED Bruce Isted
INTERNAL SERIAL PORT DRIVER
RICHKOTTKE Richard Kottke

Patches:
TANDY CC3HDISK PATCH
JEVESTAL Jim Vestal
KYUM-GAI FROM MULTI-VUE
07ESRTIMOTHY Tim Fadden

Telcom:
IRQ HACK 2
COCOKIWI Dennis McMillan
CTS MOD FOR DISTO 4IN1
BRYANC Bryan Clingman
SUPERCOMM VERSION 2.1
DPHILIPSEN Dave Philipsen

Graphics & Music:
SVIEW VERSION 1.1
SEBJMB Jeff Blower
SADDAM IMG FROM NEWSPAPER
045JOE Terry Woods
ROBOCOP GIF'S
RLORBIESKI Richard Lorbieski
DR. WHO LOGO IMG
DEANHOLDER Dean Holder
FRANK N. FURTER IMG
DEANHOLDER Dean Holder
ULTIMUSE FILE EDITOR
JBYRD Jon Byrd

Programmers Den:
INDEX FOR CGFX7.L LIBRARY
COMPER Glen Hathaway
GFX PASCAL SUPPORT LIBRARY
XLIONX Mark W. Farrell
GRAPHIC ORBIT DISPLAY
COMPER Glen Hathaway

Grits & Gravy:
SMILES.PAK
BOBKEMPER Robert Kemper

Tutorials & Education:
HITCHHIKER'S GUIDE TO C
GRIDBUG Mike Stute

CoCo SIG

General Information:
ZIP CODE BAR CODE SPECS
MARTYGOODMAN Marty Goodman

CoCo 3 Graphics:
640 IMG VIEW MASTER
DAVIDMILLS David Mills
KISS1
TRAS Richard P. Trasborg
3 MORE NICE, NUDE 4096ES
STEVEPDX Steve Ricketts
THREE YUMMY 4096 IMG NUDES
STEVEPDX Steve Ricketts
COUNTRY GIRLS IN IMG
LDMOORE Larry Moore
DS69VIEW WITH PRINT OPTION
LDMOORE Larry Moore
PHILADELPHIA TEAM LOGOS
TIND John Tindall
SHAPES 1
TRAS Richard P. Trasborg
MIKE'S GIRLS
SANNUCCI Joe Sannucci

Utilities & Applications:
BARCODE ZIPCODE GENERATOR FOR EN
DANMONDAY Dan Monday
ZIPCODE BARCODE GENERATOR
DANMONDAY Dan Monday
COCO THREE ERROR CODES
FREDMCD Fred McDonald
DISSASS.BAS
DANMONDAY Dan Monday
DISKTEST.UTL 40 OR 80 TRK PATCH
REDCOAT Don Joyce
MNSS BLOOD GROUP SYSTEM
FRANCALCRAFT Frances Calcraft
RAINBOW FILES INDEX 1.1
CHASGIBSON Charles Gibson

Hardware Hacking:
CM1 INFORMATION
MARTYGOODMAN Marty Goodman

Games:
PACDUDE
POLTERGEIST Brian Wright
TETRA
POLTERGEIST Brian Wright
DUNGEON DEPTHS V1.5
SANNUCCI Joe Sannucci

Product Reviews & Announcement:
NEW RS232 PAK FROM COCO PRO
MARTYGOODMAN Marty Goodman
REVIEW OF TANDY WP2
MARTYGOODMAN Marty Goodman

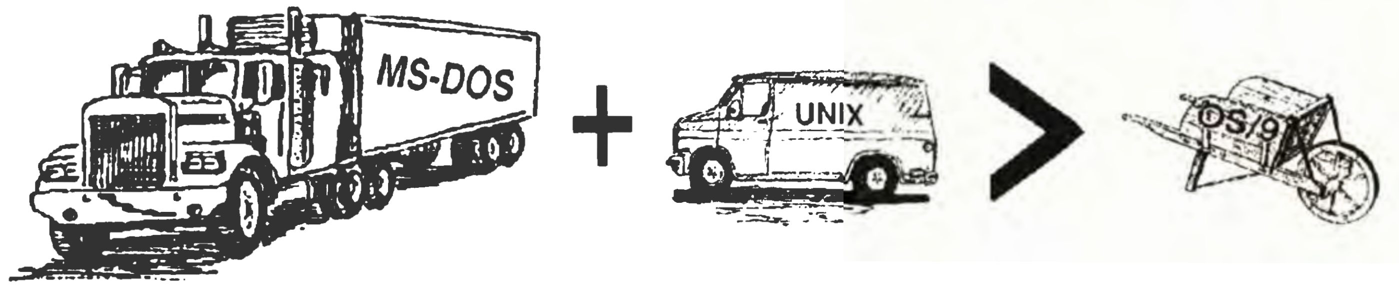
Telecommunications:
ULTIMATERM COLOR CHANGER
POLTERGEIST Brian Wright
DELPHI TIME-LOG
SANNUCCI Joe Sannucci

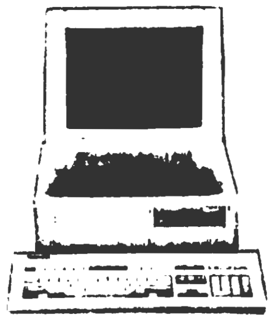
Soapbox (chitchat):
IRAQ WAR PROPRAGANDA
MARTYGOODMAN Marty Goodman
I HAVE A DREAM . . .
POLTERGEIST Brian Wright
THE COST OF THIS WAR
MARTYGOODMAN Marty Goodman
WHY ARE WE AT WAR
MARTYGOODMAN Marty Goodman
THOUGHTS ON THE EVE OF WAR
MARTYGOODMAN Marty Goodman

Would You Buy a Car You Couldn't Get Fuel For?

Why Buy a Computer You Can't Get Software For?

Many of you are now considering an alternative to the Color Computer to run OS/9. Are you really going to invest in an untried alternative with very limited software? Why not buy a computer that will run a UNIX Compatible (OS/9 like) operating system AND MS-DOS. According to published articles, this combination of industry-standard operating systems would run more than 90% of ALL of the software that has EVER been written. Compare the software bases represented below. We call these systems the OWL ATom.™ Can your alternative to the CoCo be fixed locally? On-site service is available most areas on the ATom.





These highly adaptable computers we call the **OWL ATom™** series. They are not newcomers to the computer world, but are the current versions of computers which we have been building for more than 6 years. All systems can be configured with your choice of hard drive, graphics monitor, and 6 different processors. The systems described here are typical of the new versions of our line which offer 2 industry-standard operating systems.

OWL ATom™ ET or SX

For general features of our OWL ATom including expansion capabilities, check out the table below. Comparisons are also given to several Tandy™ computers. The resolution of the optional VGA is also given.

Typical system components:

Processor: 80286-12 or 80386SX-16
Floppy drive: 1.2 or 1.44 Meg
Memory: 1MB
Hard Drive: 40 MB, 28 ms.
Monitor: TTL Monochrome (720X350)

MS-DOS Software:

MS-DOS (with all utilities)
GWBASIC
Integrated Software including:
Desktop Organizer with tracker
Outliner
Word Processor
Spelling Checker
Spreadsheet
Data Base
Graphics program
Communications

UNIX Compatible Software:

Coherent™ (A UNIX Compatible Operating System)
C Compilier
Full Screen Editor
Line Editor
Text Formattor
AWK Language and Yacc
Standard UNIX utilities

Documentation:

Over 1500 pages and VHS Tape

One of the best features is the price:

$1295/$1495

First year on-site service $45 (Most areas)
VGA Color (800X600X256 colors) $400.
Call 215-837-1917 for other options

FEATURES	OWL ATom	2500 XL*	1000 TL/2*
Processor (Speed)	80286(12)	80286(10)	80286(8)
Total Slots	8	3	4
16 Bit Slots	6	3	0
Standard Memory	1024K	1024K	640K
Max. on Board RAM	4096K	1024K	768K
Graphics Output	VGA	VGA	CGA
Max. Resolution	1024X768	640X480	640X200
Drive Slots	5	3	3
Hard Drive Interface	Yes(16 Bit)	Yes(16 Bit)	Yes(8 Bit)
Floppy Drive	1.2 or 1.44 Meg	1.44 Meg	720K
Power Supply (Watts)	200	70	67
Warranty (Parts and Labor)	1 Year	1 Year	1 Year

* Trademarks as follows: Model numbers -Tandy Corporation. MS-DOS- Microsoft, Inc., OS/9 -MicroWare, Inc., UNIX- AT&T, Inc.

OWL-WARE
P.O. Box 116
Mertztown, PA 19539
Call to Order: (800)245-6228
PA and Support: (215)837-1917

OWL-WARE

Proven

On the Razor's Edge of

The New Frontier:

The Most Advanced Color Computer Drive Systems Ever Offered!

Fast No-Halt SCSI Floppies Using Optional SCSI Controller

Proven Performance for Demanding Home or Business Users

This is the most advanced, fully assembeled CoCo hard drive system offered. **Using the optional OMTI 5200 SCSI controller with our Hard Drive Interface, our new system will support no-halt floppy drives.** You need not wait while typing or worry about clock time losses. Why be limited to 3 floppy drives? A complete system could now consist of 1-3 standard CoCo floppy drives, 1-2 (or more) hard drives, and 1-3 no-halt floppies using standard (not just CoCo) OS/9 format. You can use single or double-sided 40 or 80 track drives with the SCSI no-halt controller.

There are several new features with this improved interface. These include:

- **Full SASI/SCSI compatible (this allows many add-ons to the versatile SCSI buss)**
- **No-Halt Floppies with optional SCSI controller allows full type-ahead during access**
- **Low factory-direct prices**
- **Fast Delivery from factory stock**
- **Optional Real Time Clock with built in battery (3-10 year lifetime)**
- **With the Clock you have 240 Bytes of battery backed up RAM for password protection or data storage!**
- **Same super stable LRTech quality**

Interface Price only: $85.

Real Time Clock-RAM: $25.

	20 or 30 Meg.	**40 Meg.**	**80 Meg.**
System Prices: (Includes Hard Drive, case, & fan, SCSI Controller*, LR/OWL Interface, Software. Fully assembled and tested.)	**$495.**	**$535.**	**$875.**
Super System Prices: (LR/OWL System as above but OMTI 5200 SCSI Controller AND 3.5" or 5.5" HD 80 Track Floppy in same case)	**$595.**	**$645.**	**$1039.** (2 cases)

*SCSI controller is OMTI 5100. Add $75 for OMTI 5200 with FDC.

Introducing the *Quick-Link*™ Interface

Provides both a 64K Printer Buffer and Serial and Parallel Interfaces.

Serial to Parallel Printer Interface (64K Buffer) $65.

Converts serial computer output to parallel printer and also provides a 64K buffer in front of the printer. Serial side Female DB-25 and Parallel side is Centronics. Includes Centronics to Centronics cable and power AC-adapter. If you need an adapter from the CoCo 4-pin serial to DB-25 add $9. From 300-38,400 Baud.

Parallel to Serial Interface (64K Buffer) $45.

Same as above except it converts Centronics parallel computer output to DB-25 Female serial. Plug into an IBM printer cable to provide output to a serial plotter.

Parallel to Parallel (64K Buffer) $39.

This is parallel printer buffer with no conversion.

Drive System Parts

Hard Drives

20/30 Meg	ST/PTI	$229.
40 Meg	PTI	265.
80 Meg	ST4096	590.

SCSI Controllers

OMTI 5100	HD Only	$79.
OMTI 5200	HD/FD	169.
OMTI 5400	HD/FD/Tape	199.

(Note: We have no drivers for tape yet)

Cases and Cables

Case, 45 Watt PS, Fan	$105.
Cable set (3 pieces)	25.

Drives have a 1-year limited warranty. Other parts are 90-day warranty. Please Note - At these prices, only very limited support can be given.

Technology

the Color Computer Frontier

DISK DRIVES

Bonus! Special Bundled Software with any Disk Drive Purchase!

Floppy Drive Systems

The Highest Quality for Years of Service

Drive 0 Systems (Half Height, Double Sided, Direct Drives) **$189.**

Drive 0 systems complete with drive, controller, legal DOS, cable, case, power supply, and manual

Drive 1 Systems (Half Height, Double Sided, Direct Drives) **$115.**

New 3.5", 720K Drives for OS-9 with case & Power Supply **$149.**

Drive 1 Systems have drive, case, power supply. (You may require optional cable and/or DOS chip to use)

Special for 0/1 Combos (0,1,2,3) $259.

SALE Prices on Drives!

HALF- HEIGHT DRIVE UPGRADES FOR RS HORIZONTAL CASES

Why only double the capacity of your system when you can triple in the same case? Kit includes: double-sided to fit your case, chip to run both sides of new drive, hardware, and detailed instructions. Easy! Takes only 5 minutes!

Model Only $119.
500, 501, or 502

All drives are new and fully assembled. We ship only FULLY TESTED and CERTIFIED at these low prices. We use Fuji, YE Data, and other fine brands. No drives are used or surplus unless otherwise stated to you when you order. We appear to be the one of the few advertisers in Rainbow who can truly make this claim. We have 7 years experience in the CoCo disk drive market! We are able to provide support when you have a problem.

Drives 1 Year Warranty

OWL Phones

Order Numbers (only)
1-800-245-6228
1-215-682-6855
Fax: 1-215-837-1942
Technical Help
1-215-837-1917

OWL WARE Software Bundle

Disk Tutorial/Utilities/Games

DISK TUTOR Ver 1.1

Learn how to use your disk drive from this multi-lesson, machine language program. This tutor takes you through your lessons and corrects your mistakes for a quick, painless disk drive introduction. (This professionally written tutor is easily worth the bundle's total price.)

3 UTILITIES

A copy verify, copy, and DOS utility.

2 GAMES

We will select 2 games from our stock. These are sold for more than $20 each.

Do not mistake this software with cheap "Public Domain" software which others offer. All of this software is copyrighted and professional in quality. The tutor is unique with us and has helped thousands of new users learn their disk drive.

only $27.95
(or even better)
only $6.95 with any Disk Drive Purchase!!

512K Upgrade

Again at a popular price. Fully assembled and tested before shipping. Easy to install. Uses fast 120 ns. chips.

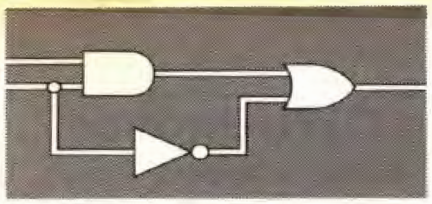
Only $85.

Now includes **memory test, Ram Disk Lighting, Printer Lighting, and Back-up Lighting.** All with an upgraded manual exclusive with OWL!

Our prices include a discount for cash but do not include shipping.

OWL-WARE has a liberal warranty policy. During the warranty period, all defective items will be repaired or replaced at our option at no cost to the buyer except for shipping costs. Call our tech number for return. Return of non-defective or unauthorized returns are subject to a service charge.

OWL-WARE
P.O. BOX 116
Mertztown, PA 19539



BreakPoint

Cause for Arguments

by Greg Law
Technical Editor

Last time, we covered the basic operation of the OS-9 Level II graphics functions that require no arguments, and I provided an overview of macros. This month I discuss the functions that require arguments, and one method of writing them. In the following text you will see references to `BColor` and `BColor()`. `BColor` refers to the generic function and/or display codes, and `BColor()` refers to the specific function written in C.

Functions that require arguments are slightly more complicated because of the conversions involved. For example, `BColor` uses PRN (the Palette Register Number) as a character (one-byte) value. For the sake of simplicity, the program being developed should work with integer (two-byte) values. This is also more logical. This adds the burden of requiring `BColor()` to convert the integer value into a character value. The conversion process is actually quite simple. As shown in Figure 1, an array is declared in the function as storage space for the string. The first and second entries in the array are assigned the escape sequence for `BColor` ($1B33), and then the conversion is performed.

Remember the function needs to convert an integer value to a character value. First `prn` is masked with $00FF to strip the upper byte. However, the result is still an integer (the upper byte is now $00) so the result is cast to a character. To cast a variable or result of an operation from one data type to another, include the desired data type in parentheses immediately to the left of the variable or result. Assume the following statement was used without the parentheses around the formula:

```
buffer[1] = (char) prn & 0x00FF;
```

In this case, `prn` is converted to a character before the calculation is performed. However, the rules of C state that all variables to the right of the equal sign are automatically cast to the highest data type appearing in the statement. Since `0x00FF` is an integer constant, `prn` is cast right back to an integer value, which is not what you want. Therefore, the parentheses are used to force the completion of the calculation first.

The process of casting is often referred to as type conversion since one data type is converted to another data type (i.e. an integer to a character, or a float to a double). Even though most C compilers automatically perform the cast without complaint, I explicitly specify the cast for several reasons. The most important reason is that it tells you a type conversion is being performed. This helps keep it straight that a type conversion is intended and not accidental. The second reason is that the newer ANSI-compliant C compilers perform stricter type checking. These newer compilers still automatically perform the type conversion, but they issue data-type mismatch warnings if the cast isn't explicitly specified. This alerts the programmer to a possible error instead of blindly assuming the programmer is always right. The third reason is to make absolutely certain the compiler performs the proper type conversion.

Another group of functions that must be addressed accept arguments as byte pairs. This is a bit more complex because it requires converting an integer value into its upper- and lower-byte values. For example, `Bar` accepts the *x* and *y* coordinates as HBX, LBX, HBY and LBY. However, the `Bar()` function should accept integer values for the *x* and *y* coordinates and convert them to byte pairs. The resulting source code is shown in Figure 2. The assignment statements for `buffer[3]` and `buffer[5]` are the same as in Figure 1. However, the assignment statements for `buffer[2]` and `buffer[4]` may look a little strange. This conversion process first shifts the values of *x* and *y* to the right eight places. This effectively moves the upper byte to the lower byte. The upper byte is then masked with $00FF and the result is cast to a character value.

Assume a value of 640 ($0280) is passed for the *x* coordinate. You need to split $0280 into $02 (the upper byte) and $80 (the lower byte). You know each byte is made up of eight bits, so one possible method to accomplish this is to divide the coordinate by 256 (2^8). Unfortunately, 640 divided by 256 is 2.5, which may be rounded to 3. Besides that, division is a bit slow because it is handled by a complicated library routine. The process of shifting the coordinate to the right eight places can be demonstrated as follows:

```
00000010 10000000    $0280
00000001 01000000    $0140
00000000 10100000    $00A0
00000000 01010000    $0050
00000000 00101000    $0028
00000000 00010100    $0014
00000000 00001010    $000A
00000000 00000101    $0005
00000000 00000010    $0002
```

In addition to being OS-9 Online SIGop, Greg Law enjoys programming on all types of computers and has worked on systems ranging from the CoCo to the Burroughs B6700 super mainframe. He lives in Louisville, Kentucky.

```
BColor(path, prn)
int path;
int prn;
{
    char buffer[3];

    buffer[0] = '\x1B';
    buffer[1] = '\x33';
    buffer[2] = (char) (prn & 0x00FF);
    write(path, buffer, 3);
}
```

Figure 1: The `BColor()` Function

```
Bar(path, x, y)
int path;
int x, y;
{
    char buffer[6];

    buffer[0] = '\x1B';
    buffer[1] = '\x4A';
    buffer[2] = (char) ((x >> 8) & 0x00FF);
    buffer[3] = (char) (x & 0x00FF);
    buffer[4] = (char) ((y >> 8) & 0x00FF);
    buffer[5] = (char) (y & 0x00FF);
    write(path, buffer, 6);
}
```

Figure 2: The `Bar()` Function

```
BoldSw(path, bsw)
int path;
int bsw;
{
    char buffer[3];

    buffer[0] = '\x1B';
    buffer[1] = '\x3D';

    if(bsw == 0)
        buffer[2] = '\x00';
    else
        buffer[2] = '\x01';

    write(path, buffer, 3);
}
```

Figure 3: The `BoldSw()` Function

```
BoldSw(path, bsw)
int path;
int bsw;
{
    char buffer[3];

    buffer[0] = '\x1B';
    buffer[1] = '\x3D';
    buffer[2] = bsw ? '\x01' : '\x00');
    write(path, buffer, 3);
}
```

Figure 4: Modified `BoldSw()` Function

```
Reverse(path, bsw)
int path;
int bsw;
{
    char buffer[2];

    buffer[0] = '\x1F';

    if(bsw == 0)
        buffer[1] = '\x20';
    else
        buffer[1] = '\x21';

    write(path, buffer, 2);
}
```

Figure 5: The `Reverse()` Function

SYSTEM IV COMPUTER

THE **SYSTEM IV** is a high performance computer system based on the Motorola 68000 microprocessor operating at a clock speed of 16 MHz and has been designed to provide maximum flexibility and versatility. Microware's Professional OS9/68000 operating system is included with the **SYSTEM IV** providing an efficient multi-user and multi-tasking environment. This provides the user with a PC for home use, small business applications and a viable low-cost solution for many industrial control applications (embedded systems). Special requirements (such as midi, sound, A-D/D-A, net-working, etc.) are easily handled with readily available low-cost PC/XT boards which can plug into the **SYSTEM IV** expansion slots. And, as user requirements change or improved special function boards become available, they may be added or replaced at the user's option. Thus, when software requiring multi-media or other new capability becomes a reality, the user will be able to add that capability easily and have the latest technology at his disposal.

TO ACCESS THE LARGEST SOFTWARE BASE available, an MS-DOS board, the ALT86, will be available shortly as a low-cost option. This board has a V30 (8086) microprocessor running at 10 MHz, includes 1 Meg of 0-wait state RAM, uses the Chips and Technology BIOS, has a socket for an 8087 math co-processor and plugs into one of the **SYSTEM IV** expansion slots. Additionally, an OS9/6809 software emulator/interpreter will be available soon. The emulator/interpreter will permit running most COCO OS9/6809 software on the **SYSTEM IV**.

OTHER OPERATING SYSTEMS may be installed. These include CPM, UNIFLEX, MINIX, STARDOS, REXDOS and most any other operating system capable of running on the 68000 microprocessor chip.

THE DESIGN OF THE SYSTEM IV is derived from previously successful designs and uses components that have been tested and proven in other systems. **SYSTEM IV**'s uniqueness stems from the ability of its designer and manufacturer, Peripheral Technology, to provide well designed, reliable hardware at a low cost. Further, only the functions necessary to the basic operation have been designed into the mother board. Seven PC/XT compatible expansion slots allow an unrestricted selection of standard PC/XT accessory boards by the user. The user is not locked into any preconceived notions of what is best.

THE MOTHER BOARD is a 4 layer XT size board which holds the microprocessor, sockets for up to 4 MBytes of 0-wait state RAM, a battery backed-up clock, 4 serial ports, 2 parallel ports, a high density (37C65) floppy disk controller, 7 PC/XT compatible expansion slots, a memory expansion connector to allow an additional 6 MBytes of 0-wait state DRAM, keyboard connector and the necessary system support chips.

THE TERMINAL SYSTEM includes the mother board with 1 MByte of on-board DRAM, a high density floppy disk drive (3 1/2" or 5 1/4"), 4 serial port connectors, a parallel printer port connector, a 200 watt power supply, mini-PC style case capable of holding 5 half-height drives and Professional OS9/68000. This configuration requires the use of an external terminal(s). You may use your existing CoCo, along with a suitable terminal program, as a terminal.

THE CONSOLE SYSTEM adds a VGA (800 x 600 x 16) graphics board and an AT style keyboard and provides full graphics capability at the console. Terminals may be added.

TERMINAL System	$ 999.00
CONSOLE System	$1,149.00

THE **SYSTEM** IV comes with a one (1) year parts and labor warranty.

OPTIONS

3 MByte additional DRAM	$120.00
Hard Disk Controller and driver	$ 69.00
40 MByte Hard Disk	$295.00
20 MByte Hard Disk	$240.00
Additional 5 1/4" or 3 1/2" HD Floppy Drive	$ 92.00
AT Style keyboard and 800 x 600 x 16 VGA Card and driver	$159.00
For 1024 x 768 x 256 VGA Card w/1 Meg of Memory in place of standard VGA card	add $170.00
Mono Display Card in place of VGA card	deduct $ 50.00

Special monitor prices when ordered with the **SYSTEM IV**.

Prices subject to change without notice

For kits, contact Peripheral Technology at 404-984-3742

RAINBOW CERTIFICATION SEAL

delmar co

Middletown Shopping Center - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

The process of shifting a value to the right once is the same as dividing the value by two (2^1), shifting twice is the same as dividing by four (2^2), thrice is the same as dividing by eight (2^3), and so on. The real difference between shifting and division is that shifting is an order of magnitude faster.

`Home`	Homes the cursor
`CurXY`	Positions the cursor at x,y
`ErasLin`	Erases the current line
`ErasEOL`	Erases to the end of the line
`Cursor`	Turn the cursor on or off
`Right`	Move the cursor right
`Bell`	Ring the bell
`Left`	Moves the cursor left
`Up`	Moves the cursor up
`Down`	Moves the cursor down
`ErasEOS`	Erases to the end of the screen
`Cls`	Clears the screen and homes the cursor
`Reverse`	Enable or disable reverse video
`Undrline`	Enable or disable underlining
`Blink`	Enable or disable blinking
`InsLine`	Insert a blank line
`DelLine`	Delete the current line

Figure 6: Function Names

Functions such as `BoldSw` accept a binary argument that is either True to enable the function or False to disable the function. True is a relative term that means any value other than zero, but False is defined as exactly zero. Since you know False is always zero, use a value of zero to determine whether or not to enable to disable the function. The easiest method to determine this is with an `if` statement:

```
if(bsw == 0)
     buffer[2] = '\x00';
else
     buffer[2] = '\x01';
```

The resulting source code is shown in Figure 3. While this method works fine, you may want to consider using the shorthand method of performing the `if` statement. This follows the general form

comparison ? true statement : false statement;

In this form, the `if` is implied and the question mark and colon can be read as `then` and `else`, respectively. Using this form, word the `if` statement in one line as follows:

```
bsw == 0 ? buffer[2] = '\x00' : bu
ffer[2] = '\x01';
```

You may notice a common denominator appears in this statement in the form of `buffer[2] =` *value*. One of the major advantages with this form of the `if` statement is flexibility. Instead of being limited in the strict sense of an `if` statement, it can be used within other statements. For example, it can be used within an assignment statement:

```
buffer[2] = (bsw == 0 ? '\x0
0' : '\x01';)
```

The difficulty with this statement is that you have to read it starting in the middle to understand what is being done. Basically, this can be stated as "If `bsw` is zero, then `buffer[2]` is assigned a value of zero, otherwise `buffer[2]` is assigned a value of one." This statement can be further broken down by taking advantage of an assumption made by C compilers:

```
buffer[2] = (bsw ? '\x01' : '\x00');
```

In this case, `bsw` appears by itself within the comparison section of the statement and an implied comparison with non-zero is performed. That is, if `bsw` is non-zero, `buffer[2]` is assigned a value of one. The modified listing is shown in Figure 4. One of the inherent disadvantages is that the clarity of the listing in Figure 4 is decreased significantly from the listing in Figure 3. However, the listing in Figure 4 is more efficient than the listing in Figure 3. Even so, neither listing is more correct than the other, so use the method with which you feel most comfortable.

The text commands on Page 5-1 of the Windows section of the Level II manual are handled with methods like those discussed so far. One of the differences is that 0 (zero) and 1 aren't used to enable and disable reverse video, blinking and underlining. One of the techniques you can use for reverse video is shown in Figure 5. A boolean value (True or False) is used to determine whether to enable or disable the attribute. It is then converted either to $20 (disabled) or $21 (enabled). This technique can also be used for underlining and blinking. In this way, one function can be used to enable and disable the attribute instead of writing two separate functions. Since all of the text commands aren't assigned official names, a list of function names you might use is shown in Figure 6.

The most complex function to write is `GPLoad`, which accepts an arbitrary amount of character data. The easiest way to write this function is shown in Figure 7. The parameters passed to the function include the group and buffer numbers, format, *x* and *y* dimensions, the number of bytes to be

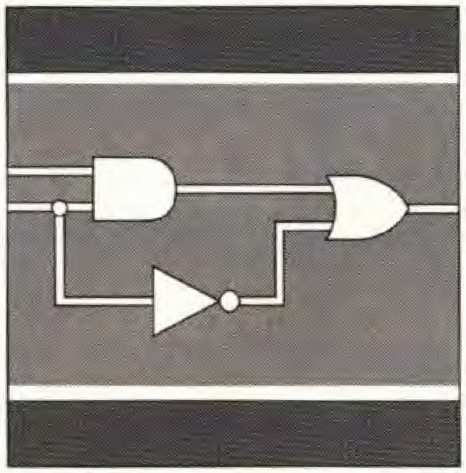
```
GPLoad(path, grp, bfn, sty, x, y, length, data)
int path;
int grp, bfn;
int sty;
int x, y;
int length;
char *data;
{
     char buffer[11];

     buffer[0] = '\x1B';
     buffer[1] = '\x2B';
     buffer[2] = (char) (grp & 0x00FF);
     buffer[3] = (char) (bfn & 0x00FF);
     buffer[4] = (char) (sty & 0x00FF);
     buffer[5] = (char) ((x >> 8) & 0x00FF);
     buffer[6] = (char) (x & 0x00FF);
     buffer[7] = (char) ((y >> 8) & 0x00FF);
     buffer[8] = (char) (y & 0x00FF);
     buffer[9] = (char) ((length >> 8) & 0x00FF);
     buffer[10] = (char) (length & 0x00FF);
     write(path, buffer, 11);
     write(path, data, length);
}
```

Figure 7: The `GPLoad()` Function

```
GPLoad(path, grp, bfn, sty, x, y, length, data)
int path;
int grp, bfn;
int sty;
int x, y;
int length;
char *data;
{
     char *buffer;

     buffer = malloc(length + 11);

     if(buffer == (char *) 0)
     {
          errno = 207;
          return(-1);
     }

     buffer[0] = '\x1B';
     buffer[1] = '\x2B';
     buffer[2] = (char) (grp & 0x00FF);
     buffer[3] = (char) (bfn & 0x00FF);
     buffer[4] = (char) (sty & 0x00FF);
     buffer[5] = (char) ((x >> 8) & 0x00FF);
     buffer[6] = (char) (x & 0x00FF);
     buffer[7] = (char) ((y >> 8) & 0x00FF);
     buffer[8] = (char) (y & 0x00FF);
     buffer[9] = (char) ((length >> 8) & 0x00FF);
     buffer[10] = (char) (length & 0x00FF);

     _strass(&buffer[11], data, length);
     write(path, buffer, length + 11);
     free(buffer);
     return(0);
}
```

Figure 8: Modified `GPLoad()` Function

stored in the buffer, and a pointer to the data to be stored in the buffer. A temporary buffer is allocated and all of the parameters,



except the actual data, are copied into the array. Finally `write()` is called to write the header and then again to write the data.

A more difficult method is shown in Figure 8. In this method, a pointer is declared instead of an array. Then `malloc()` is called to allocate a block of memory large enough to hold the data plus the header. The amount of memory required is determined by adding the length of the data and the size of the header, which is 11 bytes (`length + 11`). To be absolutely certain `malloc()` allocated the requested memory, `buffer` is compared with zero. If `buffer` points to Address 0, which is determined with the statement `buffer == (char*)0`, `malloc()` returned an error due to insufficient memory, so `errno` is assigned a value of 207 (Out of Memory) and -1 is returned to the caller.

Even though you used a pointer to a block of data, `buffer` can still be accessed as an array so most of the original code can be used. Once the header information is copied into the first 11 bytes of `buffer`, the data you want to store in the buffer is copied. The difficulty is that you need to copy `data` into `buffer` starting at the 12th byte (`buffer[11]` through the end). One method is to use a `for` loop as follows:

```
for(i = 0; i < length; i++)
    buffer[i+11] = data[i];
```

Although this achieves the desired results, it is very slow. A faster method uses one of the library routines such as `strcpy()` or `_strass()`. However, `strcpy()` is used to copy null-terminated strings and doesn't work properly with binary data. Fortunately, `_strass()` is designed to work with binary data and accepts a pointer to the block of memory in which to copy the data, a pointer to the data, and the number of bytes to copy. The last two arguments are easy since `data` is the pointer to the data to copy and `length` is the number of bytes. Now you need to determine how to get a pointer to the 12th byte within `buffer`. You could use `buffer[11]`, but that gives the contents of the 12th byte, not the address. However, the ampersand (&) is the "address-of" operator, so use `&(buffer[11])` to get the address of the 12th byte within `buffer`.

Once all of the information is copied, the data is written and the block of memory is deallocated by calling `free()` with the address of the block. Note that because the function returns a value of -1 to indicate an error, it must return a known value to indicate success. Therefore, zero is returned in the last line of the function. The final source code is shown in Figure 8.

The source code provided in the figures is not directly exectuable. The listings are functions you might use in a larger program. The source is on the June RAINBOW ON DISK. Feel free to experiment and learn more about how the graphics functions work.

That's it for this month. Next time we'll look at the system calls, and the Get Status and Set Status calls.

Sale! BUY NOW AT New LOW Prices

BANKMAN. Continuously updates your checkbook. Save, Edit and print files. Lets you analyze spending; reconciles, summarizes banking transactions. Uses 32K. Manual. . . $29.95

GENERAL STATISTICS. Computes over 50 stats, including Standard deviation & error, Covariance, Correlation, Chi-Square and ANOVA. Includes screen tables and manual. Edits, Saves and Prints data. User friendly. Has Manual. . . $29.95

ENVELOPE WRITER. Prints return and mailing addresses directly on the envelope with a DWP or on tractor envelopes with a DMP. Saves on labels! Memorizes return addresses. Prints with custom fonts on TANDY printers. Uses 16K. . . . $17.95

BASEBALL STATS. Keeps track of 20 statistics. Continuously updates. Full of options! Uses 32K. Includes Manual. $29.95
QUICK STATS. A Baseball Scoreboard! Keeps track of over a dozen stats. Lets you save, edit and print out data. . . $22.95
BALL CARDS. A database for baseball card collectors. .$19.95

BRAINBUSTERS. Multiple Choice/True-False Test Writing program. Lets you Save, Edit, Select, Randomize & Print. Many more options! Use professionally or for fun. . . $29.95
THE EXAMINER. Use with above program to take "On Screen" tests. Grades and Saves results. For home or class. . . $19.95
TRIVIA 1-3. Try these BRAINBUSTER quizzes. Each. . . $5.95

INVENTORY MANAGER from Forrest Enterprises. Writes invoices, keeps track of stock, saves time & money. . . .$24.95

COCO LABELS. Prints fancy fonts w/TANDY printers . $19.95
CALCULATOR. Prints out! Works like adding machine. $14.95

Specify COCO 1, 2 or 3. Disk Only. Send check or money order. Add $3.00 Shipping/Handling. Illinois Residents add 6.25% Sales Tax.

TRADING POST
DISTRIBUTORS OF 5-STAR SOFTWARE
P.O. BOX 3453
CARBONDALE, IL 62902

High Finance

Perform a wide variety of financial analysis calculations and create schedules of periodic data with this easy to use Multi-Vue application. Intelligent financial decisions are just a point and click away! Includes fantastic on-line tutorials which help you learn to use the program quickly and easily. Includes present/future value, sinking fund, loan ammortization, depreciation, and much more! Requires Tandy Color Computer 3 w/128k (256k/512k recommended), disk drive, mouse/joystick, OS-9 Level 2 w/Windint module. Multi-Vue recommended.

$24.95 plus $2.50 S & H

Software Developers: MV Systems needs quality OS-9 software to market. If you have written (or are writing) software you would like to market, contact us for details!

MV Systems
P.O. Box 818
Arvada, CO 80001
(303) 420-7777

The OS-9 and Multi-Vue specialists!

A revolutionary evolution for BASIC on the CoCo 3

BASIC+

by Geoff Friesen

BASIC is a useful language for creating software, but it can be improved — better editing, graphics and disk tools would be welcome. Are you a BASIC user who wants improvements but doesn't know where to begin? If so, you're part of the reason I created *BASIC+*.

A wide variety of machine-language utilities have been written to overcome limitations in BASIC. Many of these utilities have been featured in THE RAINBOW. These utilities certainly help make BASIC easier to work with, but unfortunately there are some problems with many of them. For example, each desired utility must be loaded separately, which can be time-consuming. Also, there is no guarantee they will not overwrite each other in memory, which can lead to strange bugs and crashes. Utilities often take a lot of memory normally used by BASIC programs. Many utilities cease operation when you press the Reset button. *BASIC+* was created to solve some of these problems. (However, even *BASIC+* removes some memory from BASIC programs.)

BASIC+ is a platform that extends the CoCo's BASIC through the integration of specially written machine-language utilities. It has been designed exclusively for the CoCo 3 and Radio Shack Disk BASIC 1.1. Even if you do not have this equipment, you should still read this article. It is possible to adapt *BASIC+*'s concepts and techniques to most any Color Computer system.

Geoff Friesen has a bachelor of science degree in computer science and mathematics. He is the author of several published articles about computers. He can be contacted at General Delivery, Dauphin, MB R7N 2T3, Canada; (204) 638-7302.

Getting Started

It's time to see *BASIC+* in operation. First, format a new disk using the `DSKINI` command. Enter listings 1 through 14, and thoroughly check each listing for errors before saving it to disk. It is important that you use the filenames specified in the listings. Listings 3 through 10 and Listing 12 must be saved in ASCII format. (For example, `SAVE "RESET.DAT",A` saves `RESET.DAT` in ASCII format.) This will allow you to merge the files to create the machine-language modules. When you have finished entering and saving the listings, make a backup of the disk — it would be a shame for something to destroy all that work (e.g. the `MKMOD` program updates the disk directory and can trash the disk if it is incorrectly entered).

With the backup disk in the drive, create each of the eight modules by loading `MKMOD.BAS` and merging the appropriate `.DAT` file. For example, to create `RESET.MOD`, enter `LOAD "MKMOD"` followed by `MERGE "RESET.DAT",R`. The `,R` instructs BASIC to run the complete program. It takes several seconds for `MKMOD` to create each `.MOD` file. Follow a similar procedure to make all of the modules for listings 3 through 10.

When all of the modules have been created, load `CONFIG.BAS` and merge it with `SAMPLE.CFG`. `SAMPLE.CFG` is a configuration file that contains data statements with the names of the modules to be installed. The name of each module is read from this file and displayed with a `LOADED` message underneath. If an error message is displayed, one or more of the eight modules is not present. Two other indicators are also displayed: `MS` and `BI`. `MS` (Module Size) is a tally of the number of bytes in the module file (excluding system bytes put in the file by BASIC). `BI` (Bytes Installed) is the actual number of bytes installed. The number of bytes taken up by the installation/initialization code is `MS-BI`.

You can create as many different configuration files as you want. It is not necessary to use `.CFG` as an extension, but it may be helpful. It is necessary, however, to save these configuration files in ASCII format so they can be merged with `CONFIG`. Also, don't forget to use an asterisk (*) as the last data item in the configuration file, as shown in Listing 12. (Note that an asterisk (*) must also appear at the end of all `.DAT` files). Some modules require the installation of other modules prior to their own installation. Place the names of dependent modules after the names of independent modules.

When `CONFIG` finishes, you should notice a > prompt rather than OK. `UTIL.MOD`, one of the *BASIC+* modules, allows you to customize the prompt. Following is a brief overview of each module.

— Reset Protection

During normal operation, pressing the Reset button re-creates the RAM image of the internal ROM. This wipes out any patches made to the image. However, the `RESET` module patches the Reset vector during initialization and prevents this from occurring. You do not need to use `RESET.MOD`, but if you expect you'll press Reset, it certainly helps.

— Fixes

This module repairs three bugs found in the BASIC interpreter. If any further bugs are found, the additional corrections are placed here. The octal routine contains a bug that causes 8 to be considered a valid octal digit — only 0 through 7 are valid. The other two corrections were adapted from the article

Wake up and smell the CoCo!

Bring your drawings and designs to vivid, colorful life with the amazingly popular CoCo Max III!

NOW $39.95

Zoom! Rotate! Animate! 13 Fantastic Fonts! 40 paint brush shapes! CoCo Max III dazzles the eye! Get crisp, clear picture resolution (two full hi-res 320×192 screens)... a big, big edit window... 28 "eat-your-heart-out" drawing tools. Rotate at miniscule 1.5° angles. Undo boo-boos with a click. Animate for special effect. Pick'n'choose from 16 colors out of 64. Go absolutely fruity with fonts (8 sizes, 5 styles & more available!) and an astounding 40 different brush styles! No wonder this software is recognized as the CoCo community's all-time favorite!

Works with both 128K and 512K. Colors print in 5 shades of gray. Printers supported: Epson RX, FX, MX, LX and compatibles; Star/Gemini NX-10, NX-1000; DMP100, 105, 106, 110, 120, 130, 200; OKI 82A, 182, 192 CGP-220 (B&W).

CoCo Max III Add-Ons

Max Fonts disks: 95 fonts on 4 disks. .... Now only **$29.95**
Or two sets of two disks each **$14.95**
Max Edit The font editor as creative as your imagination Only **$14.95**
Color Printer drivers NX-1000 Rainbow. CGP-220 and Okimate 20. For glorious color that wins hands down! **$14.95**

Get Max-10 – the CoCo word processor that's fun and furiously fast!

NOW AT LOWEST EVER PUBLISHED PRICE OF $34.95

Get that easy-as-pie Macintosh feel without giving up your first-born! This program will stretch your CoCo to the limit. Mix graphics, text, large headlines, arrange multiple columns, you name it... Max-10 will swiftly oblige. Includes paper-saving full-page preview (graphics, too!) to make your prose look proud. This is the CoCo word processor that will leave you speechless. Printers supported: Epson FX, MX, RX, LX and compatibles; DMP 105, 106, 130; CGP 220 (B&W); OKI 182, 92, 192; STAR NX-10, NX-1000.

Max-10 Add-Ons

Max-10 Fonts 36 super fonts on 2 disks. Call for list. .... Now **$14.95**
Spell Checker 35,000 word dictionary. Online spell checking to make you look the perfect professional. **$14.95**

Buy both CoCo Max III and Max-10 **$49.95**

Call 1-800-221-0916 1-203-656-1806

How to order: Use Visa, Mastercard or M.O. C.O.D. add $4.00 extra. Purchase Orders subject to credit approval. Connecticut residents add 8% sales tax. Shipping $4.00 per order, usually UPS ground. UPS 2nd day air $6.00 extra. Next day service available. Canada: $6.00 per order (Airmail). Outside U.S. & Canada add 10% of order total.

COLORWARE
242 West Avenue
Darien, CT 06820

"CoCo 3 Potpourri" written by Michael F. Wiens (June 1988) to correct bugs in the PALETTE command and Hi-Res Tab routine.

— New Errors
This module creates an entry called UR (Until without Repeat) in the error tables. This error code is used by the REPEAT/UNTIL command created by CMDS.MOD. Also, if an error occurs during a program's operation, the editor is entered automatically.

— Commands
The original *BASIC+* commands are located in this module. These commands include: REPEAT, UNTIL, BP, OLD, WAIT and SWAP. The ERROR module must be installed prior to CMDS.MOD.

The REPEAT/UNTIL commands form the backbone of structured-programming loops. Such loops can eliminate the excessive use of GOTO. Zero or more statements are repeated until an expression becomes true. The loop always executes at least once. As in FOR/NEXT loops, REPEAT/UNTIL loops can be nested, but it is not a good idea to enter or exit the loop via a GOTO. If an UNTIL is encountered without a matching REPEAT, a UR error occurs and the ERNO variable is assigned the value of 40 (this assignment happens only during program operation). The syntax is:

REPEAT: *statement(s)* UNTIL *expression*

For example,

```
REPEAT: PRINT I: I=I+1: UNTIL I>20
```

The SWAP command causes the values of two variables to be exchanged. Both variables must be of the same type (string or numeric) or a TM (Type Mismatch) error occurs. SWAP is useful when sorting data within a program. The syntax is:

SWAP *var1, var2*

The OLD command can be a real lifesaver if you accidentally erase a program with NEW before saving it. The program can be recovered only if you have not created any new variables, entered any program lines or generated a syntax error since entering NEW.

WAIT is used to insert a delay into a program. There are two ways to use the command. If WAIT is followed by a colon or end-of-line, the program is delayed until a key is pressed. If WAIT is followed by a number in the range 0-65535, the program delays for that number of seconds divided by 60. For example, WAIT 120 delays for two seconds. The syntax is:

WAIT *number of* $^1/_{60}$ s

BP is a powerful little command. If it is followed by a colon or end-of-line, it generates a short beep similar to SOUND 180,6. If BP is followed by a quotation mark ("), then a specific utility can be executed as explained below. The syntax is:

BP *"utility"*

— Utilities
The UTIL module contains a number of useful utilities. These utilities allow you to enable/disable lowercase text and the BREAK key. You can set the CPU/printer speeds and customize the prompt. The CMDS module must be installed prior to UTIL.

BP "BR 0"	disable break key
BP "BR 1"	enable break key
BP "CS S"	set CPU speed to slow
BP "CS F"	set CPU speed to fast
BP "LC 0"	disable lowercase
BP "LC 1"	enable lowercase
BP "PS 600"	print speed = 600 bps
BP "PS 1200"	print speed = 1200 bps
BP "PS 2400"	print speed = 2400 bps
BP "PT *ptstr*"	prompt = ptstr

The string used for *ptstr* can be 20 characters in length. An at sign (@) is interpreted as a carriage return (an ENTER). For example, BP "PT ABC@" replaces the OK prompt with ABC followed by a carriage return. BP "PT" causes the OK prompt to reappear.

— Disk
This module contains only one utility. The CMDS module must be installed before DISK.MOD.

BP "DS 1"	enable fast disk I/O
BP "DS 0"	disable fast disk I/O

Fast-Disk, an interesting program by Joel Hegberg, was featured in the November 1988 issue of THE RAINBOW. This program allows the disk drives to be accelerated for faster program loads and saves. However, the DSKINI command is unreliable at this higher speed, so always use the BP "DS 0" to disable the faster I/O before using DSKINI.

— Copy
This module is a real timesaver. It allows program lines to be copied from one place to another, and it includes an option to delete the original line. The syntax is:

COPY *srcline* TO *destline* {D}

If the D option is specified, the original line is deleted after being copied to the destination line. The braces should not be entered. For example, COPY 100 TO 500 D copies Line 100 to Line 500 and then deletes Line 100. If Line 100 does not exist, a UL error occurs. Line 500 is overwritten if it exists.

— Dir
Richard Estrado's utility, featured in the June 1989 issue of THE RAINBOW, adds wild-card capability to the DIR command, which is very useful. I have adapted his utility and added file/granule tallies that appear at the bottom of the listing. The syntax is:

DIR *"filespec.ext"*

The parameter *filespec.ext* may contain the * wild-card character. An asterisk (*) may appear in both the filename and extension components. Some examples are as follows:

DIR "DIR.MOD" — show the file DIR.MOD, if it exists
DIR "*.DAT" — show all files with an extension of .DAT
DIR "*.*" — show all files
DIR "RESET.*" — show all files named RESET, regardless of their extensions
DIR — show all files

Technical Notes
BASIC+ loads its utilities into a region of memory known as the USR area. (The USR area is the most common place for utility installation.) This region begins above the BASIC string pool and ends below the ROM area. Specifically, addresses $27 and $28 contain the address of the top of the string pool. (The MSB, or high byte, is in $27, and the LSB is in $28.) Addresses $74 and $75 contain the address of the last (highest) memory location that can be used by BASIC. On the CoCo 3, this last address is always 32767. If you issue a CLEAR 200,30000 statement, BASIC reserves 200 bytes for the string pool and places the address value 29999 into $27 and $28. The USR area then ranges from 30000-32767, inclusive.

The *BASIC+* configuration tool selects the appropriate loading addresses. Therefore, its utilities must be relocatable. In other words, they cannot reference addresses inside themselves using absolute (fixed) addressing. An example of absolute addressing is JMP ERROR. An example of relocatable addressing is JMP ERROR,PCR, or the simpler form, BRA ERROR.

BASIC+ is comprised of 14 BASIC programs, some of which are used to create modules that serve as cornerstones to the whole package. A *BASIC+* module is a machine-language program composed of utilities, and it is stored in a file with a .MOD extension. Since this file has the same physical structure as a `.BIN` file, I used the `.MOD` extension to prevent confusion. This article introduces eight useful `.MOD` files and shows you how to create many more.

[*Editor's Note: Due to space limitations, we are unable to publish the assembly-language source code used to create the modules. However, for those interested in learning more about* BASIC+*'s assembly code, the source listings are included on the June 1991 RAINBOW ON DISK. The commented source code for these modules is compatible with* EDTASM+.]

Each *BASIC+* module must start with an installation/initialization sequence. Each module is responsible for ensuring that it is not installed more than once. Duplicate installation results in less memory for BASIC programs. An error code of -1 is returned if the module already exists in memory. Some modules rely on the previous installation of other modules. An error code of -2 is returned if the module requires the installation of another module not currently in memory. Each module must contain a unique *signature* for identification.

The signatures I chose for the eight modules presented here might cause some problems. For example, consider the creation of a new module that contains the string "RESET," which is currently used as the signature for the `RESET` module. If this new module is installed before the `RESET` module, the `RESET` module assumes it has already been installed and refuses to load. For this reason, future modules will use `BP:` as a prefix for the signature. I suggest you follow this example. The possibility of the duplicate string problem occurring when using the prefix is remote.

I used the *EDTASM+* cartridge-based editor/assembler to enter each module. Unfortunately, the *EDTASM+* cartridge saves only to cassette. I wrote `MKDAT.BAS` (Listing 1) to read the machine-language file from cassette and create an appropriate `.DAT` file on disk. The following instructions show the steps used to create a `.DAT` file. These steps are also relevant to *Disk EDTASM+* users — the `CLOADM` in `MKDAT` can be changed to `LOADM`.

— Write the module on paper and check for errors.
— Enter the module into *EDTASM+*.
— Check for errors and assemble using `A/AO/IM/WR`. The first line must contain an `ORG $7000` directive.
— Enter `ZBUG` and use the `P` (punch) command to save memory to tape.
— Connect the disk drive to the CoCo. Run `MKDAT` and specify the name of the file punched from within `ZBUG`. The tape must be positioned at the start of the file.
— If all goes well, the corresponding `.DAT` file is written to disk.

The `DATA` statement on Line 415 of the `CONFIG.BAS` program contains a default value of 2048. This value represents the maximum number of bytes to reserve for all modules, and it can be adjusted as necessary. `CONFIG` makes sure there is at least 4K available for BASIC programs. If the required 4K is not available, `CONFIG` does not install *BASIC+*, but displays an Insufficient Memory message.

There are two additional programs fundamental to *BASIC+*. The first program (Listing 13) is called `TPL.BAS` (TPL for template). `TPL` illustrates how to write a program to detect a *BASIC+* module. When you write a program that uses *BASIC+* utilities, it is a good idea to first ensure that the appropriate module(s) are present, otherwise your program will, undoubtedly, churn out syntax errors. The other program, `UNLOAD.BAS`, is used to remove *BASIC+* from memory. It clears the `USR` area and resets the computer. The `USR` area must be erased with zeros to eliminate signatures that can cause problems if `CONFIG` is re-run.

CoCo 3 Disk 1.1

Listing 1: `MKDAT`

```
100 'MKDAT.BAS
105 '
110 CLEAR 200,&H7000
115 WIDTH 32
120 PRINT "MKDAT V1": PRINT
125 INPUT "FILE";F$
130 CLOADM F$
135 IF PEEK(&H1E2)=2 THEN 145
140 PRINT "INVALID FILE": END
145 SZ=PEEK(&H7E)*256
150 SZ=SZ+PEEK(&H7F)-&H7000
155 OPEN "O",#1,F$+".DAT"
160 PRINT #1,"900 '"+F$
165 PRINT #1,"901 '"
170 PRINT #1,"902 DATA "+F$
175 LN=903: AD=&H7000
180 LN$=STR$(LN)
185 LN$=RIGHT$(LN$,LEN(LN$)-1)
190 PRINT #1,LN$;
195 PRINT #1,"DATA ";
200 NB=8
205 IF SZ<8 THEN NB=SZ
210 FOR I=0 TO NB-1
215 B$=HEX$(PEEK(AD+I))
220 IF LEN(B$)=1 THEN B$="0"+B$
225 PRINT #1,B$;
230 IF I<NB-1 THEN PRINT #1,",";
235 NEXT I
240 PRINT #1,CHR$(13)
245 SZ=SZ-NB: AD=AD+NB
250 LN=LN+1
255 IF SZ<>0 THEN 180
260 LN$=STR$(LN)
265 LN$=RIGHT$(LN$,LEN(LN$)-1)
270 PRINT #1,LN$;
275 PRINT #1,"DATA *"
280 CLOSE #1
```

RAINBOW ON TAPE/DISK Users

Although the BASIC programs listed in this article are included on RAINBOW ON TAPE, their use requires a CoCo 3 with at least one disk drive and Disk BASIC 1.1.

The listings for *BASIC+* are saved on RAINBOW ON TAPE/DISK in binary format. Before proceeding with the installation, format a blank disk and use the `(C)LOAD` and `SAVE` commands to copy all of the *BASIC+* programs to the new disk. All files with `.DAT` and `.CFG` extensions (listings 3 through 10 and Listing 12) should be saved to the new disk in ASCII format. (See Getting Started, above.)

For those of you who are interested in learning more about *BASIC+*, and perhaps writing your own modules, the commented assembly-language source code for each of the modules is included on this month's RAINBOW ON DISK. You will need *Color Disk EDTASM+* to load the source files, or a utility to transfer the files to cassette for use with the *EDTASM+* cartridge.

A Note to All *BASIC+* Users

Another thing you need to know is that each command in the `CMDS` module requires a token, which I will not address in this article. If you write a BASIC program without installing the `CMDS` module and your program includes some of these commands, the commands are not tokenized and subsequent loading of the program (following `CMDS` installation) generates syntax errors. If you forget to load `CMDS` before entering a program that uses these commands, save your program in ASCII format. Then load `CMDS`, followed by your program, and save your program as a normal BASIC file.

Summary

I hope you enjoy using this package. I also hope that you create your own modules and programs that use these modules. Send your creations to THE RAINBOW for possible publication. If you submit any modules, please use only single-letter or three-or-more-letter names. Also, identify this issue of THE RAINBOW for future reference by new users. If you have any problems using *BASIC+* let me know and I will try to solve them. For a reply, please send an SASE. *BASIC+* is here to stay and the best is yet to come. ❑

Listing 2: MKMOD

```
100 'MKMOD.BAS
105 '
110 CLEAR 1000: DIM S$(2)
115 ON ERR GOTO 435: WIDTH 32
120 PRINT "MKMOD V1": PRINT
125 READ F$: PRINT "MAKING "+F$
130 BC=0
135 READ B$
140 IF B$="*" THEN 160
145 IF B$="@" THEN 155
150 BC=BC+1: GOTO 135
155 READ NZ: BC=BC+NZ: GOTO 135
160 OPEN "D",#1,F$+".MOD",1
165 FIELD #1,1 AS A$
170 LSET A$=CHR$(0)
175 PUT #1
180 LSET A$=CHR$(BC/256)
185 PUT #1
190 LSET A$=CHR$(BC AND 255)
195 PUT #1
200 LSET A$=CHR$(0)
205 PUT #1
210 PUT #1
215 RESTORE
220 READ F$
225 FOR I=1 TO BC
230 READ B$
235 IF B$<>"@" THEN 270
240 LSET A$=CHR$(0)
245 READ NZ
250 FOR J=1 TO NZ
255 PUT #1
260 NEXT J
265 I=I+NZ-1: GOTO 280
270 LSET A$=CHR$(VAL("&H"+B$))
275 PUT #1
280 NEXT I
285 LSET A$=CHR$(255)
290 PUT #1
295 LSET A$=CHR$(0)
300 FOR I=1 TO 4: PUT #1: NEXT I
305 CLOSE #1
310 '
315 'UPDATE DIRECTORY
320 '
325 F$=F$+STRING$(8-LEN(F$),32)
330 F$=F$+"MOD"
335 FOR S=3 TO 11
340 DSKI$ 0,17,S,S$(0),S$(1)
345 FOR I=0 TO 1
350 GOSUB 375
355 NEXT I,S
360 F$=LEFT$(F$,8)
365 PRINT F$+" CREATED"
370 END
375 FOR E=0 TO 3
380 F2$=MID$(S$(I),32*E+1,11)
385 IF F$=F2$ THEN GOSUB 395
390 NEXT E: RETURN
395 C2$=CHR$(2): C0$=CHR$(0)
400 MID$(S$(I),32*E+12,1)=C2$
405 MID$(S$(I);32*E+13,1)=C0$
410 DSKO$ 0,17,S,S$(0),S$(1)
415 E=4: I=2: S=12: RETURN
420 '
425 'ERROR HANDLER
430 '
435 IF ERNO=3 THEN 450
440 PRINT "FILE ERROR IN": ERLIN
445 UNLOAD: END
450 PRINT "OUT OF DATA"
```

Listing 3: RESET.DAT

```
900 'RESET
901 '
902 DATA RESET
903 DATA
30,8D,00,57,E6,8D,00,58
904 DATA
8D,1D,8C,00,00,27,05,CC
905 DATA
FF,FF,20,0F,9E,72,AF,8D
906 DATA 00,3F,30,8D,00,33,9F,72
907 DATA CC,00,51,BD,B4,F4,39,34
908 DATA 14,33,8D,00,34,E6,E4,AE
909 DATA 61,11,93,74,22,16,A6,80
910 DATA A1,C0,26,F1,5A,26,F2,E6
911 DATA E4,4F,34,06,1F,30,A3,E1
912 DATA 1F,01,35,C4,8E,00,00,20
913 DATA
F9,12,B7,FF,DF,6E,9D,00
914 DATA
00,FF,00,52,45,53,45,54
915 DATA 05
916 DATA *
```

Listing 4: FIXES.DAT

```
900 'FIXES
901 '
902 DATA FIXES
903 DATA 30,8D,00,57,E6,8D,00,58
904 DATA 8D,27,8C,00,00,27,05,CC
905 DATA FF,FF,20,19,86,24,B7,88
906 DATA 03,86,10,B7,E6,49,86,7E
907 DATA B7,A3,73,30,8D,00,3A,BF
908 DATA A3,74,CC,00,5B,BD,B4,F4
909 DATA 39,34,14,33,8D,00,4A,E6
910 DATA E4,AE,61,11,93,74,22,16
911 DATA A6,80,A1,C0,26,F1,5A,26
912 DATA F2,E6,E4,4F,34,06,1F,30
913 DATA A3,E1,1F,01,35,C4,8E,00
914 DATA 00,20,F9,46,49,58,45,53
915 DATA 05,86,28,D6,E7,26,07,D6
916 DATA 89,C4,1F,7E,A3,77,C1,01
917 DATA 26,05,8E,10,20,20,04,48
918 DATA 8E,10,40,F6,FE,02,7E,A3
919 DATA 7C
920 DATA *
```

Listing 5: ERROR.DAT

```
900 'ERROR
901 '
902 DATA ERROR
903 DATA 30,8D,00,54,E6,8D,00,55
904 DATA 8D,24,8C,00,00,27,05,CC
905 DATA FF,FF,20,16,30,8D,00,46
906 DATA BF,E4,B1,86,7E,B7,AC,6B
907 DATA 30,8D,00,4E,BF,AC,6C,CC
908 DATA 00,58,BD,B4,F4,39,34,14
909 DATA 33,8D,00,55,E6,E4,AE,61
910 DATA 11,93,74,22,16,A6,80,A1
911 DATA C0,26,F1,5A,26,F2,E6,E4
912 DATA 4F,34,06,1F,30,A3,E1,1F
913 DATA 01,35,C4,8E,00,00,20,F9
914 DATA 45,52,52,4F,52,05,C1,50
915 DATA 26,0D,BD,B9,5C,BD,B9,AF
916 DATA 30,8D,00,1B,7E,E4,96,7E
917 DATA AC,49,96,68,4C,26,03,7E
918 DATA AC,73,BD,BD,C5,BD,B9,5C
919 DATA DC,68,DD,2B,7E,85,36,55
920 DATA 52
921 DATA *
```

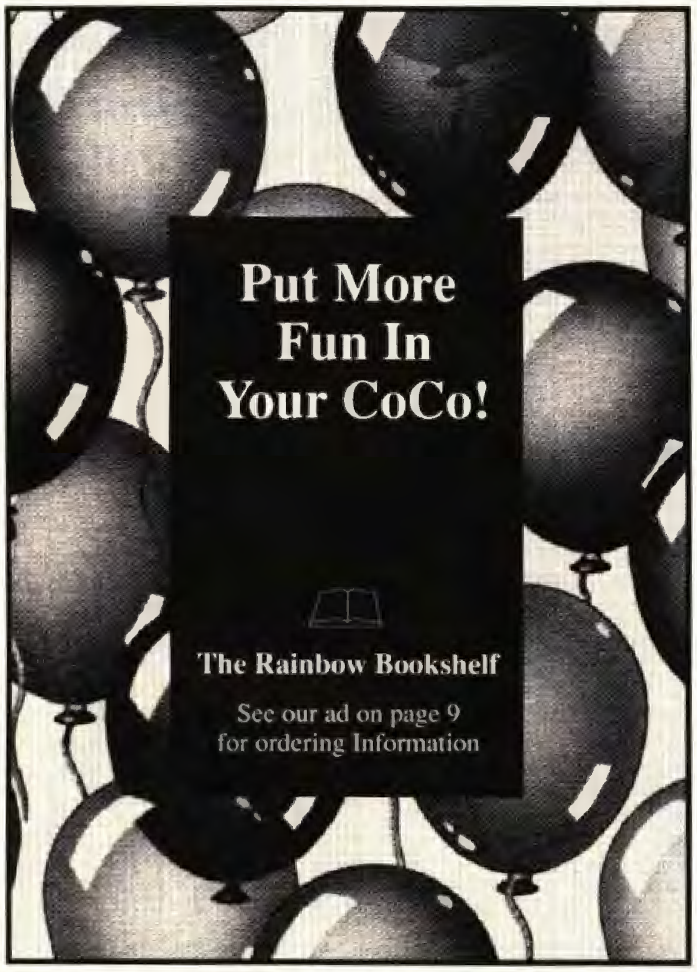
Listing 6: CMDS.DAT

```
900 'CMDS
901 '
902 DATA CMDS
903 DATA 30,8D,00,AC,E6,8D,00,AC
904 DATA 8D,76,8C,00,00,27,05,CC
905 DATA FF,FF,20,68,30,8D,00,92
906 DATA E6,8D,00,93,8D,62,8C,00
907 DATA 00,26,05,CC,FF,FE,20,54
908 DATA 30,8D,01,8F,33,8D,00,85
909 DATA EF,81,33,8D,00,92,EF,81
910 DATA 33,8D,00,B1,EF,81,33,8D
911 DATA 00,CA,EF,81,33,8D,00,E2
912 DATA EF,81,33,8D,00,F4,EF,84
913 DATA 8E,E2,36,33,8D,01,1E,A6
914 DATA C0,A7,80,8C,E2,4F,23,F7
915 DATA 86,1D,B7,E1,62,30,8D,01
916 DATA 24,BF,E1,A1,86,FE,B7,E1
917 DATA 97,30,8D,00,FC,AF,8D,01
```

```
918 DATA 4E,CC,00,B0,BD,B4,F4,39
919 DATA 34,14,33,8D,01,43,E6,E4
920 DATA AE,61,11,93,74,22,16,A6
921 DATA 80,A1,C0,26,F1,5A,26,F2
922 DATA E6,E4,4F,34,06,1F,30,A3
923 DATA E1,1F,01,35,C4,8E,00,00
924 DATA 20,F9,45,52,52,4F,52,05
925 DATA 43,4D,44,53,04,26,10,C6
926 DATA 03,BD,AC,33,DE,A6,9E,68
927 DATA 86,F9,34,52,7E,AD,9E,39
928 DATA BD,B1,41,9D,A5,26,F8,32
929 DATA 62,A6,E4,81,F9,27,05,C6
930 DATA 50,7E,AC,46,96,4F,26,0B
931 DATA AE,61,9F,68,AE,63,9F,A6
932 DATA 6E,F8,05,35,D2,27,0A,81
933 DATA 22,26,7F,9D,9F,6E,9D,00
934 DATA CE,C6,B4,D7,8C,0F,8D,C6
935 DATA 24,D7,8E,BD,A9,56,8E,4E
936 DATA 20,7E,A7,D3,26,B9,9E,19
937 DATA 33,04,A6,C0,26,FC,EF,84
938 DATA CC,FF,FF,BD,AD,03,30,02
939 DATA 32,62,CE,AC,73,34,40,7E
940 DATA AD,1F,26,03,7E,AD,FB,BD
941 DATA B7,3D,9D,A5,26,0B,CE,01
942 DATA 12,4F,5F,ED,C4,AC,C4,22
943 DATA FC,39,BD,B3,57,D6,06,34
944 DATA 14,BD,B2,6D,BD,B3,57,1F
945 DATA 13,35,14,9D,A5,26,1A,D1
946 DATA 06,27,05,C6,18,7E,AC,46
947 DATA C6,05,34,04,A6,80,E6,C0
948 DATA A7,5F,E7,1F,35,04,5A,26
949 DATA F1,39,7E,B2,77,52,45,50
950 DATA 45,41,D4,55,4E,54,49,CC
951 DATA 42,D0,4F,4C,C4,57,41,49
952 DATA D4,53,57,41,D0,F6,36,E5
953 DATA F0,E6,88,E5,45,E6,CF,E6
954 DATA F4,EB,F5,EA,49,E8,82,ED
955 DATA E5,ED,ED,ED,58,EF,3F,E3
956 DATA D4,E3,E6,F8,D2,F9,25,E7
957 DATA 61,E7,65,F3,9D,E6,76,E6
958 DATA 74,F9,B9,00,00,00,00,00
959 DATA 00,00,00,00,00,00,00,00
960 DATA FF
961 DATA *
```

Listing 7: UTIL.DAT

```
900 'UTIL
901 '
902 DATA UTIL
903 DATA 30,8D,00,73,E6,8D,00,73
904 DATA 8D,3E,8C,00,00,27,05,CC
905 DATA FF,FF,20,30,30,8D,00,5A
906 DATA E6,8D,00,5A,8D,2A,8C,00
907 DATA 00,26,05,CC,FF,FE,20,1C
908 DATA 30,89,01,17,EE,84,EF,8D
909 DATA 01,6B,33,8D,00,46,EF,84
910 DATA 30,8D,01,4C,30,1F,BF,AC
911 DATA 77,CC,00,77,BD,B4,F4,39
912 DATA 34,14,33,8D,01,51,E6,E4
913 DATA AE,61,11,93,74,22,16,A6
914 DATA 80,A1,C0,26,F1,5A,26,F2
915 DATA E6,E4,4F,34,06,1F,30,A3
916 DATA E1,1F,01,35,C4,8E,00,00
917 DATA 20,F9,43,4D,44,53,04,55
918 DATA 54,49,4C,04,9E,A6,AE,84
919 DATA 8C,42,52,26,06,33,8D,00
920 DATA 36,20,2E,8C,43,53,26,06
921 DATA 33,8D,00,47,20,23,8C,4C
922 DATA 43,26,06,33,8D,00,56,20
923 DATA 18,8C,50,53,26,06,33,8D
924 DATA 00,6C,20,0D,8C,50,54,27
925 DATA 04,6E,9D,00,E8,33,8D,00
926 DATA 8A,9D,9F,9D,9F,6E,C4,81
927 DATA 30,26,04,C6,21,20,06,81
928 DATA 31,26,48,C6,27,9D,9F,81
929 DATA 22,26,40,9D,9F,26,3C,F7
930 DATA E4,2B,39,8E,FF,D8,81,53
931 DATA 27,06,81,46,26,2D,30,01
932 DATA 9D,9F,81,22,26,25,9D,9F
933 DATA 26,21,A7,84,39,81,30,26
934 DATA 05,CE,12,12,20,07,81,31
935 DATA 26,11,CE,CA,20,9D,9F,81
936 DATA 22,26,08,9D,9F,26,04,FF
937 DATA A2,0A,39,7E,B2,77,24,FB
938 DATA BD,AF,67,81,22,26,F4,9D
939 DATA 9F,26,F0,9E,2B,8C,02,58
940 DATA 26,04,C6,57,20,10,8C,04
941 DATA B0,26,04,C6,29,20,07,8C
942 DATA 09,60,26,D7,C6,12,0F,95
943 DATA D7,96,39,27,CE,81,22,26
944 DATA 14,9D,9F,26,C6,30,8D,00
```


Put More Fun In Your CoCo!
The Rainbow Bookshelf
See our ad on page 9 for ordering Information

EVERYONE'S TALKING ABOUT...

DUAL HI-RES JOYSTICK ADAPTER
Colorware Hi-Res Tandy Hi + Low Res, cassette jack $40

HI & LO-RES JOYSTICK ADAPTER
Tandy Hi + Lo-Res $27

HAWKSoft keyboard cable $25
DOMINATION war game $18
MYDOS extended DOS for you! $15

HAWKSoft
P.O. Box 7112 Elgin, Il 60121
(708) 742-3084 eves and ends
SASE for more info and price list.
S/H (US & CAN) always included
M.O. Check C.O.D. no credit cards
1 year warranty on ALL hardware !!

```
945 DATA 37,86,4F,A7,80,CC,4B,0D
946 DATA ED,81,6F,84,39,9E,A6,33
947 DATA 8D,00,25,5F,A6,80,27,E5
948 DATA 81,22,27,13,81,40,26,02
949 DATA 86,0D,A7,C0,5C,C1,14,26
950 DATA EB,A6,80,81,22,26,CE,A6
951 DATA 84,26,CA,6F,C4,9F,A6,39
952 DATA 3E,00,00,00,FF,FF,00,00
953 DATA FF,FF,00,00,FF,FF,00,00
954 DATA FF,FF,00,00,FF,FF,00
955 DATA *
```

Listing 8: DISK.DAT

```
900 'DISK
901 '
902 DATA DISK
903 DATA 30,8D,00,6A,E6,8D,00,6A
904 DATA 8D,35,8C,00,00,27,05,CC
905 DATA FF,FF,20,27,30,8D,00,51
906 DATA E6,8D,00,51,8D,21,8C,00
907 DATA 00,26,05,CC,FF,FE,20,13
908 DATA 30,89,01,17,EE,84,EF,8D
909 DATA 00,B2,33,8D,00,3D,EF,84
910 DATA CC,00,6E,BD,B4,F4,39,34
911 DATA 14,33,8D,00,A1,E6,E4,AE
912 DATA 61,11,93,74,22,16,A6,80
913 DATA A1,C0,26,F1,5A,26,F2,E6
914 DATA E4,4F,34,06,1F,30,A3,E1
915 DATA 1F,01,35,C4,8E,00,00,20
916 DATA F9,43,4D,44,53,04,44,49
917 DATA 53,4B,04,9E,A6,AE,84,8C
918 DATA 44,53,27,04,6E,9D,00,64
919 DATA 9D,9F,9D,9F,5F,81,30,27
920 DATA 05,81,31,26,54,5C,9D,9F
921 DATA 81,22,26,4D,9D,9F,26,49
922 DATA 5D,27,1A,86,7E,B7,D7,65
923 DATA 30,8D,00,2C,BF,D7,66,7F
924 DATA D7,C0,86,CD,B7,D7,E0,86
925 DATA 14,B7,D8,16,39,86,7F,B7
926 DATA D7,65,8E,09,85,BF,D7,66
927 DATA 86,03,B7,D7,C0,86,D0,B7
928 DATA D7,E0,86,17,B7,D8,16,39
929 DATA 7F,09,85,96,EA,81,02,26
930 DATA 05,86,29,B7,09,86,7E,D7
931 DATA 68,7E,B2,77,26,2D
932 DATA *
```

Listing 9: COPY.DAT

```
900 'COPY
901 '
902 DATA COPY
903 DATA 30,8D,00,48,E6,8D,00,48
904 DATA 8D,18,8C,00,00,27,05,CC
905 DATA FF,FF,20,0A,30,8D,00,39
906 DATA BF,C2,11,CC,00,4C,BD,B4
907 DATA F4,39,34,14,33,8D,00,AB
908 DATA E6,E4,AE,61,11,93,74,22
909 DATA 16,A6,80,A1,C0,26,F1,5A
910 DATA 26,F2,E6,E4,4F,34,06,1F
911 DATA 30,A3,E1,1F,01,35,C4,8E
912 DATA 00,00,20,F9,43,4F,50,59
913 DATA 04,25,03,7E,D3,B9,BD,AF
914 DATA 67,81,A5,26,6E,9E,2B,AF
915 DATA 8D,00,6D,6F,8D,00,6B,9D
916 DATA 9F,24,60,BD,AF,67,27,0C
917 DATA 81,44,26,57,9D,9F,26,53
918 DATA 6C,8D,00,56,9E,2B,AF,8D
919 DATA 00,4C,EC,8D,00,4A,BD,AD
920 DATA 03,24,05,C6,0E,7E,AC,46
921 DATA DE,47,33,44,8E,02,DC,C6
922 DATA 04,37,02,A7,80,5C,4D,26
923 DATA F8,6D,8D,00,2D,27,16,9E
924 DATA 47,EE,84,37,02,A7,80,11
925 DATA 93,1B,26,F7,9F,1B,34,04
926 DATA BD,AC,EF,35,04,AE,8D,00
927 DATA 0D,BF,02,DA,9F,2B,32,62
928 DATA 7E,AC,B0,7E,B2,77,00,00
929 DATA FF,FF,00
930 DATA *
```

Listing 10: DIR.DAT

```
900 'DIR
901 '
902 DATA DIR
903 DATA 30,8D,00,48,E6,8D,00,47
904 DATA 8D,18,8C,00,00,27,05,CC
905 DATA FF,FF,20,0A,30,8D,00,38
906 DATA BF,C1,F1,CC,00,4C,BD,B4
907 DATA F4,39,34,14,33,8D,01,44
908 DATA E6,E4,AE,61,11,93,74,22
909 DATA 16,A6,80,A1,C0,26,F1,5A
910 DATA 26,F2,E6,E4,4F,34,06,1F
911 DATA 30,A3,E1,1F,01,35,C4,8E
912 DATA 00,00,20,F9,44,49,52,03
913 DATA 6F,8D,01,15,6F,8D,01,12
914 DATA 6F,8D,01,0F,81,22,27,05
915 DATA BD,D2,4F,20,0D,BD,C9,35
916 DATA B6,09,4C,BB,09,54,81,54
917 DATA 26,04,6C,8D,00,F3,BD,C7
918 DATA 9D,BD,B9,58,CC,11,02,97
919 DATA EC,D7,EA,C6,03,D7,ED,8E
920 DATA 06,00,9F,EE,BD,D6,F2,35
921 DATA 40,BD,A5,49,34,40,A6,84
922 DATA 27,0F,43,27,19,6D,8D,00
923 DATA C8,26,04,8D,37,27,02,8D
924 DATA 67,30,88,20,8C,07,00,26
925 DATA DE,5C,C1,0B,23,CF,30,8D
926 DATA 00,9F,C6,08,BD,B9,A2,E6
927 DATA 8D,00,A7,4F,BD,BD,CC,30
928 DATA 8D,00,96,C6,08,BD,B9,A2
929 DATA E6,8D,00,97,4F,BD,BD,CC
930 DATA BD,B9,58,39,1F,13,10,8E
931 DATA 09,4C,A6,A0,81,2A,27,0A
932 DATA A1,C0,26,21,10,8C,09,53
933 DATA 23,F0,1F,13,33,48,10,8E
934 DATA 09,54,A6,A0,81,2A,27,0A
935 DATA A1,C0,26,09,10,8C,09,56
936 DATA 23,F0,1C,FB,39,1A,04,39
937 DATA 34,14,C6,08,BD,B9,A2,BD
938 DATA B9,AC,C6,03,BD,B9,A2,BD
939 DATA B9,AC,E6,84,C1,0A,24,03
940 DATA BD,B9,AC,4F,BD,BD,CC,BD
941 DATA B9,AC,AE,61,86,42,AB,0C
942 DATA BD,CD,18,E6,0D,BD,CD,1E
943 DATA 34,02,AB,8D,00,25,A7,8D
944 DATA 00,21,35,04,4F,BD,BD,CC
945 DATA BD,B9,58,6C,8D,00,13,35
946 DATA 94,0D,46,49,4C,45,53,3A
947 DATA 20,20,47,52,41,4E,53,3A
948 DATA 20,FF,00,00
949 DATA *
```

Listing 11: CONFIG

```
100 'CONFIG.BAS
105 '
110 WIDTH 32
115 PRINT "CONFIG V1": PRINT
120 READ BC
125 IF MEM-BC>=4096 THEN 140
130 PRINT "INSUFFICIENT MEMORY"
135 END
140 AD=PEEK(&H27)*256
145 AD=AD+PEEK(&H28)+1
150 CLEAR 200,AD-BC
155 AD=PEEK(&H27)*256
160 AD=AD+PEEK(&H28)+1
165 READ BC: TM=AD+BC
170 ON ERR GOTO 350
175 READ F$
180 IF F$="*" THEN 390
185 PRINT F$: F$=F$+".MOD"
190 ON ERR GOTO 360
195 OPEN "I",#1,F$: CLOSE #1
200 ON ERR GOTO 370
205 OPEN "D",#1,F$,1
210 FIELD #1,1 AS B$
215 GET #1
220 IF ASC(B$)<>0 THEN 370
225 GET #1
230 BC=ASC(B$)*256
235 GET #1
```

```
240 BC=BC+ASC(B$)
245 CLOSE #1
250 TM=TM-BC
255 IF TM<AD THEN 380
260 LOADM F$,TM
265 DEFUSR0=TM: ST=USR0(0)
270 IF ST=-1 THEN 295
275 IF ST=-2 THEN 315
280 PRINT "  LOADED ";
285 PRINT "MS =";BC;"BI =";BC-ST
290 TM=TM+ST: GOTO 170
295 TM=TM+BC
300 PRINT "  NOT LOADED ";
305 PRINT "[IN MEMORY]"
310 GOTO 170
315 TM=TM+BC
320 PRINT "  NOT LOADED ";
325 PRINT "[NEEDS MODULE]"
330 GOTO 170
335 '
340 'ERROR HANDLER
345 '
350 PRINT "OUT OF DATA"
355 GOTO 390
360 PRINT F$+" NOT FOUND"
365 GOTO 390
370 PRINT F$+" CORRUPT"
375 UNLOAD: GOTO 390
380 PRINT F$+" TOO BIG"
385 TM=TM+NB
390 CLEAR 200,TM
395 END
400 '
405 'NUMBER OF BYTES TO RESERVE
410 '
415 DATA 2048
```

Listing 12: SAMPLE.CFG

```
900 'SAMPLE.CFG
901 '
902 DATA FIXES
903 DATA RESET
904 DATA ERROR
905 DATA CMDS
906 DATA UTIL
907 DATA DISK
908 DATA DIR
909 DATA COPY
910 DATA *
```

Listing 13: TPL

```
100 'TPL.BAS
110 '
120 DATA BD,B3,ED,1F,02,DE,27,33
130 DATA 41,E6,A4,AE,22,11,93,74
140 DATA 22,0F,A6,80,A1,C0,26,F1
150 DATA 5A,26,F2,CC,FF,FF,7E,B4
160 DATA F4,CC,00,00,20,F8,*
170 '
180 ML$="": MN$="": FD=0
190 READ B$
200 IF B$="*" THEN 230
210 ML$=ML$+CHR$(VAL("&H"+B$))
220 GOTO 190
230 AD=VARPTR(ML$)
240 AD=PEEK(AD+2)*256+PEEK(AD+3)
250 DEFUSR0=AD
260 MN$="CMDS"
270 FD=USR0(VARPTR(MN$))
280 IF FD THEN 300
290 PRINT MN$+" NOT FOUND": END
300 PRINT MN$+" FOUND": END
```

Listing 14: UNLOAD

```
100 'UNLOAD.BAS
110 '
120 WIDTH 32
130 PRINT "UNLOAD V1": PRINT
140 AD=PEEK(&H27)*256
150 AD=AD+PEEK(&H28)+1
160 IF AD<32768 THEN 190
170 PRINT "NO RESERVED MEMORY"
180 END
190 FOR I=AD TO 32767
200 POKE I,0
210 NEXT I
220 POKE &H71,0
230 AD=PEEK(&HFFFE)*256
240 AD=AD+PEEK(&HFFFF)
250 EXEC AD
```

SUPER SUPER SPECIAL

All 530 Public Domain Programs-$145 Value
All 1,020 Subscription Programs-$260 Value
All 250 Graphic Pictures-$70 Value
ENTIRE PACKAGE Regularly $475

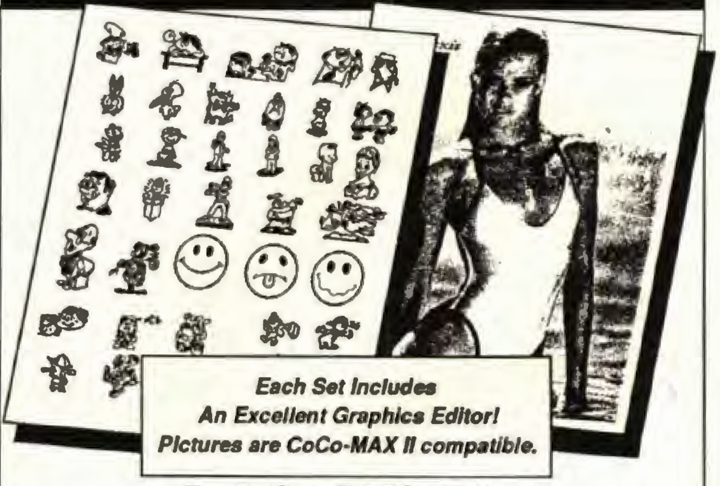
Now Only...

VISA MasterCard

$279.00!

SEE PAGE 82 FOR DETAILS

Over 250 Quality Pictures



Each Set Includes
An Excellent Graphics Editor!
Pictures are CoCo-MAX II compatible.

Each Set Of 10 Disks
Only $35.00 !
Buy 2, Get One Free!

Set #1	Set #2	Set #3
Clipart	Celebrities	Adult Only
Space Pictures	Cartoon Characters	R-Rated
Animals	Great Graphics	Beautiful Women
More	More	

T&D Subscription Software • 2490 Miles Standish Dr., Holland, MI 49423 • 616-399-9648

A guide for the assembly-language programmer on the move

OS-9 Assembly Language

by Jeff Mikel

After several years of programming with Radio Shack's *EDTASM+* assembler, I finally got OS-9 Level II and the OS-9 Level II *Development System*. This gave me access to all those things I dreamed of doing — editing files larger than 32K, editing and assembling large programs in memory, and easy disk, printer and keyboard I/O. But moving from *EDTASM+* to OS-9 was not automatic. I had a great deal to learn about OS-9 assembly-language programming. The OS-9 environment is an assembly-language programmer's dream, but with all the extra capability comes a set of design rules that must be followed if your programs are to perform properly. This is a small price to pay to reap the benefits of conditional assembly, macros, very large source files, and even local and global variables. If there is a high-level assembly language, the OS-9 Level II *Development System* has it.

Learning the new software packages was challenging since a lot of the necessary information is *hidden* in the manuals. It took a lot of manual searching and program writing before I found the information I needed.

Jeff Mikel is a recent graduate of Georgia Tech, where he studied electrical and computer engineering. He has been CoCo programming for nine years. Jeff's other interests include hi-fi audio, electronics and outdoor photography. He can be contacted at 3300 Gate Court, Rex, GA 30273. Please include an SASE when requesting a reply.

This article covers the basics you need to know, but can't seem to find. First, I cover the assembler and linker of the *Development System*. Then, I discuss how to write new OS-9 commands. Finally, we'll take a brief look at the techniques for writing subroutines for BASIC09.

Throughout this article, I assume you are familiar with the CoCo's 6809 instruction set and that you have written enough assembly-language code to be comfortable with it. I do not attempt to teach the hows and whys of assembly-language programming. Rather, I explain how to write source code for the *Relocating Macro Assembler* or *RMA*. I also assume you understand enough about OS-9 to know how and when to change directories, specify pathlists and perform the other necessary functions without being told.

If you do not have much OS-9 experience, you should first get some background with BASIC09, or some other high-level language, to familiarize yourself with OS-9 operation. However, if you have used *EDTASM+*, or other packages under Disk BASIC, and are familiar with OS-9, you are primed and ready to forge ahead to OS-9 and the *RMA* package. And, if you are already familiar with the `asm` assembler, you can probably skim this article and study only the parts that relate exclusively to the *RMA*.

Building a Working Disk

Before assembling any programs, first set up a working disk with the necessary commands and enough space to hold your source files. Do not use the original *Development System* disks — put write protect tabs on them and keep them in a safe place.

Your working disk should have, at the very least, the files `rma` and `rlink` in the `CMDS` directory. If you plan to use `scred`, an excellent text editor for assembly-language programming, make sure it is also in the `CMDS` directory. `scred` needs the file `termset` in `/dd/SYS`, so make sure that file is also available.

I recommend the `sys.l` file be on your working disk. This file can be used by the linker for resolving the system-wide symbolic names and is faster than the Level I assembler's solution of including the statement `use os9defs` in each of your source files. `sys.l` is found in the `LIB` directory on your master disk. I keep my copy of `sys.l` in the root directory to save keystrokes.

If you have 512K, you definitely want to use the RAM disk included with the *Development System* during the editing and assembly process. There are three sizes from which to choose. The 96K version is adequate for my assembly-language needs. The necessary modules for using the RAM disk are in the `MODULES` directory on your *Development System* disk.

A quick way to start using the RAM disk is to load it from your *Development System* disk each time you boot your system. The RAM disk comes in two parts: A device driver and a device descriptor. Both must be in memory for the RAM disk to work. To save memory, you should first merge the driver module with the device descriptor you want. For example, to create a 128K RAM disk, load attr if it is not already in memory. Then, put your (backup) *Development System* disk into /d0 and enter

```
chd /d0/modules
merge ram.dr r0_128k.dd >/d0/ra
m disk
attr /d0/ramdisk e pe
```

From now on, you can insert the working disk into drive /d0 and enter

```
load /d0/ramdisk
iniz /r0
```

Now device /r0 functions like one of your disk drives, except it is much faster. Although this method is easy, it removes 8K from the OS-9 system memory, which could severely limit the number of active processes or devices at one time.

The best way to add the RAM disk to your system is to create a new boot disk with the device driver and device descriptor included in the os9boot file. You can also configure the RAM disk as device /dd instead of /r0 in this way. I like this method very much. My system boots with /dd as a 96K RAM disk. After booting I use a procedure file to set up my RAM disk with the required files. The following is a sample procedure file for a single-drive system:

```
t
load makdir
chx /d0/cmds
load scred
load rma
load rlink
-x
makdir /dd/CMDS
makdir /dd/SYS
unlink makdir
copy /d0/sys/termset /dd/sys/ter
mset
copy /d0/lib/sys.l /dd/sys.l
```

With all the required files either in memory or on the RAM disk, the assembler and linker really fly.

Now that you have a working disk, it's time to explore the realm of OS-9 assembly language.

A Quick Review

Take a quick look at Listing 1, a short program that prints "Have a nice day." on the screen. Those of you who have attempted to write similar programs with *EDTASM+* will appreciate the brevity of this program. I will quickly cover its operation.

The first line, which starts with psect, gives the assembler some information about the program. (I'll discuss that in more detail later.) The program begins at Start, where Register X is set to point to the output string. Next, Register Y is loaded with the length of the string in bytes. Register A is then loaded with 1, which is the path number for the standard output device (usually the screen). The os9 I$WritLn system call writes the string to the screen and F$Exit returns control to the Shell. Since Register B was cleared earlier, the Shell assumes no errors occurred and — voila! — a successful program.

The Assembler Directives

The assembler has a set of statements called *program section directives* that direct it (hence the name) in its production of executable code. These directives are: psect, csect and vsect. Of the three, the only required directive in any program is psect. Every program *must* have a psect.

The psect, or program section directive, marks the beginning of a code section. The assembler and the linker use the information given in the psect directive in building the final OS-9 module. Proper syntax for psect is

psect *name, typelang, attrev, edition, stacksize, entry*

For Level I asm users, the psect directive takes the place of the mod directive. Each source file should have only one psect directive, but multiple psects (from multiple files) can be connected with the linker. This enables you to build a library of common routines, debug them once and use them forever.

The following is a description of each psect parameter:

name — the name for the psect. The psect name is arbitrary, but it is good practice to use a name that describes the purpose or function of the routines in the psect. Names can be as long as 20 characters. The assembler does not use this name, but the linker does when it reports errors. It is advisable to give each psect a unique name.

typelang — this specified value becomes the type/language byte for the final memory module. The four high-order bits define the module type and the four low-order bits specify the language type. Type and language codes are specified on Page 3-4 of the Technical Reference section in your OS-9 Level II manual.

attrev — this value becomes the attributes/revision byte for the final module. The four high-order bits specify the module's attributes. Currently, only Bit 7 is defined. When set, Bit 7 indicates the module is reentrant and can therefore be shared. The four low-order bits specify the module's revision number.

edition — this becomes the edition byte in the module header.

stacksize — this is an estimation as to how much memory the routines in this psect need for the stack. Be generous; remember that OS-9 uses the stack even if your program does not.

entry — the entry address of your program; the address to which OS-9 transfers control when your program is executed.

With reference to Listing 1, the program section name is first, the module's type/language byte is $11 (a 6809 object-code program module), and the attributes/revision byte is $81 (a reentrant program with Revision 1). The program is a first edition with 100 bytes reserved for stack space. The program begins at Start.

Direct your attention to the sample program header in Listing 2. This listing demonstrates the use of symbolic names in the psect directive. The only differences between this psect and the one declared in Listing 1 are the name and stack size requirements. However, the real purpose of Listing 2 is to demonstrate the other two program section directives: vsect and csect.

The vsect, or variable section directive, tells the assembler to begin a variable storage section. There are two types of vsects: Direct-page and non direct-page. If you specify a direct-page, the linker assigns the variables that are defined within the vsect to the 6809's direct page (the page referenced in conjunction with the 6809's DP register). The first vsect in Listing 2 is a direct-page vsect, since vsect dp is used. Thus, the count and path variables are located in the direct page.

Variables stored in the direct page are more quickly accessed and require less program code than those not stored in the direct page. So, store your often-used variables there. Keep in mind there are only 256 bytes in the direct page, so you may not be able to employ the direct page for all the variables you are using.

The IObuffer and filename variables are not located in the direct page. These variables should not be accessed through the same procedure to which you are accustomed with *EDTASM+* since neither you nor the assembler know the exact location of these variables at run time. Basically, don't use extended addressing. Use an index register and reference these variables as offsets from the base address.

Notice that the variables in the second vsect are large — large enough to use all the direct-page space had they been put there. It is advantageous to keep these large variables off the direct page since it is prime real estate. You are not limited to just two vsects per psect — you can have as many vsects as necessary. It is advisable, but not absolutely required, that you put all similar variables in a single vsect at the beginning of the program.

csect is the next program directive. It is similar to vsect, except it does not allocate any space. A csect, or constant section directive, is merely a convenient way of assigning successive values to a list of labels and is similar to the EQU directive. In Listing 2, fn.name has the value zero and fn.ext equals eight. You can also have more than one csect in a psect. If you specify an expression after the csect, this value becomes the base address for the section. In the second csect, first equals 10, second equals 11 and last equals 12.

Don't put your variables in a csect, even though one of the sample programs in the *RMA* manual does. Always use a vsect so the linker can properly assign values to the symbolic names. Remember that csect is used to define constants only, just like the EQU and SET directives. For example,

```
False   EQU   0
True    EQU   1
```

is the same as

```
        csect
False   RMB   1
True    RMB   1
        ends
```

With this in mind, the constants defined within a csect should normally be used with immediate addressing. For example, you can use the following source code with constants:

```
LDA    #True
CMPA   #True
```

The Four Golden Rules

When writing assembly-language programs for OS-9, keep in mind that you are writing for a multiuser, multitasking environment. This means your program does not have the degree of control you are used to with *EDTASM+*. The operating system determines where in memory your program runs and where its data storage area is. Also, the operating system must be free to interrupt your program and pause it so that the other active tasks receive their time, too. Unless your program has timing-critical code, leave the interrupts unmasked.

For all of this to work smoothly you must follow a few simple rules. First, you must write position-independent code. Your program must be able to execute from any position in the microprocessor's 64K address space. The only way to ensure this is to avoid using absolute addressing. This means that when you need to access strings, variables or constants in your program, you must specify them as an offset from the program counter (PC) register if they are in a psect or as an offset from Register U if they are in vsect and not as an absolute address. That is, instead of using either

```
ldy #name
jmp label
```

use

```
leay name,pcr
bra label
```

This is called *position independent code* and is a requirement in OS-9. The assembler takes care of the calculations for you, so the process is painless and soon becomes second nature.

Second, your program should never modify any memory outside its variable space. Since OS-9 allocates memory for each process at run time, your program might overwrite itself, or some other important data, if it is allowed contact with memory that does not belong to it. Remember, your programs are no longer running alone. Multiple copies of your program can be running at the same time. Therefore, your program must never modify itself.

Third, when your program begins, OS-9 gives it some important data located in the 6809's registers. With this data, a program knows where its variable storage is and whether or not any parameters have been passed to it. Some programs don't need to know this information, but you should make sure you save it, or use it, before loading the registers with other data.

Fourth, never directly access the CoCo's hardware. You should always use a system call to handle the desired function. This ensures your program runs on OS-9 systems other than your own. OS-9 has built-in routines for keyboard, screen, printer, and disk I/O, so it is a waste of time to write your own.

Interfacing with the Shell

Now, the $20,000 question: If you don't know in advance where your variable storage is, how do you access it? Fortunately, OS-9 provides enough data, via the registers, to provide a simple solution.

When OS-9 forks a *process* (which is what your completed program is when it executes), it puts the following data into the registers:

B — contains the number of characters in the parameter list, including the carriage return at the end.

X — points to the first byte of the parameter list.

Y — points to the top of the process' memory area, which is the end of the parameter list.

U — points to the bottom of the memory area, which is the first byte of the direct page.

DP — contains the most significant byte of the direct-page address. Never change the direct-page register.

Terminate your programs by using the F$Exit call. The value in Register B at the time of the F$Exit call is returned to the Shell as an error number. A value of zero indicates no errors. Make sure to load Register B with the proper value before calling F$Exit.

Since the direct-page register is set for you, any variables in the direct page can be accessed with direct addressing or as offsets from Register U, as long as it is not changed. Using the variables from Listing 2, you could use

```
lda path
stx count
```

Variables that are off the direct page should always be referenced as offsets from Register U to ensure your programs are using the memory space OS-9 has assigned to them. To ensure this process, use

```
leax IObufr,u
leay filename,u
```

instead of

```
ldx #IObufr
ldy #filename
```

Following the proper rules becomes automatic after a short while. Pay particular attention to the sample programs, since they demonstrate important principles.

Building An Executable Program

Program development with the *Development System* is a two-step process. First, the assembler gathers your source code into a Relocatable Object File (ROF). Then, the linker processes one or more ROFs, adds the proper module header and CRC bytes, and creates the final product. Although the two-step process for generating an executable module may seem tedious, it is this very process that makes *RMA* so powerful.

The power of this method comes from the fact that any symbolic name not defined in a psect is passed to the linker, which attempts to resolve it based on information from other psects. Thus, program code in

PCM®

The *Premier* **P**ersonal **C**omputer **M**agazine for Tandy® Computer Users

Not only does Tandy produce our favorite CoCo, we think it produces the best portable and MS-DOS computers as well. We've found that when satisfied Color Computer users decide to add portability or MS-DOS to their computing habits, many stick with Tandy. For these people we publish PCM, The *Premier* Personal Computer Magazine for Tandy Computer Users.

Each month in PCM, you'll find information and programs for the Tandy 100, 102 and 200 portable computers. And you'll find even more coverage for Tandy's MS-DOS machines — from the graphics of the 1000 to the power of the 5000.

PROGRAMS AND PROGRAM DISKS!

We learned from THE RAINBOW that readers want programs to type in, so each month we bring you an assortment of them: games, utilities, graphics, and home and business applications. For those who don't have time to type in listings, we offer a companion disk with all the programs from the magazine. Also included in PCM each month is the *Software Shopper*, an "onmail" database service from which you can order the latest shareware products from our Delphi databases for Tandy MS-DOS and PC users — even if you don't have a modem!

TUTORIALS AND PRODUCT REVIEWS!

As if all this weren't enough, we offer regular tutorials on *DeskMate*, telecommunications and hardware; assembly language, BASIC and PASCAL programming tips; and in-depth reviews of the new software, peripherals and services as they are released. Add it all up and we think you'll find PCM to be the most informative and fun magazine for this market today!

YES! Please send me a one year (12 issues) subscription to PCM for only $28.* A savings of 22% off the newsstand price.

Name ____________	In order to hold down costs, we do not bill.
Address ____________	My check in the amount of ________ is enclosed.
City ____________	Charge to my: ❑ VISA ❑ MasterCard ❑ American Express
State ____________	Acct. # ____________
ZIP ____________	Exp. Date ____ Signature ____________

To order by phone (credit card orders only) call 800-847-0309, 9 a.m. to 5 p.m. EST. For other inquiries call 502-228-4492.

* Canadian subscribers U.S. $38 plus 7% GST. Surface rate elsewhere $64, airmail $85. Allow 6 to 8 weeks for first copy. Kentucky residents add 6% sales tax. U.S. currency *only*, please.

Mail to: PCM, The Falsoft Building, P.O. Box 385, Prospect, KY 40059

one psect can use symbolic names defined in another psect.

To make this process easier to implement, the assembler gives you the ability to declare your symbolic names as either local or global names. Local names are defined only in their psect and are unknown to other psects. In fact, other psects may use the same names for different values without any "multiply defined symbol" conflicts.

Global symbols are made available to all psects, but they are referred to only if the psect has not also defined the same name. If one psect defines the symbol buffer as a global symbol and a second defines buffer as a local symbol, the symbol in the second psect is local and different from the global symbol of the same name.

To make a symbol global, include a colon (:) after the symbol name when it is defined. All other symbols are local symbols by default.

When using *RMA*, it is important to note that the assembler is case-sensitive. Thus, the labels Buffer, buffer and BUFFER are all different, even though they are spelled the same. Avoid the temptation of using the same name with different upper- and lowercase combinations in the same psect. Doing so is asking to become easily confused when debugging time comes around.

When multiple psects are linked, there must be one, and only one, mainline code segment. The mainline code segment is that section of code OS-9 transfers control to when the program begins executing. The psects in listings 1 and 2 are mainline psects. psects used with the mainline psect should be written as subroutines. They can have their own vsects, but they should not attempt to define the *typelang*, *attrev*, *edition* or *entry* bytes of the module. Make these entries in the psect statement zero or the linker refuses to link the code sections together.

Using Multiple psects

Direct your attention to the source code in listings 3, 4 and 5. These three psects, when linked together, form a program that waits for a keypress from the keyboard, dumps the values of the 6809 registers to the screen, and prints the ASCII value, in decimal, of the key pressed. Each listing is a separate file and is assembled separately from the others.

Notice that each psect has a symbol named buffer that labels a non direct-page variable. These three symbols are all local to their psects and, thus, have different values.

There are three global symbols in this group of psects: Dec, RegDump and Divide. There is also only one mainline psect — main. Notice that the other psect directives specify the *typelang*, *attrev*, *edition* and *entry* bytes as zero.

Variable names that have been declared as global symbols can be accessed from other psects simply by using that symbol name in your source code. Keep in mind that the assembler passes only unresolved references to the linker, so if the code in a psect defines a symbol with the same name as a global symbol, the global symbol is not used.

To access a global variable from another psect, use one of the following methods. If the variable is a non direct-page variable, access them in the normal manner:

```
leay IObuf,u
ldx counter,u
```

If a global direct-page variable is to be accessed, precede the variable name with the less-than symbol (<):

```
lda <Accum
stx <pointer
```

RMA is smart enough to know when to use direct addressing for direct-page variables in its own psect, but symbols defined elsewhere are assumed to be extended (16-bit) addresses. Using the < forces the assembler to generate object code that uses direct addressing. Again, you can access variables as offsets from Register U, which also works for the direct-page variables.

Armed with this information, you can build a library of common routines, debug them once, and then use them repeatedly. Each library routine can declare its own variable space, so you can "set it and forget it" since the linker automatically allocates enough space when you link files together. This gives you the ability to write "black box" subroutines that perform their function without affecting the other parts of your program.

Offset	Size	Description
0,S	2 bytes	return address — don't change it
2,S	2 bytes	number of parameters passed
4,S	2 bytes	pointer to first parameter
6,S	2 bytes	size, in bytes, of first parameter
8,S	2 bytes	pointer to second parameter
10,S	2 bytes	size, in bytes, of second parameter

Figure 1: A Sample Data Block

Type	Size	Comments
BOOLEAN	1 byte	$FF = true, 0 = false
BYTE	1 byte	
INTEGER	2 bytes	if left-most bit set, number is negative
REAL	5 bytes	8-bit exponent, 31-bit mantissa, 1 sign bit
STRING	varies	terminated with $FF if less than max size

Figure 2: The Five Basic Data Types

A Sample Assembler Session

Using the source code files in listings 3, 4 and 5 you would make an executable file by following these steps:

— Use a text editor to create the files demo.a, AsciiConv.a, and RegisterDump.a.

— Generate three ROFs, demo.r, AsciiConv.r, and RegisterDump.r, in the following manner:

```
rma -o=demo.r demo.a
rma -o=AsciiConv.r AsciiConv.a
rma -o=RegisterDump.r RegisterDu
mp.a
```

— If the file sys.l is in the current data directory, you can use the following command to invoke the linker and generate an executable file:

```
rlink -l=sys.l -o=demo demo.r As
ciiConv.r RegisterDump.r
```

— If the sys.l file is not in the current data directory, specify the complete pathlist after the -l option as follows:

```
rlink -l = /dd/lib/sys.l -O = de
mo demo.r AsciiConv.r RegisterDu
mp.r
```

Now, you should have a file called demo in your current execution directory. Notice how the file with the mainline psect is specified first in the ROF list for the linker. Other than that, it doesn't matter in what order you specify the modules, as long as the mainline section is first.

A quick word on the sys.l file. This file is a library file and is an ROF generated from the file defs/os9defs.a. You can insert the use /dd/defs/os9defs.a directive in each of your source code files and have the assembler resolve the OS-9 symbol constants, which slows the assembly process, or you can have the linker resolve all of the symbols in one step during the linking process. The second method is faster; the choice is yours.

OS-9 Level II

Listing 1: first.a

```
* listing 1
          psect     first,$11,$81,1,100,Start

Start     leax      name,pcr  get prompt
          ldy       #name.len and its length
          lda       #1        get standard output path number
          clrb
          os9       I$WritLn  output prompt
          os9       F$Exit    return to shell

name      fcc       /Have a nice day./
          fcb       13
name.len  equ       *-name
```

Listing 2: second.a

```
* listing 2
* sample vsects and csects - not executable

Type      equ       $10
Lang      equ       $01
Att       equ       $80
Rev       equ       $01
Edtn      equ       $1
Stack     equ       200

          psect     second,Type+Lang,Att+Rev,Edtn,Stack,Entry

          vsect     dp
count     rmb       2
path      rmb       1
          endsect

          vsect
IObuffer  rmb       256
filename  rmb       11
          endsect

          csect
fn.name   rmb       8
fn.ext    rmb       3
          endsect

          csect 10
first     rmb       1
second    rmb       1
last      rmb       1
          endsect

Entry     code
          etc.
          endsect
```

Listing 3: demo.a

```
* listing 3
          psect     main,$11,$81,1,100,Entry

          vsect
buffer    rmb       2
          endsect

Entry     lbsr      RegDump   dump registers to screen
```

To execute the program, type demo at the OS9: prompt. To see how the hardware registers change, try adding some parameters after demo on the command line. For example, try demo abcd, demo #16K and demo abcd #16K.

Interfacing with BASIC09

The rules for writing machine-language programs for BASIC09 are somewhat different. These programs execute as subroutines, not as individual processes, so your program development is different in several ways. First, the *typelang* byte in the psect directive must be set to $21 instead of $11. Second, BASIC09 does not allocate memory for the subroutine, so your subroutine has severe restrictions on memory usage for its variables. All communications between your subroutine and your BASIC09 program are done via the hardware stack (the S register). Finally, and of great importance, your program must end with an rts instruction, not the F$Exit call. Using F$Exit terminates BASIC09 as well. Your programs must still be written in position-independent code. Programs not written in position-independent code do not work with BASIC09.

When BASIC09 calls your machine-language subroutine, Register S points to a data block your subroutine can access. A sample data block might appear as shown in Figure 1.

The size of the data block varies depending on the number of parameters sent. If no parameters are sent, the data block contains

((((((((NEW!))))))))

From MillSoft Co.

MillSofts' Baseball Card Catalog

by Paul Miller

NOW YOU CAN PUT ALL YOUR Baseball Cards on disk! Requires COCO3 with two Diskdrives and 128k. It is also available for 1 and 2 with tape drives.

To order, please send a check or money order to:
MillSoft Co. • Box 2377
Amagansett, NY 11930

$18.95. Please specify which System you have. NY Res., Please add 7.5% Tax.

just the return address and the parameter count, which would be zero in this case. Each parameter is defined by a pointer to the area used to store the parameter and a count of how many bytes are required for storage. Therefore, each takes up four bytes on the stack. Any BASIC09 data type can be sent as a parameter to your subroutine. The five basic data types, and their sizes, are shown in Figure 2.

The size of a string variable will always be the number of bytes specified in the `DIM` statement, not the actual length of the string stored in that section of memory. If the length of the string is shorter than its dimensioned size, which is 32 bytes unless specified otherwise, the string is terminated with $FF (255 decimal). You must find the length manually by searching for this character. Also, make a note not to store the parameter values directly on the stack — rather they are accessed indirectly. See the sample programs for more clarification.

At the completion of your subroutine, you may return an error code to BASIC09 by loading Register B with the desired error number and setting the carry flag. If the carry flag is not set, BASIC09 assumes no errors occurred, regardless of the contents of Register B.

Your subroutine knows only the size and location of a parameter passed to it. There is no type checking, so your subroutine is not able to discriminate between variables of different types that are the same size. For example, assume the following statements appear in a BASIC09 program:

```
DIM name:STRING[5]
DIM mass:REAL
DIM state(5):BOOLEAN
```

Also, assume you have created a machine-language subroutine called `Sample`, which can be executed from a BASIC09 program through the following command:

```
run Sample(parameter)
```

You could execute `Sample` in any of these three ways:

```
run Sample(name)
run Sample(mass)
run Sample(state)
```

In each of the above cases, the subroutine`Sample` will not know which of the three BASIC09 variables was sent. Naturally, `Sample` expects a certain variable type — `STRING`, `REAL` or `BOOLEAN` — but it does not *know* whether the parameter sent to it is of the proper type. You needn't worry too much about this situation since the contents of a variable of the wrong type will, in all

```
          leax      prompt,pcr
          ldy       #prompt.l
          lda       #1
          os9       I$Write
          leax      buffer,u  get input buffer address
          ldy       #1
          lda       #1
          os9       I$Read    read from std in
          ldb       ,x        get character read
          clra
          lbsr      Dec       convert character to decimal ascii
          lda       #'-       insert '-' at start of string
          sta       ,-x
          leay      1,y       add one to string length
          lda       #1
          os9       I$WritLn  write ascii string to screen
          clrb                tell shell no errors
          os9       F$Exit
prompt    fcc       /Press a key: /
prompt.l  equ       *-prompt
          endsect
```

Listing 3: `demo.a`

```
* listing 3

          psect     main,$11,$81,1,100,Entry

          vsect
buffer    rmb       2
          endsect

Entry     lbsr      RegDump   dump registers to screen
          leax      prompt,pcr
          ldy       #prompt.l
          lda       #1
          os9       I$Write
          leax      buffer,u  get input buffer address
          ldy       #1
          lda       #1
          os9       I$Read    read from std in
          ldb       ,x        get character read
          clra
          lbsr      Dec       convert character to decimal ascii
          lda       #'-       insert '-' at start of string
          sta       ,-x
          leay      1,y       add one to string length
          lda       #1
          os9       I$WritLn  write ascii string to screen
          clrb                tell shell no errors
          os9       F$Exit
prompt    fcc       /Press a key: /
prompt.l  equ       *-prompt
          endsect
```

Listing 4: `AsciiConv.a`

```
* listing 4

* Function
*   subroutine Dec converts a 16-bit, unsigned binary number to a 5-digit
*   ASCII string terminated with a linefeed character
*
* Usage
*   call from the mainline code by
*
*             bsr   Dec
*
* Entry
*   D holds the 16-bit number
*   U points to dp base address
*
* Exit
*   X points to the first byte of the ASCII string
*   Y contains the ASCII string length

          psect     AsciiConv,0,0,0,32,0

          vsect     dp
Accum     rmb       2
          endsect

          vsect
buffer    rmb       6
          endsect
```

```
Dec:     pshs     u
         leax     buffer+5,u move to assumed end of buffer
         ldy      #1
         leau     Accum,u   get address of accumulator
         std      ,u        save number in accumulator
         lda      #13
         sta      ,x        mark buffer end
DecLoop  ldb      #10       load divisor
         bsr      Divide    do division
         adda     #'0       convert remainder to ASCII
         sta      ,-x       save in buffer
         leay     1,y
         ldd      ,u        all done?
         bne      DecLoop   branch if not
         puls     u,pc

* following subroutine divides an unsigned 16-bit number by a positive 8-bit
*  entry: B - divisor
*         U points to 16-bit number
*   exit: A - remainder
*         U points to result

Divide:  pshs     b,x
         ldb      ,u        get first byte
         clra               clear remainder
         bsr      DoDiv     divide
         stb      ,u        store result
         ldb      1,u       get second byte
         bsr      DoDiv     divide
         stb      1,u       store result
         puls     b,x,pc
DoDiv    ldx      #8        loop counter = 8
DivLoop  aslb     rotate    D left
         rola
         cmpa     2,s       is A larger than divisor?
         blo      TooBig    branch if not
         incb
         suba     2,s
TooBig   leax     -1,x      done all 8 repetitions?
         bne      DivLoop   branch if not
         rts
         endsect
```

likelihood, look like garbage. Just keep in mind there is no way to explicitly check a variable's type.

As a matter of fact, the same is true for BASIC09 procedures. If a calling procedure sends as its argument a `REAL` (floating-point) number, the procedure called can declare its parameter as a `REAL`, a five-element array of type `BYTE`, or a five-element array of type `BOOLEAN` (try it!). No errors are generated because there is no type checking. Any problems that arise are as a result of the way the called procedure interprets the data sent to it. Again, this is not something you have to worry about. You just need to know you cannot check a parameter's type. Rather, just check for the expected size and for valid data and all will be fine.

Program Memory Under BASIC09

As stated earlier, your subroutine will not have any memory reserved for it. Very few programs can keep track of all the relevant data in the CPU registers, so you must have some memory somewhere. Where do you get it? There are two ways to solve this problem.

First, your program can reserve some extra space on the hardware stack. For example, if your program needs 10 bytes of

Burke & Burke
P.O. Box 733
Maple Valley, WA 98038

U.S. ORDER DESK: (800) 237-2409

INT'L & TECHNICAL: (206) 432-1814

OS9 Software (* >= 256K; ** >= 512K required):

WORLD CLASS CHESS* -- Use your Tandy Cyrus Chess cartridge under OS9 L2	$29.95
FILE SYSTEM REPACK 1.1 -- Popular disk defragmenter. NEW VERSION!	$29.95
FILE RECOVERY SYSTEM -- helps rebuild crashed disks	$24.95
R. S. B.* -- Real Disk BASIC for Level 2 (BASIC ROM required).	$39.95
EZGEN 1.09 -- OS9 bootfile editor	$19.95
CYBERVOICE (Software only)* -- Run your S/S SuperVoice w/ OS9!	$24.95
PERTASCII* -- Multi-user scrambled letter word game.	$19.95
WILD & MV -- Wildcard & move directory entry utilities.	$19.95

BASIC Software (* >= 256K; ** >= 512K required):

HYPER-I/O -- 16K Hard disk / big floppy BASIC. Specify B&B or DISTO hard disk	$29.95
DAGGORPATCH -- Disk patch for Dungeons of Daggorath.	$ 9.95

HYPER-I/O comes on floppy disk & is EPROMable. HYPER-I/O allows large floppy and hard disk directories, with good M/L compatibility. We recommend HYPER-I/O for BBS & BASIC use, but it may be incompatible with some of your commercial ML S/W.

VISA MasterCard

Affordable Color Computer Hardware:

CYBERVOICE -- Speech Synthesizer w/ OS9 software.	$99.95
COCO XT -- Adapts PC hard drives to CoCo. OS9 S/W included.	$69.95
COCO XT-RTC -- CoCo XT, with battery backed real-time clock.	$99.95
XT-ROM -- Boots OS9 from B&B hard disk automatically.	$19.95
4' Hard disk cable set (Extra long -- standard cables are 2' or shorter)	$17.50

News About Upcoming Products...

A few of the products you can expect to see over the next few months are the OSKenLook user interface for OS9/OSK; the PIXELBOX adapter that lets you use LogiTech ScanMan handheld scanners with the CoCo 3, and OSK ports of several popular Burke & Burke utility packages -- Chris

New Ego Killer!

World Class Chess lets you use your Tandy Cyrus Chess cartridge under OS9 Level 2. We've added mouse, disk, and windows support. Looks great, and plays a mean game of chess! Only $29.95.

(Cyrus Chess Cartridge available from TANDY Consumer Mail, P/N 26-3064, $29.95)

WA RESIDENTS ADD 8.2% SALES TAX. U.S. COD's add $3.75. Min. U.S. shipping $4.00. Min. to Canada $5.00. Please allow 2 weeks for delivery. Overnight or 2nd- day available for in-stock items. Software upgrades $5.00 each w/receipt, including U.S. shipping.

variable storage space, you can use the instruction

```
leas -10,s
```

at the beginning of the program. Then, at the end, you'll need to "bump" the stack pointer back to its original position. Exit the subroutine using

```
leas 10,s
rts
```

This method is simple, but it does not allow your subroutine to have any static variables, which means all variables are erased when your subroutine ends. They do not exist except when your subroutine is running and they cannot be saved for later use by the subroutine.

If you need static variables, meaning that they don't "go away", you can use a BASIC09 variable as storage. For example, the 10 bytes of storage space used above could also be stored in an array declared by the BASIC09 program:

```
DIM memory(10):BYTE
```

Then, so that the machine-language subroutine has access to the space occupied by `memory`, use

```
RUN Sample(memory)
```

Subroutine `Sample` can then use the 10 bytes of Variable `memory` for its variables. Keep in mind that variables in BASIC09 are local to the procedure that defines them. Therefore, unless explicitly passed from one procedure to the next, any *static* storage space is also local. Also, the *static* storage space exists only while the BASIC09 procedure that calls your machine-language subroutine is running, which makes it dynamic storage just like all other BASIC09 variables. This is also why the word static is italicized. In the true sense of the word, BASIC09 does not offer any static variables, so be careful when using this technique.

It is important to note that when writing BASIC09 subroutines, your program is not the primary module, thus it should not attempt to adjust the amount of memory set aside for the process. BASIC09 is in charge. You must work with it, not around it.

Listing 6 is a simple subroutine that takes three parameters, all integers. The first two integers are added together and returned via the third parameter. This program is fairly straightforward and is intended to illustrate how to access and change the parameters. Notice the use of a `csect` to keep track of all of the offsets for accessing the parameters. This way, the offsets are

Listing 5: `RegisterDump.a`

```
* listing 5

*
* OS-9 Register Dump
* Written 89/07/06 by Jeff Mikel
*
* Call using a BSR to RegDump
*
*  Entry: U points to dp base address
*
*   Exit: register dump displayed on terminal
*         all registers are preserved

          csect     0
regs.cc   rmb       1
regs.a    rmb       1
regs.b    rmb       1
regs.dp   rmb       1
regs.x    rmb       2
regs.y    rmb       2
regs.u    rmb       2
return    rmb       2
          endsect

          psect     RegisterDump,0,0,0,64,0

          vsect
buffer    rmb       80
          endsect

RegDump:  pshs      cc,a,b,dp,x,y,u save registers on stack
          leax      buffer,u  get output buffer address
          leay      data,pcr  get prompt address
          bsr       copy      copy first reg name to buffer
          lda       regs.cc,s get CC register
          bsr       hexout    convert
          lda       regs.a,s  get A
          bsr       hexout
          lda       regs.b,s  get B
          bsr       hexout
          lda       regs.dp,s get DP
          bsr       hexout
          ldd       regs.x,s  get X
          bsr       double
          ldd       regs.y,s  get Y
          bsr       double
          ldd       regs.u,s  get U
          bsr       double
          tfr       s,d       get S
          addd      #return+2 compute S before subroutine call
          bsr       double
          ldd       return,s  get return address
          bsr       double
exit      lda       #$0d      flag end of buffer
          sta       ,x
          leax      buffer,u  get buffer start
          ldy       #80       get max output characters
          lda       #1
          os9       I$WritLn  display on terminal
          puls      cc,a,b,dp,x,y,u,pc restore registers and return

double    bsr       HexConv   convert first byte
          tfr       b,a       get LSB
hexout    bsr       HexConv   convert second byte
          lda       #$20      insert a space in buffer
          sta       ,x+
copy      lda       ,y+       copy next prompt
          beq       retn
          sta       ,x+
          bra       copy
retn      rts

HexConv   pshs      a
          rept      4         convert MS nibble to ascii
          lsra
          endr
          bsr       Digit
          lda       ,s        get byte again
          anda      #$0f      convert LS nibble to ascii
          bsr       Digit
          puls      a,pc

Digit     cmpa      #9        in range 0-9?
          bls       ToAscii
          adda      #7        digit is A-F
ToAscii   adda      #$30      make it ascii
          sta       ,x+       save in buffer
          rts

data      fcc       /cc=/
```

```
         fcb      0
         fcc      /a=/
         fcb      0
         fcc      /b=/
         fcb      0
         fcc      /dp=/
         fcb      0
         fcc      /x=/
         fcb      0
         fcc      /y=/
         fcb      0
         fcc      /u=/
         fcb      0
         fcc      /s=/
         fcb      0
         fcc      /pc=/
         fcb      0
         fcb      0
         endsect
```

Listing 6: `AddInteger.a`

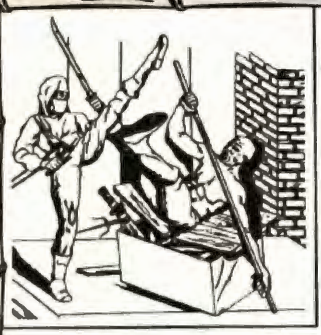
```
         psect    AddInteger,$21,$81,1,100,Start

         csect
return   rmb      2         Basic09 return address
pcount   rmb      2         number of parameters passed
param1   rmb      2         pointer to first parameter
psize1   rmb      2         size of first parameter
param2   rmb      2         pointer to second parameter
psize2   rmb      2         size of second parameter
param3   rmb      2         pointer to third parameter
psize3   rmb      2         size of third parameter
         endsect

Start    ldb      #100      prepare for possible error
         ldx      #3        exactly three parameters passed?
         cmpx     pcount,s
         bne      error     branch if not
         ldx      #2        each parameter size of type integer?
         cmpx     psize1,s  check first parameter
         bne      error     branch if wrong size
         cmpx     psize2,s  check second parameter
         bne      error     branch if wrong size
         cmpx     psize3,s  check third parameter
         bne      error     branch if wrong size
         ldd      [param1,s] get value of first parameter
         addd     [param2,s] add to value of second parameter
         std      [param3,s] store as value of third parameter
         clrb               signal no errors
         rts
error    orcc     #1        set error flag
         rts
         endsect
```

automatically computed, which saves you time, trouble and prevents mistakes due to incorrect offsets.

Executable programs for BASIC09 are created in the same manner: assemble and link them. BASIC09 automatically loads them when they are called, provided the subroutines are in the current CMDS directory. For example, the instructions to make an executable module out of Listing 6 are as follows:

— Create the file `AddInteger.a` with a text editor.
— Assemble `AddInteger.a` to a ROF, `AddInteger.r` using

```
   rma -o=AddInteger.r AddInteger.a
```



Kyum-Gai: to be Ninja (OS-9 Version) is the culmination of a project started almost a year ago. The talents of *Glen R. Dahlgren* (RS-DOS game writer for Sundog Systems), *Kevin Darling* (a legend for his work in OS-9), and *Eddie Kuns* (author of KBCom) have been pooled to create a masterpiece of game software under the OS-9 operating system. Fast martial arts action with outstanding graphics, great digitized sound effects, and incredible animation are featured in this arcade game, all in the OS-9 environment.

Always wanted to play the great CoCo 3 games but didn't want to sacrifice your OS-9 features? ***Multitask while playing*** Kyum-Gai. ***Have multiple Kyum-Gai's*** running in memory. Don't worry about switching windows, because Kyum-Gai: OS-9 auto-pauses, to wait for your return.

Put simply, this is an unprecedented piece of software for the CoCo; a landmark game sure to be a major part of the Color Computer history. Don't miss out on this game!
Req. 512K CoCo III with OS-9 Level 2 and joystick. **$29.95**

VISA, Mastercard, Money Order, and COD (USA only, please) accepted. All foreign orders must be sent in US currency Money Orders. Include $2.50 for shipping in USA and Canada, $5.00 Foreign. $3.00 extra for COD orders. VA residents please add 6% sales tax. Dealer inquiries welcome. Authors: we're looking for new software!

Sundog Systems
P.O. Box 766
Manassas, VA 22111
(703) 330-8989

OS9/68000
JUST THE FACTS!

The PT68K4 SINGLE BOARD COMPUTER - As seen at the ATLANTA COCOFEST - is available for IMMEDIATE SHIPMENT in Assembled or Kit Form!

Upgrade to a powerful 68000 microprocessor on an XT-size system board with 4 serial/2 parallel ports and up to 12MB of memory. The K4 is IBM PC/XT hardware compatible - and also supports VGA, SVGA plus high density floppy drives!

16MHZ PT68K4 Assembled Board with 1MB DRAM - $499.00

BARE BONES KIT: System Board, 1MB RAM, Cabinet, PS, High Density Floppy, OS9 - $849.00

Professional OS9 Operating System with C COMPILER $299.00

See the DELMAR AD for systems information!

PERIPHERAL
TECHNOLOGY
1480 TERRELL MILL RD. #870
MARIETTA, GA 30067
404/984-0742
OS9 is a trademark of Microware

— Use the linker to generate an executable module using

```
rlink -o=IntAdd AddInteger.r
```

— From BASIC09, call the subroutine in the same manner as you would any BASIC09 subroutine:

```
run IntAdd(x1, x2, x3)
```

where x1, x2 and x3 are all integers. Run Listing 8 for a real-life sample of this routine in operation.

Listing 7 is another simple subroutine. It gives a hexadecimal dump of the contents of a single parameter. Use this subroutine to examine how different data types look in the computer's memory. This program uses the stack for all of its variable storage. Notice again the use of a `csect` to keep all of the offsets straight. Also notice the stack pointer is restored to its original value before that final `rts` instruction. This is a must, but it is easy to forget. Assemble and link `ParamDump.a` (Listing 7) in the same manner as you did for Listing 6 and then run the program in Listing 9 to see the program in action.

Be careful when writing programs for BASIC09. Remember, do not attempt to write to memory that has not been given to you by a BASIC09 procedure. Be especially careful if you are rewriting a program for BASIC09 that ran under the Shell. Remember you do not have access to the direct page — that belongs to BASIC09. Careful planning prevents any mishaps.

Hopefully, you will find that assembly-language programming under OS-9 is enjoyable, especially since you do not have to worry about writing disk and keyboard drivers or Hi-Res graphics routines anymore. Plus, if you follow the rules, your programs automatically work with all other OS-9 programs. This is all made possible by OS-9's standard file structure and graphics routines. I believe once you start programming in assembly language under OS-9, you won't want to go back to Disk BASIC.

I would like to think this article has answered all your OS-9 assembly-language questions, but I know it hasn't. Please feel free to send me any questions or comments. Please include an SASE for a reply. ❑

Listing 7: `ParamDump.a`

```
         psect    ParamDump,$21,$81,1,100,Start

         csect
buffer   rmb      3         i/o buffer
return   rmb      2         Basic09 return address
pcount   rmb      2         number of parameters passed
param1   rmb      2         pointer to first parameter
psize1   rmb      2         size of first parameter
         endsect

Start    leas     -return,s set up variable space on stack
         ldb      #100      prepare for possible error
         ldx      pcount,s  get parameter count
         cmpx     #1        exactly one?
         bne      error     exit with error if not
         leax     buffer,s  get i/o buffer address
         ldb      #32       put space character at end
         stb      2,x
         ldy      psize1,s  get parameter size
         ldu      param1,s  get parameter location
loop     pshs     y
         lda      ,u+       get next byte of parameter
         pshs     x
         bsr      ConvHex   convert to Ascii hex
         puls     x
         lda      #1        standard output path
         ldy      #3        number of characters to write
         os9      I$Write
         puls     y
         leay     -1,y      done entire parameter?
         bne      loop      repeat if not
         ldb      #13       else send carriage return to output path
         stb      ,x
         ldy      #1
         os9      I$WritLn
exit     leas     return,s  restore stack
         rts
error    leas     return,s  restore stack
         orcc     #1        indicate error occurred
         rts

ConvHex  pshs     a
         rept     4         get four high bits
         lsra
         endr
         bsr      Digit     convert to Ascii digit
         puls     a
         anda     #$0F      get four low bits
Digit    adda     #'0       make Ascii number digit
         cmpa     #'9       outside range 0-9?
         bls      legal     branch if not
         adda     #7        else make in range A-F
legal    sta      ,x+       save in i/o buffer
         rts
         endsect
```

Listing 8: `Tester.b09`

```
PROCEDURE Tester
 0000     (* listing 8
 000C     (* illustrates how to call the machine-language subroutine
 0046     (* IntAdd that is made from listing six
 006D     DIM first,second,sum:INTEGER
 007C     first=1991
 0084     second=2001
 008C     RUN IntAdd(first,second,sum)
 00A0     PRINT first; " + "; second; " = "; sum
```

Listing 9: `ParamTest.b09`

```
PROCEDURE ParamTest                   0060
 0000     DIM f:REAL                  0061     PRINT "REAL    = ";
 0007     DIM i:INTEGER               0070     RUN ParamDump(f)
 000E     DIM b:BYTE                  007A     PRINT "INTEGER = ";
 0015     DIM bo:BOOLEAN              0089     RUN ParamDump(i)
 001C     DIM s:STRING[20]            0093     PRINT "BYTE    = ";
 0028                                 00A2     RUN ParamDump(b)
 0029     f=123.45                    00AC     PRINT "BOOLEAN = ";
 0034     i=12345                     00BB     RUN ParamDump(bo)
 003C     b=123                       00C5     PRINT "STRING  = ";
 0043     bo=TRUE                     00D4     RUN ParamDump(s)
 0049     s="This is a string"        00DE     END
```

Ultralace the Second Loop

by H. Allen Curtis

Last month I presented the programs that generate *Ultralace*'s Font menu and two font sets. This month in Part II we'll take a look at programs that create the Design menu and screen dumps for Tandy DMP-series and Epson-compatible printers.

Let's cover the Design menu creator first. When you have finished entering GENDMENU (Listing 1), save it on the Program Disk (created last month) and run it. As it executes, you see the Design menu form on the screen. When the menu is completed, you will see that it is arranged in four rows of designs. More precisely, each row consists of slots containing designs. The sizes of the slots in rows 0, 1, 2 and 3 are 16 dots by 16 dots, 32 dots by 32 dots, 48 dots by

H. Allen Curtis lives in Williamsburg, Virginia. He is interested in 17th and 18th century history and enjoys biking through the colonial capital. He balances the past and present in his computer work. He can be contacted by writing 172 Dennis Drive, Williamsburg, VA 23185 or by calling (804) 229-7086.

48 dots, and 48 dots by 48 dots, respectively. The slots of Row 0 are labeled from a through z and A through L. The slots of Row 1 are labeled from a through s. The slots of Rows 2 and 3 are labeled from a through j. There is a 96-dot-by-96-dot slot at the right end of Rows 2 and 3. This slot stores the currently chosen design. Some designs use more than one slot. For example, CoCo Cat employs slots 2e, 2f, 3e and 3f, and is contained in a 96-dot-by-96-dot area.

Figure 1 serves as a means of checking your screen for drawing errors. (Figure 1 also illustrates the results of an *Ultralace* screen dump — the one that prints four screens in a single column. Each of the screens dumped in the design menu show a different chosen design. We'll cover how to use the screen dumps a little later in this series.) Compare your screen with any of the four screens shown in Figure 1. I should point out that screen designs are always proportionally much more narrow than their printed counterparts. Designs can also be their normal size, twice their normal width, or twice their normal width and length. Furthermore, mirror images of the designs can be selected. Some of these options are illustrated in the chosen-design slots of the four screens in Figure 1.

When comparing the design menu screen with the screens in Figure 1, make a note of any slot with erroneously drawn designs. Then press BREAK and answer the next prompt by pressing N. Check each line of GENDMENU — the REM statements at the ends of the lines identify the design positions for the slots. After you have corrected any mistyped lines, save GENDMENU again. Run the program and repeat the screen comparison. If you still find errors, repeat the correction procedure. Otherwise, press BREAK, insert the File Disk in the drive, and press Y. The files DMENU.HR1 and DMENU.HR2 are saved to the File Disk.

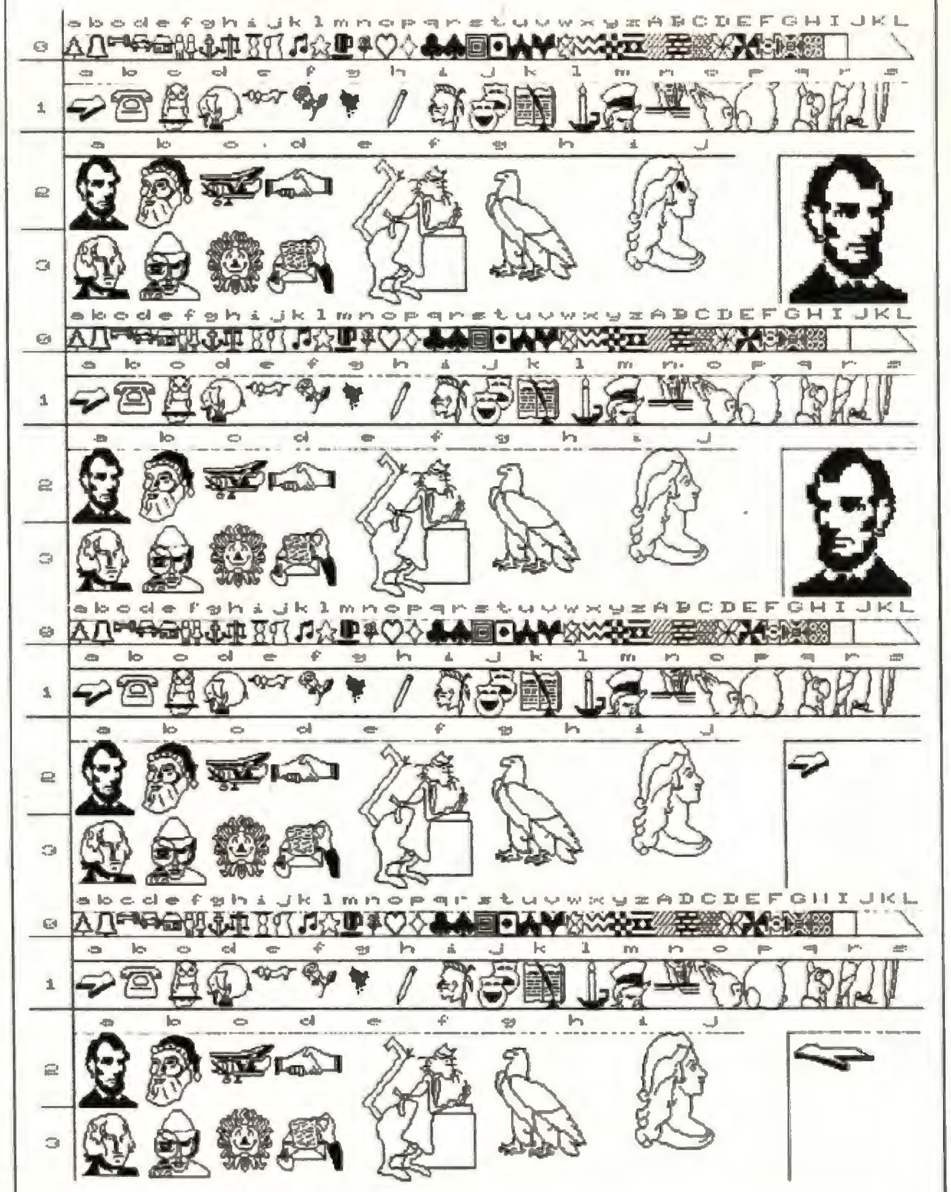

a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L
0
1
2
3

Figure 1

Personalized Messages

Listing 2, GENSTR, generates an ASCII file containing six strings. Three of the strings are phrases or short sentences often used in documents. Each of these strings can be retrieved from the file and printed in the font shown on the screen. The other three strings can be retrieved to draw complete horizontal and vertical borders composed of multiple images of the current design selection.

When you have entered the GENSTR listing, save it on the Program Disk. Run it, and follow the onscreen instructions. As you follow these instructions, you will be formulating strings of phrases and sentences tailor-made just for you. The program ends after saving the ASCII file, STR.DAT, on your File Disk.

Preparing for Output

GENMLT and GENMLE are the programs in Listings 3 and 4, respectively. You should need to enter only one of the programs, depending on the type of printer you use. If you use an older Tandy DMP-series printer that prints in a graphics mode with 800 or more dots across a page, enter GENMLT. Some Tandy DMP-series printers, such as the DMP-130 and 130A, will not print more than 480 dots across a page unless they are connected to a CoCo with a serial/parallel interface and are set for IBM printer emulation. Those of you with these DMP-series printers and interconnections should enter GENMLE. Similarly, those of you with Epson/IBM and compatible printers should enter GENMLE. Many of the newer Tandy printers fall into this category.

Both GENMLT and GENMLE generate several machine-language routines used by *Ultralace* as screen dumps for older Tandy DMP-series printers, and Epson/IBM and compatible printers, respectively. The ML routines generated by both GENMLT and GENMLE allow you to select designs in normal size, twice normal width, twice normal width and length, and regular or mirror image. To augment the 3-column screen dump, GENMLT and GENMLE generate routines for saving and loading the left half of an *Ultralace* screen. All of these routines are written in machine language for optimum execution speed and memory usage.

Control Codes for Epsons

Those of you who enter GENMLE may have to change some of the DATA statements. Specifically, lines 800, 810, 820, 900, 910 and 920 contain printer control codes needed to execute the screen dumps. The first value in each of these lines specifies the number of values in the control code sequence given in the DATA statement.

Lines 800 and 820 specify control-code sequences for printer line spacing. For normal printing, your printer provides a line spacing of 1/6-inch with six lines of print per inch. For the graphics modes, the line spacing must be closer — 7/72-inch. For most Epson/IBM and compatible printers, the control-code sequence to produce a 7/72 line spacing is 27, 49. (Check your printer manual to see whether or not 27, 49 is the control-code sequence for this line spacing.) If your printer uses a different sequence, modify Line 800 accordingly. For example, if the control-code sequence has three values, you will have to alter the initial value to 3 in addition to changing the control-code sequence values. In such a case, the DATA statement ends with six 0 values instead of seven.

Line 820 contains the control-code sequence for the return to 1/6-inch line spacing. If your printer requires a sequence other than 27, 50 for this spacing, make changes in Line 820 in a similar fashion to those made in Line 800.

Lines 900 and 920 contain two control-code sequences each. In Line 900, the first sequence is a line-spacing code sequence as in Line 800. The second control sequence in Line 900 is for setting the left margin of the page to seven spaces. The usual code sequence for such a margin setting is 27, 108, 7. Check your printer manual for the control code sequence needed for a left margin setting of seven spaces. If your printer uses a different sequence for the 7/72-inch line spacing or for the left margin setting of seven spaces, make appropriate changes to Line 900. Also, change the first value in Line 900 to correspond to the total number of values used by the two control-code sequences. Every DATA statement must contain exactly 10 values. Make sure that concluding values not used in the control-code sequences are set to zero.

Line 920 contains control-code sequences for 1/6-inch line spacing and a left margin setting of zero. If necessary, alter Line 920 in a similar way to Line 900.

Lines 810 and 910 of GENMLE contain control-code sequences for specifying graphics modes. The graphics mode specified in Line 810 provides a density of 960 dots per line (120 dpi). The codes 27, 42, 1 select the mode while 192 and 3 specify that 192 + 3*256, or 960 dots are to be printed on each line.

The codes in Line 910 select a dot density of 720 dots per line (90 dpi). The codes 27, 42, 6 select the mode, while 128 and 2 indicate that 128 + 2*256, or 640 dots of the 720 are to be printed per line. The left margin setting given in Line 900 was specified to center the 640 dots on the 720-dot line. If your printer uses a different control-code sequence for these graphics modes, change lines 810 and 910 accordingly.

For Both Printer Drivers

In both GENMLT and GENMLE, sets of strings of hexadecimal characters are used to generate the machine-language routines. Since it is not easy to enter these strings with complete accuracy, GENMLT and GENMLE have built-in error checking. When run, each of these programs reports the line numbers of any lines in which string-typing errors are encountered.

Neither GENMLT nor GENMLE should be run before being saved to disk. When you have completed entering either of these programs, save it on the Program Disk. Then run it. If the program reports an error, compare your listing with the accurate version printed here and correct the error. Run the program and, if needed, continue the error-correction process. When all errors have been corrected and the program runs properly, the program advises you as to the routines being generated. Finally you are prompted to insert the File Disk in your drive. The file MLR.BIN is then saved on that disk.

At this point, you have produced two disks. Your Program Disk should have six BASIC programs on it: GENFMENU, GENFONTI, GENFONTN, GENDMENU, GENSTR, and either GENMLT or GENMLE. Your File Disk should contain seven files: FMENU.HR1, FONTI.DAT, FONTN.DAT, DMENU.HR1, DMENU.HR2, STR.DAT and MLR.BIN.

If you have any DESKTOPH font files, make a backup copy of the File Disk and copy the DESKTOPH font files, with their appropriate *Ultralace* filenames as discussed last month, to the backup disk.

RAINBOW ON TAPE Users

While *Ultralace* requires a disk drive, the programs printed in THE RAINBOW that create the *Ultralace* package are included on RAINBOW ON TAPE. You must load these files and save them to the Program Disk before using them.

PRICES SLASHED ON ALL
ADOS PRODUCTS
LIMITED TIME ONLY!

EXTENDED ADOS-3 .... $29.95
(Requires ADOS-3)
Disto real-time clock driver, $5.
Adapter for controllers lacking 28-pin ROM socket, $10.

ADOS-3 .... $24.95

EXT. ADOS-3 PLUS ADOS-3 .... $49.95

SMARTWATCH REAL-TIME CLOCK $29.95
Usable in controllers with 28-pin ROM socket, or in ROM pack, $10. Includes OS-9 Level II driver. Ext. ADOS-3 driver, $5 with clock, $10 separately.

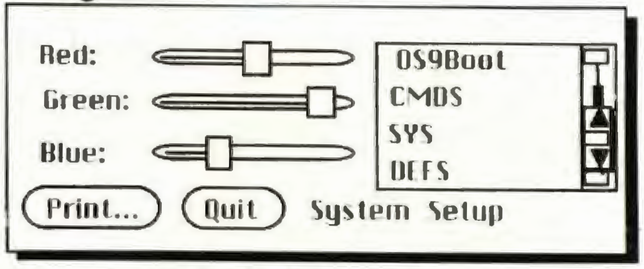
ADOS FOR CoCo 1 and 2 .... $14.95

—All ADOS software is sold on disk with configuring utilities that generate an EPROMable binary file. Information is provided for having an EPROM burned by mail for $15.
— See our earlier ads and reviews in July 1987 and October 1989 Rainbows (available on request) for product descriptions.

SPECTROSYSTEMS
11111 N. Kendall Dr.
Suite A108
Miami, FL 33176
(305) 274-3899

PLEASE ADD $2 SHIPPING • NO DELAY ON PERSONAL CHECKS
WE CANNOT ACCEPT CREDIT CARDS

Burke & Burke
P.O. Box 733
Maple Valley, WA 98038



OSKenLook: The Next Interface...

Next Time

Now that we have completed the auxiliary files, next month we'll look at one version of the main program. As with the printer drivers, depending on the type of printer you use, you need enter only one of the two versions of the main *Ultralace* program.

In the meantime, three font-file disks are available from me at the address given above: Fonts T through Y ($5), Fonts J through Y ($12), and Fonts A through Y ($19). Please include payment to me by check or money order. ❑

CoCo 3 Disk

70	2	580	148	890	198
120	131	610	145	920	129
190	8	640	167	950	231
240	36	680	142	980	156
290	35	720	190	1010	152
340	128	750	123	1040	191
400	93	790	145	1080	13
450	156	820	13	END	71
500	103	860	108		

Listing 1: GENDMENU

```
10 GOTO60
20 CLS:LOCATE4,10:PRINT"SAVING DMENU/HR1 AND DMENU/HR2":FORI=0TO1500:NEXT
30 POKE&HE6E4,&HE6:HSCREEN3:POKE&HE6E4,&HE7
40 POKE&HFFA2,&H70:SAVEM"DMENU/HR1",&H4000,&H5FFF,&HAC73:POKE&HFFA2,&H71:SAVEM"DMENU/HR2",&H4000,&H5BFF,&HAC73:POKE&HFFA2,&H7A
50 LOCATE4,10:PRINT" SAVED DMENU/HR1 AND DMENU/HR2":END
60 CMP:PALETTE0,63:PALETTE1,0:WIDTH80:HSCREEN1:HCLS
70 COLOR1:S$="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL":HPRINT(2,0),S$:S$="a b c d e f g h i j k l m n o p q r s":HPRINT(2,4),S$:S$="a  b  c  d  e  f  g  h  i  j":HPRINT(2,10),S$
80 HPRINT(0,2),"0":HPRINT(0,7),"1":HPRINT(0,14),"2":HPRINT(0,20),"3
90 POKE&HE6E4,&HE6:HSCREEN3:POKE&HE6E4,&HE7:HBUFF1,1824:HGET(32,0)-(639,7),1:HLINE(32,0)-(639,7),PRESET,BF:HPUT(32,4)-(639,11),1
100 HGET(32,32)-(631,39),1:HLINE(32,32)-(631,39),PRESET,BF:HPUT(40,36)-(639,43),1
110 HGET(32,80)-(623,87),1:HLINE(32,80)-(623,87),PRESET,BF:HPUT(48,84)-(639,91),1
120 HGET(0,16)-(15,23),1:HLINE(0,16)-(15,23),PRESET,BF:HPUT(8,20)-(23,27),1:HGET(0,56)-(15,63),1:HLINE(0,56)-(15,63),PRESET,BF:HPUT(8,60)-(23,67),1:HGET(0,112)-(15,119),1:HLINE(0,112)-(15,119),PRESET,BF:HPUT(8,116)-(23,123),1
130 HGET(0,160)-(15,167),1:HLINE(0,160)-(15,167),PRESET,BF:HPUT(8,164)-(23,171),1
140 HLINE(32,15)-(639,15),PSET
150 HLINE(32,47)-(639,47),PSET
160 HLINE(32,95)-(511,95),PSET
170 HLINE(543,96)-(543,191),PSET:HLINE(543,95)-(639,95),PSET:HLINE(543,95)-(543,191),PSET
180 HLINE(31,0)-(31,191),PSET:HLINE(0,32)-(639,32),PSET:HLINE(0,80)-(639,80),PSET:HLINE(0,144)-(31,144),PSET
190 REM LINES 150-520 DRAW DESIGN ELEMENTS FOR KEY COMBINATIONS 0a-0z AND 0A-0L
200 HDRAW"BM32,16BR7BDG3RNE2DG3RNE3DG3RNE3R6ND2R7H3LNF2ENH3RH3UNHRH2"'0a
210 HDRAW"BM48,16BR5R5FL7ND6GD8HNUD2LGDR7D2RU2R7UH2DNU3HU8LD5"'0b
220 HDRAW"BM64,16BDD5RUR2DRU2R7DFR2EU4HL2BD2D2FEU2HGBU2GDL9UL2BR16BU"'0c
230 HDRAW"BM80,16BR3R8M+1,+4L10R5ER2FR3FD2GNL12LD2L2U2BUR2EHL2GFBL7L2HER2FGBDD2L2U2LHU2ERM+1,-4"'0d
240 HDRAW"BM96,16BR4R2DR5F4L2NL13D6L2BUU4L3D4R3BDL6U4L2D4L2U6L2E4NR2UBR2"'0e
250 HDRAW"BM112,16BR2R2FD2GL2HU2BD3NR4GD4FD5R2NU4R2U5EU4HBR4BUU2ER2FD2GL2HBDNR4D4G2M+2,+1RNR2D4R2U4RM+2,-1H2U4"'0f
260 HDRAW"BM128,16BR7D3L2D2R2D8HLHLHURUL2D2F2RFRF2E2RERE2U2L2DRDGLGLGU8R2U2L2U3L2"'0g
270 HDRAW"BM144,16BR6D2L5DRD7L2FR2EL2U7R4D11L3DR8UL3U11R4D7L2FR2EL2U7RUL5U2L2"'0h
280 HDRAW"BM160,16BR2R12GL10NHRBD13LGR12HL10R2BU13R2G2D2F2DG2D2F2R2E2U2H2UE2U2H2R2BR4"'0i
290 HDRAW"BM176,16BR3BDR4M+6,-1M-3,+5M+2,+7D2GL8HU3M+1,-3L2M-1,-3U2ER3D3M-1,+3"'0j
300 HDRAW"BM192,16BR14D10GL2HNR4UNR4ER2FU7M-8,+2U2NM+8,-2DNM+8,-2D9GL2HNR4UNR4ER2F"'0k
310 HDRAW"BM208,16BR8G2DG2L3DF3DM-2,+4RM+5,-4M+5,+4RM-2,-4UE3UL3H2UH2"'0l
320 HDRAW"BM224,16BR4R7FR2FD5GU6L3D5NR2DNR2D2GL5HLNU9RNU9RNU9FRNU10RBD2L8FRFR9ENL11RENL9":HPAINT(233,20),1,1'0m
330 HDRAW"BM240,16BR9BDGL3GDFDFNRGL2FNR2FRFND3RNU3E2R3HLHLH2R2U2L2NGR2F2NDHEUH"'0n
340 HDRAW"BM256,16BR3G2NUDLD5NF7EDF6NE6RE6NHU5LU2LNL3HL2G2NRNH2DH3DL"'0o
350 HDRAW"BM272,16BR7DGDG5UF5DFNDEUE5DH5U"'0p
360 HDRAW"BM288,16BR6R3F2D4R2F2D3G2L3HLD3NRL2RU3LGL3H2U3E2R2U4E2":HPAINT(296,24),1,1'0q
370 HDRAW"BM304,16BR7GDG6D3F2R2E3D5L2ER3FL2U5F3R2E2U3H6UH":HPAINT(312,24),1,1'0r
380 HDRAW"BM320,16R15D15L15U14BR2BDR11D11L11U10BR2BDR7D7L7U6BR2BDR3D3L3U2"'0s
390 HDRAW"BM336,16BR2R11F2D11G2L11H2U11ER13D13L13U13BF4GD2F2R2E2U2H2L2":HPAINT(344,24),1,1'0t
400 HDRAW"BM352,16BD7R2ERE2UEU2RD2FDF2RFR2BL15D8R2ERE2UEU2RD2FDF2RFR2U8":HPAINT(354,26),1,1'0u
410 HDRAW"BM368,16R2FRF2DFD2RU2EUE2RER2BL15D8R2FRF2DFD2RU2EUE2RER2U8":HPAINT(372,20),1,1'0v
420 HDRAW"BM384,16BR7RD2R5G11R5D2RE3RFNU5H11D5LGF3RBU8E3BR5BD4F3G"'0w
430 HDRAW"BM400,16BDF4E4F4E3UG4H4G4H3BD6F4E4F4E3UG4H4G4H3"'0x
440 HDRAW"BM416,16R3DL3DR3DNL3FR3DL3DR3DL3GL3DR3DL3DR3FR3DL3DR3DNL3BU3ER3UL3UR3UNL3ER3UL3UR3UL3HL3UR3UL3UR3BD11FR3DL3DR3DNL3"'0y
450 HDRAW"BM432,16R15DL15BD3F7L7E7NL7R8G7R7NH7BD3L15DR15":HPAINT(436,21),1,1:HPAINT(444,21),1,1:HPAINT(436,25),1,1:HPAINT(444,25),1,1'0z
460 HDRAW"BM448,16BD3E3BR4G7BD4E11BR4G15BR4E11BD4G7BR4E3"'0A
470 HDRAW"BM464,16R15D3L7NU2LNU2L7NU2DR15L4D3NR4L7NU2LNU2L3DR15D3L7NU2LNU2L7NU2DR15L4D3NR4L7NU2LNU2L3"'0B
480 HDRAW"BM480,16BD3E3BR4G7BD4E11BR4G15BR4E11BD4G7BR4E3BU8H4BL4F8BD4H12G3F12BL4H8BD4F4"'0C
490 HDRAW"BM496,16R3F3DFD2L2HLH3U3BR15L3G3DGD2R2ERE3U3BD15U3H3LHL2D2FDF3R3BL15U3E3RER2D2GDG3L3"'0D
500 HDRAW"BM512,16R7D7NH7RE7D7L7DF7L7U7LG7U7R7":HPAINT(517,19),1,1:HPAINT(514,26),1,1:HPAINT(525,22),1,1:HPAINT(522,29),1,1'0E
510 HDRAW"BM528,16R3D5RE2R3F2RU5R3BD3D9LNU9RBD3L3U5LG2L3H2LD5L3BU3U9RD9U4BR5R3U2L3DR2"'0F
520 HDRAW"BM544,16F5R5NE5D5NF5L5NG5U4BU5LR7LG2LHR2BD5LDRBR4F2DU7DGD2HBL9DG2DU7DFD2BD4BR5RF2RL7RER2"'0G
530 HDRAW"BM560,16R15D15L15U14FE2RF11DG3LH11D7E11RF3DG11LH3BR3RULBU7URDBR7URDBD7DLUBH3LUR"'0H
540 HDRAW"BM576,16R15"'0I
550 HDRAW"BM592,16D15"'0J
560 HDRAW"BM608,16BD15R15"'0K
570 HDRAW"BM624,16F15"'0L
580 REM LINES 540-730 DRAW DESIG
```

```
N ELEMENTS FOR KEY COMBINATIONS
1a-1s
590 HDRAW"BM32,48BD16BR2R13GL12D
R11G2D3R2UR2UR2UR2UR2UR2UR2UR2UR
2U2L2DRM-16,+8M+16,-8LL2DL2DL2DL
2DL2DL2DL2DNL2BR16BU8UH5LG3L12G6
"'1a
600 HDRAW"BM64,48BD7BR2E2RER20FR
F2D2NL3D2L6H2U2F3D3FD2FD5GL20HU5
EU2EU2L5U3DNR3D2R5UE3D2G2E2U2R3D
R6UR3BD3BL3L5GLGD2FRFR4EREU2HLBD
7R4L14"'1b
610 HDRAW"BM96,48BR15R3FRFR2UD3F
D2G3L3HU2E3NED2RUL2G2LND2H3D2LUE
H2RERLGL2UD3GD2F3NR3GD2F2HG2D2BU
10G2D2GD2GD3RFRBL3D2FDL3DR3DFDF2
R2ER6E3UL3NR8DLUL4DLUL5UR4UR2DLR
4ULR2DR3NR5E2U2EU4HNU5L2D5FHU5LG
BDBL2L4"'1c
620 HDRAW"BM128,48BR11BD2GLGLGDG
6D4F2DFR3HR2GND3L2DL4NUR4D6RNU4D
RUBR2D2R9UNL8UL3U5E2U4LGD2GHLHLH
LU2BL3LBR4UHU3BU2BRR2ERERFGD2GD8
FGD6L3URU4NH3D5R6U2RE2R3ER2E3U3E
RD4RLU4LU4HUHUH4L2HL9"'1d
630 HDRAW"BM160,48BD6R4ULRER2F2E
2F2R5ERER4FR5L3DLGD2GD2L2EH2LGLG
L4H2EUDGDGDL2EUH2U2D3G2H2UNUL3UL
"'1e
640 HDRAW"BM192,48BR9BD2GLG2DNFG
DF2R2DNFE2UL2BU2UFBR2DFE3F2D2GD3
NRLULHLHBD3R2DG2UER3D2HE2RFDRD5G
DG2E2UEU2RUE2LER2EREUEU4GND3LGFD
3LNU2LDL2U2EUEBD2BL6DRDELBU7BL4U
HNL4GDGHL"'1f
650 HDRAW"BM224,48BR8BD4D3G2RDFD
2NU6RU5FD4RU2EU4L3UF2NLDR2DR2GND
LD3LDRD2FBD2NRBL2BDDBU3LBRHBU2R6
HL2ERE2RLU4G2FDLDBU3L2NFD3L2NFD2
L"'1g
660 HDRAW"BM256,48BR18BD4R2FNLDL
4NUGDGDGDGDGDGDGDGD5E2NLE2UEUE
UEUEUEUEUEUEUL"'1h
670 HDRAW"BM288,48BR10BDNR9GFD2G
D2G2DRF2L3GR3FNR3GBD2R3FBL6U4LGD
2RNU2F2LBD2RD3R2ER3NE2D3RER2ER3N
U4D4FDU2EU2EU4HNU9LNU6BL7LE3U3HL
GFDBR12D2FDU7H2RHL2D2F2DBU9G2RGL
H4NDL4HL3GE2H2R3F4R2UH3R2FR2DRND
4F3NGE3LGR3DRLHG"'1i
680 HDRAW"BM320,48BR20R2FR6FGD2G
DF3D5GDG3LGL3D4GDG3LGL8HLH3UHU5E
3UHU2HER6ER2FR9FRFRD3LUL6U2ER4DL
4DNR5G2DF3BL4D2GDG3L4H3UHU2FR10B
U6NL4HL2BL4L2GR4BU5E3UHU2HER6BD3
NRLGLR4BR2R4LHL2":HPAINT(332,72)
,1,1'1j
690 HDRAW"BM352,48BR3BD2NRD25R6E
R3FR2NU23R6G3R8H3R5NU24DE2U22H2L
G2L2HL5GL4HL6HLBR8UFDF6DFDFD2FD5
ED6NLNRU6H2NDU2GNU4UHNU4UHNU4UHN
U3UH2U2LU3BL7BD4R3BRRBD2NRL2BL2L
3BD2R2BR2R5BD2L3BL2L4BD3BRR3BR2R
2BD2NRL3BL2L3BD2R2BR2R2BR2RBD2NL
9BR4NR3BU2NRBU2R
700 HDRAW"BR5R3BD2L2BD2RBU7L3BU2
NR3BU2LNR5BU2LR2BR2R2BU2LBL2L4":
HPAINT(376,77),1,1'1k
710 HDRAW"BM384,48BR19DF2D2G2DR2
D19R3U2R5U5L4UR5D7LG3L2D2L9UHLH4
R2DRDR5NR6U19R2UH2U2E":HPAINT(40
0,78),1,1'1l
720 HDRAW"BM416,48BR15R5FR3FNRD2
GDGD8RER3NUDGDG7R2D7RU2FRFBL18U2
HUEG2H2LUHU5EUE2RER9FR6U6L9GL2L2
GLGLGFH5UE4RERER2ER2BD16R9NF2L9G
L3DLGDNFLGND4DBR2NFBR4ERFRFNR2L4
FR2BDBR3GD2NRLD2R4EUHU2BD5BL14HU
2LGL3":HPAINT(426,60),1,1'1m
730 HDRAW"BM448,48BR21D2GD5FD4FN
U11DUHG2L2UHUHUHU4LHLH2UHBR11BDL
4NGFR2BD2L2BD3D2FDBR2NDU2HUBR8GD
2GD4U4EU2EU3REFD2NFU2HU2HBL3F2L2
D2RBD11BL13HUHU3HU3H2U2HBL6BD15N
R31BD2NR31BD2R13E2RF2NR13F2R3NR8
"'n
740 HDRAW"BM480,48BR3RF2RFR2FRFR
F2D2FER5FRF5D7GD3FGLGL2D2BU9HUHU
2E2R2FBD3LDRBD11BL15UENR2HUHUHUH
U3EU2L3H4UH2UH2U2R4U2HFD2FDFDFDF
DF3BU4BRH5U2FRFRF3"'1o
750 HDRAW"BM512,48BR12GLGLGLG4DG
DGD11FND3EU2E2R3F4D5G4L2H3UDF3R2
E2FR5FR8UR2E4U2LG2DNG3UEHEU2EU5H
U5HU2H3F2E3REUL4GLU"'1p
760 HDRAW"BM544,48BR16G3D3FD2FE4
HUHUHBR4BD4R2FG4D3R3E2UHGEFEU2NH
EFGFERUNHDF2D6GDG2UH2LHLNU2BD2BL
D3FE3GFDGDFDND5GLH2LGF2DGDEUH2LU
LHL3HGL2DGND5EUE3U7EU4"'1q
770 HDRAW"BM576,48BR9LGERF2R3F2R
E3UBR7D3FD4GD6GD3G5R2E2FD3G2NR6U
NL4HUL6DNR3UER2L2U3L2GD4NRGD4BM5
83,49DGD3GD8GD5GDRDNR3GDGD2G2RE4
UEGD5BR5U11LGD2GEU2E4U2E3U5EU3D3
GD3L2GLGLGL3G"'1r
780 HDRAW"BM607,48BR7GD4GD17FD8U
2E2U2ER2U4EU5EU8EU5BL2GD11GD6GD4
"'1s
790 REM LINES 750-1050 DRAW DESI
GN ELEMENTS FOR KEY COMBINATIONS
 2a-2j AND 3a-3j
800 HDRAW"BM32,96BR20G4L3GDGDGD3
GD2LF3L2HRD5U2LUF2D3ED2ED4R2FD3F
G9LNG2RE7FDF3DFDREREF2RU2R2F2R3U
ER2FRF2R2L2H2LHLH3LGL6HLHLH4BR3U
3LH4NLEFNDR4HUH2L2ND3U3HUHLHLDBR
7DNGUH2U3EU2E4R3FRF2DFDR5UEFRUHU
H5LHL8BD16R7D2G2LH3RGNDL2DBR13DE
R2E3NDEU3LH3
810 HDRAW"D3FD2G2L3DFR3D8GD3FD3N
G2BL8L3HNL3BR5R2EULHL3BL3BUER7UL
H2R3NGENUR3U2RU3BL11LH2GUL2DE3ND
NL2R5D2GBD11BR4E":HLINE(35,141)-
(76,143),PSET,BF
820 X=32:Y=96:HPAINT(X+24,Y+2),1
,1:HPAINT(X+37,Y+14),1,1:HPAINT(
X+25,Y+17),1,1:HPAINT(X+27,Y+37)
,1,1:HPAINT(X+14,Y+43),1,1:HPAIN
T(X+17,Y+25),1,1'2a
830 HDRAW"BM32,144BR28BD4L5GL4GL
GLG2ND4LGDGD4LGD2DGD5F2DFL3DR6NR
6G2L2DGR3ENL2FDGD2G5LR10URUHUEHU
2RF2RFRF2R2DRLU2NHR2ER4DG2DRFNL1
7R13HUHUL4HL3HL8HLHLH4U2NLRFREU3
HUHUHU2G2DGHU2HLGHU3E3ND2EUHBR5G
D5FD
840 HDRAW"R3DU2R6GDBL2LBR3DFDNR2
D2NR2D2NR2GDR6L2HU6HND3U2BR3D2NF
ENRBU2R2EU2HU2NHFRFDFD3F2DFD3HGD
2NRL2NU5DGD2GDGDGDBL3BU3L2BU3LGE
R6L2BU27LHNLFRF2RGFNRF2DBL16UEUG
2FD2NLD2NLFDL2
850 Y=144:HPAINT(X+15,Y+34),1,1:
HPAINT(X+35,Y+41),1,1'3a
860 HDRAW"BM80,96BR22BDNR4GL2GL2
GLG3LGLGD2FDGD3F2R2HLG2DGD3FD2GD
3GD4GD2GDFRFRFR2FR6ER2ERER3EUEU2
EUE6U6H3RE2RFD3FD2GDF2RE2UH2NLU6
EUEH4UHU4H3L3HL3BD6RFNG2RF3D2FNG
3D5GH2U2HUHNE2H3LHLNE3L2NU2L5NU2
LGL2GNL2EUHUERENF2R4ERND2R2FRF
870 HDRAW"BD8D4G2NL2E3NUDF4NEDG3
LGNLE4U3HD4G3NUL7H3NRHUR3UL2URU2
EUR2UR5UL3BD4NR2L2GNR4BL5UH3NL2F
3DGDGD2GDNL2FR2ERBD3RG2L4U2NLR4D
RBD5DFD2FDFBL10EUEUEU2BL3UNED2GD
EDUBR16HUHUHUBR5UNHDFDF":HPAINT(
106,100),1,1'2b
880 HDRAW"BM80,144BR22BD2G6DGDGD
GLGD3G2D8FDF3R2D2L3G3LG3D4R42U3H
4LHLHL2HL2HGD2GDG3L5H3UL6GLGD2FN
DE3R4FGFGLBL3E2RBL10ND2EUE4R5NFH
2NDR2ER2E3UEU8H3L2HL4GL2F3R2F3D3
G2R3EGL9HBR3NR4HNR6U4LGER2ND3RND
3R3ND3FNL3
890 HDRAW"ND2U2R2GR2DER4D8F2LGL3
NHD3L3RFDR5U2NR2LNL2U3BR4UE2RD2G
E2RE3U5EU7LDGDG2DR2D7HD2LDLU4EU5
L2GFGDGL4U2NR4UR5HBU6ER2EFRLHUHU
HUH6L5BD13NR8L4HL2":HPAINT(118,1
63),1,1'3b
900 HDRAW"BM128,96BRBD12R22ER6ER
2ER3ER2ER5D2G2LGLGLGLGL3G3R2D2FD
FDR3ER2ERE2U5HL2G2DGL8GDG2D2LGR1
4DGL15G2DR2DGL3HE4L4HER6HUH2UH2L
12G2LUHLF4U2FDR2F5GDR2DGL3HE4NR2
H2EU3H2GD5NHNDF2L5H2L3FR3U6
910 HDRAW"ER3U2LUR5DGFRE3R11DG2D
G2DGU3NRH2LNF2LF3DH4L2D6GBU11L15
NUR4DR9ER15ENL3ER3ND2ER2ERER2E2"
:HPAINT(169,106),1,1:HPAINT(158,
118),1,1'2c
920 HDRAW"BM128,144BR25BDRFRD4GH
2UEND2BL3EG2DF2DGD2R4EUNHEUE2UHU
R2FDFDGHGDNGRDGD2NL3F4D3FD4GD4FD
5F2NLERDG2L2HUH3UHE3GFD2UBL3L3GD
7G2H2U7G2DGDG2HU4EUENR3H2GD2U2EH
LGLG3LHU2E3RFNG2HER2ND2UHGL2UHL2
DNR2F2L3HU4ER2DR2FR2NDU2HLH3GDNF
2NDH2L
930 HDRAW"U2ER2F3RFR2ND2U2H5E2L3
GD2F4BR3REH4U5RFE2UER2DGDFNF2G2N
H2RNF2GFD2NR3GH2UHUBR20DGDGFE3UE
F2NG5H2R3FD2G4DGLRERE3R2FD2L2DG5
NLER2UNER2FD4GL3E2NL3UL2G2LDFRF3
D2GLH3ENHFBL11HL4GBU3LH3BU2LURBR
6G2R6HNL3HNLUL2EBD19U5NL
940 HDRAW"R2F2DFDF2EU3BL17GDGL2H
2URFRUEU4BU4BR8R7E3BU2RULBU3BLUH
LGDBL6UHLGDBU4BR2REBR4FRBR6BU8ER
D2BD12RE3"'3c
950 HDRAW"BM176,96BD15D11R2U11LN
D8R3D9LER3F2RFNU5R3NU5R3NU5DR2EN
U4EFREFRENH2FREUNH3DFRE2NH5ER8NU
9DRFNU11R2U11LND10L7HL3HLHL2HLHL
3GHLG2LGL2GLGL2GL4BR22DNF2UHUHG2
L3UE2BD10BL3L8GL2"'2d
960 HDRAW"BM176,144BD34FDF5R5U2E
U4H4U3LGD3F2D2L2HNENU13LHUNEHGND
E2UE2U5R7U3HU3NG5UHU3ER3ER4ER5ER
3D5R2F2DFE2UEU3HL2GLGL2UE2RERER2
F3DFDFD2GD4FD4ERDFDFDFD13L4UHU2H
U2HUHUHUHU2R2ERBL4UHLD12NL22H2UH
2LHLHERE2RE2NL4ULHU2HU2H3FD5FD6G
L
970 HDRAW"GL11GLGLG2BU7L2NHR2FRF
NR2BU2UBR6RBF2NLERBR2BURBU3GBL2G
LBU3NLRER2BU3LGL2BL2BDLG2L2BU3GL
BU3REREBR2RERBR3RBU3NRGLBL3GL3BL
2BDL3BD11BR3REREBU6GNLER2BR2BUR3
BD11BL8L":HPAINT(221,184),1,1'3d
980 HDRAW"BM224,96BD39BR47LGL2H5
GD2LRF4R2NUFR3LU2RLD2L2DGD2BL3U2
E2G3L2GBL14HUE2UE6UE10UE3UE5U5D3
LH3UH7G6F3E3F3DF2DG4DG9DG3DG11F2
"'2e
990 HDRAW"BM224,144BR18F3DF5U2E6
RER2BL13F5DGDG3D4FD3F3D2FD11G2DG
2D2R2FR7FRFR2FR6U2H4L2HL3HL2HU2E
U2R2ER2E2FR3ERLG2DGDGD3R3FRBU9BL
```

```
6E3REUEU3LHL2HU2EUE4UBM267,144D2
FDGL2HU2HU"'3e
1000 HDRAW"BM272,96BD43BRLRFR2FD
2BU6R2E3RUE2ND12U3HUBL3G9D2NR4U2
E9UE3R2F2NED5ND5R2NU3D4GLDFD5FND
EU2EU4EU7HNL2E3L6H2UL3NG2RU2RL4G
LRER2U3ENF2HL5GER5FER3FRLHU2EUER
ERE4D3FRFR2URNRDG2DL4HL3H2LGDFRF
R3BD2LRBU2FR3FR4NH3L3DBL3LBR4D2N
L4NR3D
1010 HDRAW"F2DNL2R2FRFHLHL2D2L2R
2FRD2FDUHU2L2GLHFDFD2U2HL5H2BD19
GER3EUEFEFEFD2L2GLRER13DUL11U2E2
UE3U3R2U5LRD8GDGDG2DBU10BR4REFGD
FG2DG2DGBM272,108NE2BD2R3"'2f
1020 HDRAW"BM272,144BR5GDGDGDG2D
3BD11FR2F2D2L3NH3RFGLGLBD4D3R5FR
3F2NR35F3DGLHL2HLHL3HL2BU7R3E6LN
H2RF2R4EU4HU15BL2BU9D2F2RND5R30N
D36L2NU4L28ER2ER3EREREBL5L2GLU"'
3f
1030 HDRAW"BM320,96BR20BD14DFER3
FDG4DGD2ND3L4GLD3GD5FDF4DFDNF3UH
UHU3HU3EUEUER2DUR3GDGDFD2F3DF8RB
R11BU13LHLHL2HLHLHLHL2HNL5BU14BL
2L2EBL7GDUE2R4ER8GF2D2FD8GD2F2"'
2g
1040 HDRAW"BM320,144BR24F7RFRF13
D6L4DL3HG2D2RER4DFDR2U2RBU3GL2BR
3BU8H5U3HU3HU4ENR6H3LRF4RFRF2BL2
2NU5D5FD4FD2LD4L2GL5UD2L2D2R2UR3
FR2ER2L2GL2HD2LDRERFRER6ERLHLU4N
H3FRFR2EFR3U3L2U5NU3L4HU2"'3g
1050 HDRAW"BM368,96BD34F3RF4DF4R
F"'2h
1060 HDRAW"BM368,144BD3R6ER2ER2E
RDF4D2NL2RDF4DL2HFR2DF3DNL2DF2DF
DL3HLH2LH2LH2LH2L2H2LNDHLG2LRE2H
LH4BD24RUR4UNRHL2U2R4HLHL2H2F2R2
FRF2RE2NU10F2RDFRFR3DFR3UEUH2UH4
F2R2FR2U"'3h
1070 HDRAW"BM416,96BR47BDL3HL3GL
2GL2G2LG3DG2D2GD2GD3GD7GDFD5G4D2
F3R2ER2DGDGBU6LR2U2H2NL2F2E2U2EU
6EU4E2UE2RERER2ER2E4BD9GBD3DG3LG
L2G7DGDBR16BU11LDGDFNRGD2G2DFDGD
2FRNEDGL3U2D2GD"'21
1080 HDRAW"BM416,144BR41FD4GDG2L
F5RFRFBD11LHL2HL2HLHL2HLHNL4E2UE
U6RE2G2L2G3BD7L3GFDG2L3HNH5U2E2R
2H3BL6ND2U3ERER3E2U2HBU2E2UDG2LG
4DNF2BU3HU4ER3E2UE"'3i
1090 HDRAW"BM464,96BDER3F6D4HLGH
L3HNL2G2BR7D2GDGDG3NL2D3FD5GFR2U
G2L2NHGBD3BRF5RFRND6RER2ER2EU5EH
LGEREUHGLUBR4GE2UH6BL2LG2FREU2BU
2BLL2NG3BR5BDU2HUHUHU"'2j
1100 HDRAW"BM464,144BR9D3GD2F3RF
RFR2F3D3G2L11HL3HLHBD11R15ER2ERE
4UEUH2UHLRFDGDG3LGL2"'3j
1110 ON BRK GOTO 1130
1120 GOTO1120
1130 WIDTH40:CLS:LOCATE2,10:PRIN
T"DO YOU WANT TO SAVE THE JUST D
RAWN       SCREEN?  (Y/N) ";
1140 K$=INKEY$:IFK$=""THEN1140EL
SEIFK$="Y" OR K$="y"THEN20ELSEEN
D
```

```
120 .......... 244
180 .......... 153
END ......... 125
```

Listing 2: GENSTR

```
10 '***     GENSTR     ***
20 ' BY H. ALLEN CURTIS
30 '   COPYRIGHT 1990
40 CLEAR500:DIMI$(11)
50 AM$(1)="1: BORDER (HORIZ. 16
DOT DESIGNS)
60 S$(1)=STRING$(39,93)
70 AM$(2)="2: BORDER (HORIZ. 32
DOT DESIGNS)
80 S$(2)=STRING$(19,93)
90 AM$(3)="3: BORDER (VERTICAL)
100 FORI=1TO11:S$(3)=S$(3)+CHR$(
93)+CHR$(94):NEXT:S$(3)=S$(3)+CH
R$(93)
110 AM$(4)="4: ADDRESS
120 WIDTH40:CLS:LOCATE5,8:PRINT"
PRINT YOUR STREET ADDRESS":LOCAT
E5,10:LINEINPUTS$(4)
130 CLS:LOCATE5,8:PRINT"PRINT YO
UR CITY, STATE, AND ZIP":LOCATE5
,11:LINEINPUTQ$:S$(4)=S$(4)+CHR$
(94)+Q$+CHR$(94)+CHR$(94)
140 AM$(5)="5: LETTER CLOSING
150 S$(5)="Sincerely yours,"+STR
ING$(3,94)
160 CLS:LOCATE8,8:PRINT"PRINT YO
UR NAME":LOCATE8,10:LINEINPUTQ$:
S$(5)=S$(5)+Q$+CHR$(94)
170 AM$(6)="6: PHRASE OR SENTENC
E
180 CLS:LOCATE2,8:PRINT"PRINT SH
ORT SENTENCE OR PHRASE":PRINT"
THAT YOU MAY OFTEN USE, SUCH AS"
:PRINT"  A NEWSLETTER TITLE.  (U
SE ^ TO":PRINT"  SIGNAL CARRIAGE
 RETURN)":LOCATE2,12:LINEINPUTS$
(6)
190 CLS:LOCATE2,8:PRINT"INSERT T
HE FILE DISK IN YOUR DRIVE.":LOC
ATE10,10:PRINT"THEN PRESS ENTER.
"
200 K$=INKEY$:IFK$=""THEN200
210 CLS:LOCATE2,8:PRINT"THE FILE
 STR/DAT IS NOW BEING SAVED.
220 OPEN"O",#1,"STR"
230 FORI=1TO6:PRINT#1,AM$(I):NEX
T
240 FORI=1TO6:PRINT#1,S$(I):NEXT
:CLOSE#1
```

Listing 3: GENMLT

```
5 REM GENMLT
10 CMP:WIDTH40:CLS3:LOCATE8,2:AT
TR0,2:PRINT"GENERATING ML ROUTIN
ES":LOCATE6,5:PRINT"NOW GENERATI
NG SCREEN DUMPS"
20 LOCATE0,8
30 DIMS$(17),C(17)
40 X=&H1200
50 S$(0)="1F314F8D3027028A018D27
27028A028D2127028A048D1B27028A08
8D15":C(0)=1763
60 S$(1)="27028A108D0F27028A208D
0927028A408A807EA285308850E684D4
5039":C(1)=2617
70 S$(2)="C680D7508DC0045026FA39
861B8DE686108DE24F8DDF865020DB86
1B97":C(2)=3652
80 S$(3)="528D338DE98D2A8DDA3341
0A5126F88D1B20008E7273BFFFA28DC8
3341":C(3)=3319
90 S$(4)="0A5126F88D09860D8DB10A
5226D6393388151E32C63CD751398E70
71BF":C(4)=2850
100 S$(5)="FFA2398E0EF01E1434108
D1B861B8DB58D238DDE20008DE48D1B8
E7A7B":C(5)=3134
110 S$(6)="BFFFA235101E410F6F39C
6FED76F8612CE40001F327EA2852000C
6788C":C(6)=3147
120 S$(7)="C6F08CC6508E4000A6C0A
780A6C0A7805A26F5398E0EF01E14341
08DD0":C(7)=3661
130 S$(8)="17FF698D17200017FF9B8
D104C17FF5E861E8DC78D0220AB860D2
0BF30":C(8)=2885
140 S$(9)="C90190C6504FA780A7805
A26F939860E97528D3821E517FF6A17F
F1933":C(9)=3305
150 S$(10)="410A5126F717FF598D2B
17FF0B33410A5126F717FF4B33C9FDD0
8D1D17":C(10)=3032
160 S$(11)="FEF933410A5126F717FF
398DB40A5226C7398670B7FFA2398671
20F886":C(11)=3601
170 S$(12)="7220F48E0EF01E143410
8DE98D258DEB8D218DEB8D1D17FF438D
9CCE40":C(12)=3464
180 S$(13)="C81F32860D8D958DCF8D
398DD18D358DD18D3116FF1C2000C602
CE4028":C(13)=3195
190 S$(14)="8E5E0086289751A6C0A7
800A5126F833C8283088285A26EB39C6
01CE5E":C(14)=3106
200 S$(15)="A08E40C820E0C60420F4
2000C603CE5D388E400020D0C606CE5D
3820F4":C(15)=3169
210 S$(16)="C602CE5F6820ED8E0EF0
1E14341017FED717FF2F8D1117FF688D
0C17FF":C(16)=3176
220 S$(17)="698D0720850102030405
8E5EA016FF0B":C(17)=1117
230 K=17:N=50:GOSUB570
240 LOCATE4,5:PRINT"NOW GENERATI
NG DESIGN OPTIONS"
250 GOSUB580
260 X=&H1000
270 S$(0)="8E0EF01E1434108E7071B
FFFA2CE7C004F5F1F02362611834002
6F88E":C(0)=2758
280 S$(1)="7A7BBFFFA235101E41390
103000045044F045E046D048E0EF01E1
43410":C(1)=1959
290 S$(2)="8E7071BFFFA2C660D750C
E7C444F5F1F0233C8BC3626362636260
A5026":C(2)=3012
```

```
300 S$(3)="F3318CCDA6A430B64A810
826024A4A974FE6213D3085964F48484
89750":C(3)=3007
310 S$(4)="A63F4A2633D64FA680A7C
05A26F9C650D04F308533C50A5026EB2
061A6":C(4)=3367
320 S$(5)="845F482402CAC0482402C
A30482404CA0C2100482402CA03E7C03
94A26":C(5)=2357
330 S$(6)="1FD64F34048DDB8DDB300
135045A26F3C650D04F3085D04F33C50
A5026":C(6)=2986
340 S$(7)="E32023D64F34048DBCE7C
84F8DB9E7C84F300135045A26EDC650D
04F30":C(7)=3402
350 S$(8)="8533C533C50A5026DDA62
21027FF25866097508E5E448D50A60B8
D4EE7":C(8)=3138
360 S$(9)="80D64FE70A8D44A6098D4
2E780D64FE7088D38A6078D36E780D64
FE706":C(9)=3598
370 S$(10)="8D2CA6058D2AE780D64F
E7048D20A6038D1EE780D64FE7028D14
A6018D":C(10)=3293
380 S$(11)="12E780D64FE781308848
0A5026B116FECCA684D74F5F482402CA
014824":C(11)=3179
390 S$(12)="02CA02482402CA044824
02CA08482402CA10482402CA20482402
CA4048":C(12)=1962
400 S$(13)="2402CA8039":C(13)=42
5
410 K=13:N=270:GOSUB570
420 LOCATE4,5:PRINT"NOW GENERATI
NG HALF SCREEN IO"
430 GOSUB580
440 X=&HF00
450 S$(0)="34761F52313A1E428E707
2BFFFA28D188E7A7BBFFFA28E7173BFF
FA28D":C(0)=3741
460 S$(1)="0A8E7A7BBFFFA21E4235F
68E4000CE6000A684E6C4A7C0E7808C6
00026":C(1)=3624
470 S$(2)="F339108E0EF01E428D1CC
628A6C0A7805A26F930882833C8280A5
026ED":C(2)=3125
480 S$(3)="8E7A7BBFFFA21E42398E7
071BFFFA2CE5E008E4028C660D750391
08E0E":C(3)=3487
490 S$(4)="F01E428DE7C628A680A7C
05A26F930882833C8280A5026ED20C91
08E0E":C(4)=3261
500 S$(5)="F01E428E7077BFFFA28D8
C8E7A7BBFFFA21E4239":C(5)=2746
510 K=5:N=450:GOSUB570
520 CLS5:LOCATE2,8:PRINT"INSERT
THE FILE DISK IN YOUR DRIVE.":LO
CATE10,10:PRINT"THEN PRESS ENTER
."
530 K$=INKEY$:IFK$=""THEN530
540 CLS5:LOCATE7,5:PRINT"NOW SAV
ING ML ROUTINES":LOCATE7,7:PRINT
"IN FILE CALLED MLR/BIN":LOCATE2
0,10
550 SAVEM"MLR/BIN",&HF00,&H13FC,
&HAC73
560 CLS3:END
570 FORI=0TOK:FORJ=1TO29:PRINT".
";:A$=MID$(S$(I),2*J-1,2):A=VAL(
"&H"+A$):C=C+A:POKEX,A:X=X+1:NEX
T:IFC<>C(I)THENCLS3:LOCATE8,12:P
RINT"TYPING ERROR IN LINE";N+10*
I:ENDELSEC=0:NEXT:RETURN
580 LOCATE0,8:FORJ=1TO14:PRINTST
RING$(40,32);:NEXT:LOCATE0,8:RET
URN
```

Listing 4: GENMLE

```
10 PCLEAR1:CMP:WIDTH40:CLS3:LOCA
TE8,2:ATTR0,2:PRINT"GENERATING M
L ROUTINES":LOCATE6,5:PRINT"NOW
GENERATING SCREEN DUMPS"
20 LOCATE0,8
30 DIMS$(13),C(13)
40 X=&H1200
50 S$(0)="1F314F8D3F27028A808D36
27028A408D3027028A208D2A27028A10
8D24":C(0)=2063
60 S$(1)="27028A088D1E27028A048D
1827028A028D1227028A017EA285C680
D750":C(1)=2268
70 S$(2)="8DC4045026FA39308850E6
84D45039861897528E7071BFFFA21700
B08D":C(2)=3447
80 S$(3)="2A8DDC33410A5126F88D1B
20008E7273BFFFA28DCA33410A5126F8
8D09":C(3)=3061
90 S$(4)="860D8DBB0A5226D1393388
151E32C63CD75139860C97528670B7FF
A28D":C(4)=3136
100 S$(5)="748DEE8DA033410A5126F
88DDF8671B7FFA28D9133410A5126F88
DD033":C(5)=3679
110 S$(6)="C9FD808672B7FFA217FF7
D33410A5126F78DBC860DBDA2850A522
6C139":C(6)=3665
120 S$(7)="8E0EF01E143410C6FED76
F8D24CE40001F32200017FF678D298E7
A7BBF":C(7)=2993
130 S$(8)="FFA235101E410F6F398D9
4CE40281F328D8D20E63410308C208D1
13510":C(8)=2503
140 S$(9)="393410308C2020F434103
08C2320EDE6802708A680BDA2855A26F
83902":C(9)=2799
170 K=9:N=50:GOSUB700
180 X=X+31:FORI=0TO59:READA:POKE
X+I,A:NEXT
210 LOCATE4,5:PRINT"NOW GENERATI
NG DESIGN OPTIONS"
220 GOSUB710
240 X=&H1000
250 S$(0)="8E0EF01E1434108E7071B
FFFA2CE7C004F5F1F023626118340002
6F88E":C(0)=2758
260 S$(1)="7A7BBFFFA235101E41390
103000045044F045E046D048E0EF01E1
43410":C(1)=1959
270 S$(2)="8E7071BFFFA2C660D750C
E7C444F5F1F0233C8BC3626362636260
A5026":C(2)=3012
280 S$(3)="F3318CCDA6A430B64A810
826024A4A974FE6213D3085964F48484
89750":C(3)=3007
290 S$(4)="A63F4A2633D64FA680A7C
05A26F9C650D04F308533C50A5026EB2
061A6":C(4)=3367
300 S$(5)="845F482402CAC0482402C
A30482404CA0C2100482402CA03E7C03
94A26":C(5)=2357
310 S$(6)="1FD64F34048DDB8DDB300
135045A26F3C650D04F3085D04F33C50
A5026":C(6)=2986
320 S$(7)="E32023D64F34048DBCE7C
84F8DB9E7C84F300135045A26EDC650D
04F30":C(7)=3402
330 S$(8)="8533C533C50A5026DDA62
21027FF25866097508E5E448D50A60B8
D4EE7":C(8)=3138
340 S$(9)="80D64FE70A8D44A6098D4
2E780D64FE7088D38A6078D36E780D64
FE706":C(9)=3598
350 S$(10)="8D2CA6058D2AE780D64F
E7048D20A6038D1EE780D64FE7028D14
A6018D":C(10)=3293
360 S$(11)="12E780D64FE781308848
0A5026B116FECCA684D74F5F482402CA
014824":C(11)=3179
370 S$(12)="02CA02482402CA044824
02CA08482402CA10482402CA20482402
CA4048":C(12)=1962
380 S$(13)="2402CA8039":C(13)=42
5
390 K=13:N=250:GOSUB700
410 LOCATE4,5:PRINT"NOW GENERATI
NG HALF SCREEN IO"
420 GOSUB710
440 X=&HF00
450 S$(0)="34761F52313A1E428E707
2BFFFA28D188E7A7BBFFFA28E7173BFF
FA28D":C(0)=3741
460 S$(1)="0A8E7A7BBFFFA21E4235F
68E4000CE6000A684E6C4A7C0E7808C6
00026":C(1)=3624
470 S$(2)="F339108E0EF01E428D1CC
628A6C0A7805A26F930882833C8280A5
026ED":C(2)=3125
480 S$(3)="8E7A7BBFFFA21E42398E7
071BFFFA2CE5E008E4028C660D750391
08E0E":C(3)=3487
490 S$(4)="F01E428DE7C628A680A7C
05A26F930882833C8280A5026ED20C91
08E0E":C(4)=3261
500 S$(5)="F01E428E7077BFFFA28D8
C8E7A7BBFFFA21E4239":C(5)=2746
570 K=5:N=450:GOSUB700:GOSUB720
580 CLS5:LOCATE7,5:PRINT"NOW SAV
ING ML ROUTINES":LOCATE7,7:PRINT
"IN FILE CALLED MLR/BIN":LOCATE2
0,10
590 SAVEM"MLR/BIN",&HF00,&H1380,
&HAC73
600 CLS3:END
700 FORI=0TOK:FORJ=1TO29:PRINT".
";:A$=MID$(S$(I),2*J-1,2):A=VAL(
"&H"+A$):C=C+A:POKEX,A:X=X+1:NEX
T:IFC<>C(I)THENCLS3:LOCATE8,12:P
RINT"TYPING ERROR IN LINE";N+10*
I:ENDELSEC=0:NEXT:RETURN
710 LOCATE0,8:FORJ=1TO11:PRINTST
RING$(40,32);:NEXT:LOCATE0,8:RET
URN
720 CLS5:LOCATE2,8:PRINT"INSERT
THE FILE DISK IN YOUR DRIVE.":LO
CATE10,10:PRINT"THEN PRESS ENTER
."
730 K$=INKEY$:IFK$=""THEN730ELSE
RETURN
800 DATA 2,27,49,0,0,0,0,0,0,0
810 DATA 5,27,42,1,192,3,0,0,0,0
820 DATA 2,27,50,0,0,0,0,0,0,0
900 DATA 5,27,49,27,108,7,0,0,0,
0
910 DATA 5,27,42,6,128,2,0,0,0,0
920 DATA 5,27,50,27,108,0,0,0,0,
0
```

Turn of the Screw

On CoCo Time

by Tony DiStefano
Contributing Editor

Those who designed the Color Computer built it to be cost- effective but with enough options to make it a good computer. It's a good little computer. In fact, it's a great little computer. I use the term "little" because it is just that — little. To me, this doesn't mean "inadequate"; it means the CoCo is a computer possible of future expansion. You can add things to it and watch it grow. I have added many things to my CoCos over the years. One of the things I added to the CoCo was a disk drive controller. This controller had room to expand (it was "little," too). I have designed several gadgets that go into the controller and are useful in everyday computing.

Most of the gadgets I have designed for my own computer, I have made available to others through my own company, Disto. When I thought a gadget was good enough for others to use, I gave that design to another company to produce. I did this because I haven't had the time and money required to put a product on the market. The company I chose to market my designs is CRC, Inc., a small local computer dealer that works mainly with Radio Shack computers. Christian Rochon and I struck a deal that is mutually beneficial. I design products to further expand the CoCo computer and he produces and markets them. Our original agreement still stands. Many people haven't understood the relationship between Disto and CRC, and I wanted to pass the word along. With this bit of background out of the way, let's move ahead in time — the subject for the next few installments of "Turn of the Screw."

Tony DiStefano is a well-known early specialist in computer hardware projects. He lives in Laval Ouest, Quebec. Tony's username on Delphi is DISTO.

A Timely Idea

One gadget the CoCo has lacked is the ability to track real time. What is real time? Look at your watch. That is real time. Real time means the time of the present, and it changes every second. Real time includes seconds, minutes, hours, days, months and years.

The CoCo has built-in timekeeping capabilities, derived from the vertical sync of the video signal. Vertical sync occurs 60 times-per-second, and every time the signal occurs the CPU receives a hardware interrupt via the *IRQ line. (The asterisk indicates this line is *active low*, meaning the indicated action occurs when the line is Low.) In the Extended BASIC ROMs, the *IRQ service routine increases a value in memory — the memory location used is increased 60 times each second. This location, then, represents time. But not real time; just relative time. BASIC provides a TIMER function, but it is used only to keep track of elapsed time — the time between two events. This function does not give you the real time unless you have a small program to do so. Also, you would have to set the real time every time you turned your computer on. Under OS-9, real time is easier to deal with, but you must still set the clock every time you boot OS-9. My solution to this problem is to install the equivalent of a digital watch into the CoCo.

What components make a digital watch? It's really quite simple. There is a display. The display shows the various parts of time. Driving the display are transistors and logic counters, usually all contained in one chip. Then there are miscellaneous parts, such as a battery, switches and a crystal. Combine all of these parts and you have a watch.

Since we're working with a computer, we already have a display. And software allows us a way of emulating the switches. Therefore, those will not be needed either. We do, however, need a way of allowing the CPU access to the time counters. And the battery and crystal are needed. So what do we need? A chip that has CPU memory-map capabilities, and a battery and crystal.

While there are many such chips on the market, the real-time clock chip I chose is the OKI MSM6242B. This is a CMOS device in an 18-pin DIP package. Figure 1 shows the pinout of the MSM6242B chip, which from this point I will refer to as the RTC. The RTC looks very much like other memory-mapped devices. It has address and data lines and the usual read/write chip-enable stuff that memory chips have. In fact, the RTC is controlled in the same way as a memory device — you read from and write to it using the load and store assembler commands or PEEK and POKE in BASIC. Following is a pin-by-pin description of this RTC:

Pins 14, 13, 12 and 11 — D0, D1, D2 and

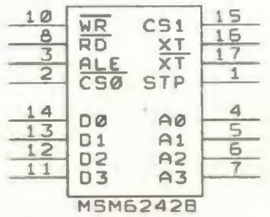


Figure 1

D3, respectively. This is the data bus. These are bidirectional lines on which the CPU (CoCo) transfers data to and from the RTC clock/calendar registers and control registers. As with the CoCo, D0 is the Least Significant Bit (LSB) and D3 is the Most Significant Bit (MSB).

Pins 4, 5, 6 and 7 — A0, A1, A2 and A3, respectively. This is the address bus. These input-only pins are used to decode and select the 16 resisters available in the RTC.

Pin 3 — Address Latch Enable (ALE), which is used to strobe the address into the address line. For those of you who do not know, the ALE line is used mainly for Intel products, such as the 8085. This CPU has common data and address pins, which are demultiplexed by the ALE output. In this case the data pins and the address pins are tied together and the ALE strobes the address information from the data pins into the RTC. If the RTC is connected to a CPU that does not have an ALE line, the ALE pin is tied to V_{dd}.

Pin 10—the *WR signal. This input-only pin signals the RTC that you are writing data into the RTC. With the selection of CS1 and *CS0, data is written into the selected register at the rising edge of *WR.

Pin 2—the *CS0 signal. This active-low Chip Select is used to memory-map the RTC registers into some I/O area of the host CPU.

Pin 15 — the CS1 signal. This special Chip Select is connected to a power-failure detection circuit. It is designed to deselect at a lower voltage than all other pins. Therefore, when power is removed from the chip, it deselects (shutting down) before any erroneous writes can occur.

Pin 1 — the STD.P. This open-collector output is the standard interrupt. It is software-controlled and is used to signal the CPU when a predetermined amount of time has passed.

Pins 16 and 17 — XT and *XT. These are the crystal inputs used to increment the time counters. The frequency of the crystal is 32.768 KHz.

Pins 18 and 9 — V_{dd} and GND, respectively. The range for this device is from +2 volts to +6 volts. Ground is always 0 volts.

Figure 1 shows four address lines. These four address lines represent 16 address locations in the memory map. Within these locations are the data registers and control registers that make up the RTC. The following is a description of each register and how many bits are needed. Note that not all four bits are used in some registers. When a bit is not used, it is ignored during a write operation and held at Logic 0 (or Low) during a read. The address locations of the RTC are relative to where it is memory mapped (the base address). The 16 registers of the RTC are found at locations that run from $0 to $F (15 in decimal). The real CPU address for a given register is determined by adding the register address to the base address.

All address locations referred to here are Register addresses only. Register 0 is indicated when A0 to A3 are all Low (zeros). The A0 line represents the least significant digit of the address.

Register 0 — S1, the 1-second digit register. All 4 data bits are used and have a count value from zero to nine.

Register 1 — S10, the 10-second digit register. Only D0 to D2 are used for a count value of zero to five.

Register 2 — MI1, the 1-minute digit register. This requires all 4 data bits, with a count value of zero to nine.

Register 3 — MI10, the 10-minute digit register. Only D0 to D2 are used for a count value of zero to five.

Register 4 — H1, the 1-hour digit register. It allows a count value from zero to nine, and all four bits are required.

Register 5 — the first multi-use register is called H10. D0 and D1 are the 10-hour digit register. The count value is zero to two. D2 is the PM/AM indicator. D3 is not used.

Register 6 — D1, the 1-day value. All four bits are needed to hold a value from zero to nine.

Register 7 — D10, the 10-day digit register. Only D0 and D1 are used to form a value from zero to three.

Register 8 — MO1, the 1-month digit. This register uses all four bits and has a value from zero to nine.

Register 9 — MO10, the smallest of all registers. The 10-month digit only uses one digit, D0, with zero and one as the only valid count values.

Register A (Hex) or 10 (Dec) — holds the 1-year digit and is called Y1. Four data bits are required for a count value of zero to nine.

Register B or 11 — Y10, contains the 10-year digit. Its four data bit entry allows zero to nine count values.

Register C or 12 — the W Register and contains the day of the week. Only three data lines (D0, D1, D2) are required to have a value of zero to six. This matches the seven days of the week. Value 0 is Sunday, 1 is Monday, etc.

Registers D, E, and F hold no time information, but are control registers that help the RTC do its job. Register D uses all four data bits. D0 is the Hold bit. Setting this bit to 1 stops the counting of the clock, which allows the host CPU to read the contents of the clock without the contents changing. There is no time lost as long as the halt is less than one second. D1 is the Busy bit. After setting the Hold bit to 1, the host CPU must read the Busy bit. If it is not busy, then reading or writing to the RCT is permitted. If it is busy, the host CPU must release Hold and try again. The busy condition exists when the time and date counters are in the middle of changing when the Hold bit is set. This condition only happens once every second and lasts only a fraction of a second. D2 is the IRQ Flag. This bit corresponds to the output level of the STD.P output. The IRQ flag indicates the occurrence of an interrupt. Finally, D3 is the 30-second Adjust. The S1 and S10 registers are read-only. To adjust the seconds, you must write a 1 in this bit. 125 us later, D3 automatically returns to 0. When D3 is set to 1, the seconds are reset to 0. If the number of seconds is greater than 30, the minutes value is increased by 1.

Register E is located at Address 14. It deals mostly with the interrupts. D0 is the Mask bit. When this bit is 1, the IRQ is disabled. D1 is the INTRPT/STND. Don't ask me what this stands for; the data manual is unclear as to the function of this bit. Although its meaning is unclear, its function is simple. A zero in this bit makes the STD.P output pulse at 7.8 ms. A one in this bit makes the output a complete cycle long. D2 and D3 are T0 and T1, respectively, and create the length of a cycle. Values and timings are as follows:

T1	T0	Period
0	0	1/60 second
0	1	1 second
1	0	1 minute
1	1	1 hour

Register F is the last register, and it uses all four bits. D0 is the Reset bit. A one in this bit clears all counters in the RTC. A zero releases all counters. If CS0* is zero, then a reset is done automatically. D1 is the Stop bit. Writing a 1 into this bit stops all counting. This is used when setting the time. A zero in this bit operates the counters normally. D2 is the 24/12 hour mode. A 1 in this bit selects the 24-hour mode, and a 0 selects the 12-hour mode. The 24/12-hour mode can only be changed when the Reset bit is High. D3 is the Test bit. Writing a 1 to this bit causes the chip to go into the Test mode, in which the Stop and Reset bits are ignored. The Test mode clocks the seconds at 5416.3 Hz instead of 1 Hz. Writing a 0 to this bit restores normal operations.

That's just about all the information we need on the particulars of this RTC. Next time, I'll show you how you can connect this RTC to the CoCo.

Barden's Buffer

EZ Assembler Guidebook

by William Barden, Jr.
Contributing Editor

I presented in the April 1991 issue a simple assembler program that allows you to enter up to 46 lines of assembly-language code on the Hi-Res screen of your CoCo 3. Last month we covered the instructions supported by the mini-assembler. *EZASM* translates the source code entered on the screen into binary machine code that is then saved in a machine-language file loadable with Disk BASIC's LOADM command, or callable by Extended Color BASIC. This assembler is available free. Write to me at P.O. Box 3568, Mission Viejo, CA 92692. Send an SASE disk mailer with a formatted 35-track disk, or see the April and May issues of THE RAINBOW. This month I continue with assembler topics in general and how they relate to *EZASM*.

Steps in Writing an Assembly Program

As previously mentioned, writing an assembly-language program is much more difficult than BASIC, but the reward is speed — up to hundreds of times faster than interpretive BASIC and dozens of times faster than OS-9's BASIC09. Just what are the steps in writing an assembly-language program?

The first step is to design the program. Put down on paper what you want the program to do. This might involve flowcharting the program, or listing a step-by-step sequence of instructions, including any program loops.

Next, create the source code for the program by entering the assembly-language listing. That's where *EZASM* comes in. This program provides a simple way to enter short assembly-language programs. Assemble the program (by pressing F1 within *EZASM*) to translate the listing (mnemonics and labels) into machine language (ones and zeros). You'll need some kind of guide as to what the individual instructions do. There's plenty of material on this in the pages of THE RAINBOW. Also, get the official Motorola guide to the instruction set of the 6809 (contact Motorola, Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721). You may also be able to locate an out-of-print book I wrote for Radio Shack called *Color Computer Assembly Language Programming*, which provides a tutorial geared to the CoCo. Unfortunately, I have only one file copy at this point and can't supply any copies.

During the development of your machine-language program, you'll probably encounter assembly errors. These are indicated by short messages on the lines where the errors occur. It may be necessary to reassemble several times to get a clean listing. Simply overwrite the text, add new lines, or delete bad lines in the process. Here's a recap of *EZASM*'s keys:

Up Arrow — moves the cursor up one line at a time.

Down Arrow — moves the cursor down one line at a time.

Left Arrow — moves the cursor one character to the left.

Right Arrow — moves the cursor to the right one character.

CLEAR — clears the screen and the assembled data when pressed *twice*.

SHIFT-Up Arrow — deletes the line at the cursor's current position.

SHIFT-Down Arrow — inserts a blank line at the cursor's position.

F1 — assembles the current text and saves the source and object code.

F2 — reads in a previously written source input file.

When you have a clean listing with no assembly errors, the result is an object file on disk that can be loaded into memory using the Disk BASIC LOADM command. Load the program, either from BASIC or as a stand-alone program, and try it. Chances are it won't work properly the first time, and you'll have to rethink some of the code. Change the source code as required, and reassemble to generate a new loadable file. Keep at it until the program works. It may be necessary to edit, assemble and debug several times to get a working program.

The debugging steps are probably the most painful process in assembly-language programming. *EDTASM+* has a very interactive debugger. You can set breakpoints throughout the program to see if you've successfully reached certain sections and then investigate registers and memory locations. With *EZASM*, you don't have the interactive debugging capabilities of *EDTASM+*. But you can accomplish similar things with careful planning, even though it may take a little longer to do so.

The 6809 Instruction Set and *EZASM*

The 6809 instruction set consists of about 60 different instructions that perform rudimentary operations. There are several registers within the 6809 that are essentially internal memory locations, addressable by letter. The A and B registers are general

Bill Barden has written 35 books and hundreds of magazine articles about small computers . His newest Color Computer project, Connecting the CoCo to the Real World, *is a book of CoCo interfacing projects. He has over 20 years experience in the industry on systems ranging from mainframes to micros.*

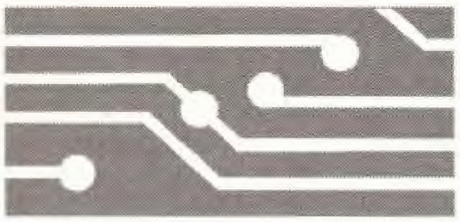
eight-bit (one byte) registers that hold intermediate results for many instructions. The D register is the combination of A and B, and it holds a total of 16 bits (two bytes).

The X and Y registers are two 16-bit registers used in indexed addressing modes. Indexed addressing modes are typically used to gain access to tables of data. The X or Y register is set with a "base" value that points to the beginning of the table. Different elements within the table can then be read or written by adding displacement values to the base value. For example, a table at symbolic location T could be put in the Y register and the 33rd byte of the table loaded into the A register as follows:

```
LDY #T
LDA +032,X
```

The first instruction loads the table address into Register Y. The second line loads the 33rd byte in the table into Register A.

The U and S registers are 16-bit *stack registers*. A stack is an area of memory set aside to store data retrievable in first-in, last-out order. The S register is the main stack register that points to a typical assembly-language stack used to store return addresses for subroutine calls (similar to a BASIC GOSUB). The U register points to an auxiliary stack that can be used as a second stack storage area.



The PC, or program counter register points to the next instruction to be executed. The location value represented in the left-most column is actually held in the PC register. Each machine-language instruction is read one byte at a time until the entire instruction is processed. At the end of the processing, the PC register points to the next instruction in the sequence.

Most 6809 instructions operate using the A or B register to hold the result. The instruction may clear the A or B register (CLRA, CLRB), increment the A or B register (INCA, INCB), or shift the contents of A or B (ROLA, ROLB, LSRA, LSRB, LSLA, LSLB, ASLA, ASLB, ASRA, ASRB). Many instructions operate with a value in the A or B register and a second operand in memory. The ADDA instruction, for example, adds the contents of the A register to the contents of a memory location and puts the result in the A register. The SUBB instruction subtracts the contents of a memory location from the value in Register B and puts the result back into B. There are instructions to add, subtract and compare A or B with the contents of a memory location (ADD, ADC, SUB, SBC, CMP).

Other instructions operate with a second operand contained within the machine code of the instruction itself, a so-called *immediate operand*. Immediate addressing is used to conveniently add, subtract or compare the contents of A or B with a constant value.

EZASM allows most 6809 instructions. However, such seldom-used instructions as SYNC, SWI, SWI2, SWI3 and RTI are deleted from the *EZASM* repertoire. They relate to system functions that are for advanced operations. The remaining instructions may generally use the more popular addressing modes. Often, a second operand is held in a memory location. The memory location in *EZASM* has a label of A through Z. This label is converted to an absolute address by *EZASM* during assembly. Such an addressing mode is called *extended addressing*. This addressing mode, in a subtraction situation, would appear as follows:

```
SUBB G
```

This instruction tells the computer to subtract the contents of memory location pointed to by G from the value in Register B.

Another variation on the assembler representation is specifying the address by a known location. If the constant or variable at G is a value of 123 at Location $7FEE, extended addressing using the known location could be used instead:

```
SUBB $7FEE
```

If the value 123 is simply a constant value, an immediate addressing mode could be used instead:

```
SUBB #123
```

Jumps and branches accomplish the same actions as GOTOs and IF/THEN GOTO in BASIC; they transfer control to another instruction in a different location in the program. The path of execution is therefore altered. There are two types of jumps or branches. *Unconditional jumps* (JMP, LBRA) *always* transfer control to the specified location. Here are two forms of unconditional jumps:

JMP G (go to Location G)

JMP $7FEE (go to Location $7FEE)

LBRA G (go to Location G)

Most instructions affect the Condition Codes register, which is a set of bits taken together as a group that tells the 6809 whether the result of a prior operation was zero, negative, overflow, or a carry, among other things. *Conditional jumps* or branches transfer control only if a stated condition is met (e.g. if the value in Register A or B is zero or negative.) The condition tested is the result of a former operation, usually the last instruction executed. You may want to add 10 numbers until a count reached zero. Starting with 10 in the count, you'd add a number and decrement the count by one.

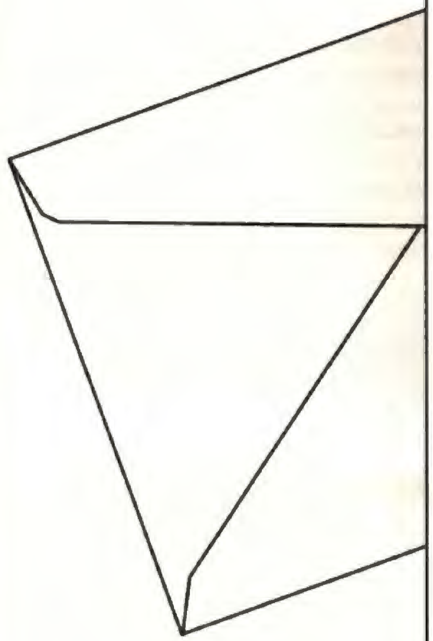
ACE

American Computing Equipment
P.O. Box 39281
Louisville, KY 40233
(502) 459-7966
1-800-346-2794
24 Hours. VISA/MC/Check
Shipping: US $3, Canada $5, Other Countries $7
C.O.D. add $4
KY add 6% sales tax

NEW! Graphics Oriented Assembly Language

A brand new language from a brand new Color Computer supporter. GOAL (Graphics Oriented Assembly Language) is modeled after 68000 and VAX-11 assembly language to make programming in assembly easier.

- **Programs are assembled into memory-saving object code**
- **Low-level principles mean GOAL programs execute fast**
- **Over 170 instructions with 25 special graphics instructions**
- **Multitasking feature allows a BASIC program and many GOAL programs to run simultaneously**
- **17 powerful memory-saving addressing modes**
- **100+ pages documentation, sample programs, editor, assembler, interpreter, and fonts included**
- **CoCo 3 version requires at least 128K, disk drive**
- **CoCo 2 version requires 64K, disk drive**
- ***Introductory Offer: $29.00***
- **GOAL Icons Disk w/300 characters: $24.00**
- **GOAL Fonts Disk w/36 fonts: $19.00**



The Zero condition code would be set only when the count reached zero. An `LBNE` (Long Branch on Not Equal to Zero) would allow the program to return to the beginning of the Add loop until the count reached zero, at which time the `LBNE` would "fall through" and not cause a transfer of control. Here are some typical conditional branches in *EZASM* form:

`LBNE G` (go if result not equal to zero)

`LBEQ H` (go if result equal to zero)

`LBLE $7FEE` (go if result < or = to zero

Jumps and branches, when taken, directly affect the contents of the PC register. The new address is forced into the PC register, causing the 6809 to start executing the instruction found at the new location. For subroutines, a `JSR` (Jump to Subroutine) saves the return address of the instruction following the `JSR` in the memory stack. An `RTS` (Return from Subroutine) pops the return address, forcing it into the PC register to effect the return.

CoCo 3 Disk

Listing 1: `SORTDRV`

```
100 'SORT SCREEN DRIVER
110 CLEAR 1000,&H7EFF
120 WIDTH 32
130 SCREEN 0,0
140 CLS
150 FOR I=0 TO 510: PRINT CHR$(R
ND(63)+32);: NEXT
160 LOADM "SORT.BIN"
170 DEFUSR0=&H7F00
180 A=USR0(0)
190 GOTO 190
```

Assembly-language programs are long lists of simple operations. They tend to be much longer than equivalent BASIC programs. One of the problems with assembly language is that everything has to be broken down into working with 8- or 16-bit quantities. There are no built-in instructions to hold floating-point data, such as the mixed number 12345.678. The integer and fractional parts of such a number have to be held in separate variables in assembly language, or a complicated floating-point software package has to be designed.

On the other side of the coin, however, is the fact that many operations can easily be handled in 8 or 16 bits. Eight bits can hold values from 00000000 through 11111111 (decimal 0 through decimal 255). Sixteen bits can hold values from 0000000000000000 through 1111111111111111 (decimal 0 through 65,535). By stringing together a few such 8- and 16-bit variables, much larger quantities can be handled. Four bytes, or 32 bits, can represent quantities up to 4,294,967,296, which should be enough to handle most household budgets directly!

Interfacing to *EZASM* Output Code

When you assemble a program using *EZASM*, the assembler portion compiles the machine-language code for the assembly-language source in memory. It then writes the code in a binary file. This file can then be loaded into memory at any time using the Disk BASIC `LOADM` command. The format is as follows:

`LOADM"`*filename*`"`

where *filename* is the same name you used for the object output file when you started *EZASM*. Once the file is loaded, you can use an `EXEC` command to execute the program just loaded. Normally, routines executed this way are complete programs. You can also execute a short assembly-language code segment using Extended Color BASIC. To do so, write your BASIC program to protect the memory (using `CLEAR`) where the file will be loaded, load the file (with `LOADM`), define the transfer address (using `DEFUSR`), and then make the transfer (`USR`*n*). If your program is working properly, this process calls the assembly-language program, executes it, and returns to the BASIC program. Last month I provided an assembly-language program to clear the low-resolution screen. Another assembly-language example is shown in Listing 2. It's a short program to sort data on the Low-Res screen. The BASIC version takes over an hour, but this assembly version takes only 10 seconds! Listing 1 shows the BASIC calling sequence, a prototype for setting up your own assembly-language calls. ❑

Listing 2: `SORT.ASM`

```
A LDX   #$400
  LDY   #0
B LDA   +000,X
  LEAX  +001,X
  CMPA  +000,X
  LBLS  C
  LDB   +000,X
  STB   -001,X
  STA   +000,X
  LDY   #1
C CMPX  #$5FF
  LBNE  B
  CMPY  #0
  LBNE  A
  RTS
```

CoCo Consultations

Terminal Programs

by Marty Goodman
Contributing Editor

Q *I need a Disk BASIC-based terminal program that can download files bigger than the 42K buffer* Mikeyterm *provides. Can you recommend one?*

Tom Disch (BASCO)
Brookfield, Wisconsin

A If you use a Color Computer 1 or 2, your only choice is *Greg-E-Term V2.5*, a shareware program available on Delphi. The program permits direct-to-disk downloads so you can download any size file up to the capacity of your disk drive. This allows you to download files of at least 150K. *Greg-E-Term* may support larger than 35-track single-sided drives, but I'm not certain of this. *Greg-E-Term*'s implementation of direct-to-disk downloading is flawed since it compromises the speed of the download by turning the drive on and off to save each sector, but it is reliable. If you use a 512K CoCo 3, you have a wide choice of terminal programs that receive files larger than 42K in a number of different ways. My favorite is *V-Term*, which boasts a software character font that makes by far the clearest and most readable display of all Disk BASIC-based terminal programs. *V-Term* allows you to receive files to a 490K RAM disk and then permits you to save the files in the RAM disk to 40- or 80-track drives if you use *ADOS 3*. This allows you to save files of up to 360K. *V-Term* also has a direct-to-disk download option. Best of all, the *V-Term* implementation of the direct-to-disk download is relatively well done and quite fast. *V-Term* is available as a commercial product for about $35. See ads in THE RAINBOW for further information. *Delphiterm* and *Ultimaterm* are shareware programs available on Delphi that also support downloading larger files. Both offer direct-to-disk downloading.

Martin H. Goodman, M.D., a physician trained in anesthesiology, is a longtime electronics tinkerer and outspoken commentator — sort of the Howard Cosell of the CoCo world. On Delphi, Marty is the SIGop of RAINBOW's CoCo SIG and database manager of OS-9 Online. His non-computer passions include running, mountaineering and outdoor photography. Marty lives in San Pablo, California.

DC Modem Pak

Q *I've been trying, without success, to use my DC Modem Pak with my disk drive controller and a Y cable. When I have both the DC Modem Pak and the disk controller hooked to the Y cable, my CoCo 3 refuses to recognize the disk controller software and boots with the Extended Color BASIC sign-on message instead of "Disk Extended Color BASIC." Can you help correct this problem?*

Joel Kuntze (JOELKUNZE)
Laurel, Maryland

A The problem is that Disk Extended Color BASIC is in a ROM inside the disk controller and is addressed to the same location as the ROM that has the terminal program software for the DC Modem Pak. Since you'll want to use the Modem Pak's ACIA and modem with a disk-based terminal program rather than the one inside the Modem Pak's ROM, the best solution is to disable that Modem Pak ROM. This is easily done. Locate the ROM chip on the Modem Pak board. This is the 28-pin 2364 chip, which likely has the Tandy part number TCC1009A on it. It is located adjacent to the two small-scale logic chips on the Modem Pak board (the 74LS133 and the 74LS04). Be sure you are not looking at the physically similar 6551 ACIA chip, which is also a 28-pin chip. Carefully cut Pin 20 of the 2364 chip where it is soldered to the board, and bend up from the chip the stub of that pin. Solder a wire to that stub. Solder the other end of that wire to Pin 28 of the 2364 chip. You have disconnected the chip select line of the ROM and tied it to +5 volts, thus forcing the chip to always be disabled. Now the Modem Pak will work properly on a Y cable (with your disk controller) as a 300-bps modem. Note: You may be interested to learn that I designed a kit now sold by CoCo PRO! that allows owners of Tandy DC Modem Paks to convert them into general purpose RS-232 Paks that can communicate from 300 through 19,200 bps. Once converted, your DC Modem Pak behaves like a Tandy RS-232 Pak. Of course, you'll need an external modem to use with the converted DC Modem Pak since the conversion involves cutting off and discarding the 300-bps modem that came with the original DC Modem Pak.

Tandy 2000 Power Supply

Q *I bought a Tandy 2000 case and power supply to use in repackaging my Color Computer 3 system. But I find the power supply for the Tandy 2000 does not work when hooked to the CoCo 3. Does it in some way require the Tandy 2000 motherboard in order to work?*

Steve Flock (THEJAILER)
Vancouver, Washington

A The power supply for the Tandy 2000, like those in most PC-compatibles, is a switching power supply. Switching power supplies require some degree of load on them before they operate. Indeed, they can be damaged if you attempt to operate them without enough load. Some of the more modern switchers have some ballast or a load resistor built-in so you never need to worry about this. But the older ones, such as the one used in the Tandy 2000, may not. Try this: Buy a couple of 10-ohm, 10-watt resistors (sold at Radio Shack stores). Hook one of them across the 5-volt output of the Tandy 2000 power supply. Now measure the voltage. Do you get 5 volts now on this line? Do you now get all the other voltages you are supposed to get? Fine. When you repackage the computer, include that resistor in the circuit. Be careful to mount it and place it in a way that the heat it generates does not hurt other components. If one 10-ohm, 10-watt resistor is not sufficient to make the supply wake up and operate, try using two in parallel — even three or four. Note that four of them in parallel present a resistance of 2.5 ohms, which draws a grand total of 2 amps from the Tandy 2000 power supply's 5-volt line. That 5-volt supply can, I believe, deliver eight or more amps on that line. Most PC-compatible switching power supplies that require such a load are designed so they require it only on the 5-volt line. If this fix makes the 5-volt line function, but the +12 and -12 volt lines do not function, try putting loads on them. I doubt you'll have to do that, though.

High-Speed Pokes

Q *Is there a simple poke that can fix Disk Extended BASIC so it works properly at the 2-MHz CPU speed?*

Steve Ricketts (STEVEPDX)
Boring, Oregon

A I'm sorry, but the problems that cause Disk Extended BASIC's unreliability (it tends to crash disks!) if used to save files while the computer is running at the 2-MHz CPU speed cannot be fixed with a simple poke. About a half dozen parts of the code need to be appropriately patched. Art Flexser, after extensive research, tells me that all occurrences of the `EXG A,A` instruction followed by a second `EXG A,A` instruction in the disk sector read and disk format part of the Disk BASIC ROM need to be patched for a longer delay. Much to Art's dismay, even substituting one "Push the entire stack" instruction followed by one "Pull the entire stack" instruction for those `EXG` instructions did not give reliable operation — an actual patch to a delay loop is needed. In addition, there is at least one other delay used in the code (the "long" delay that loads the 16-bit X register with zero then decrements it to zero) that must be further lengthened to produce reliable operation. My advice for anyone wanting to accomplish Disk I/O with the computer running at 2 MHz is to buy *ADOS 3* or *Extended ADOS 3*, which has all the required patches and has been shown to be reliable. However, Art Flexser cautions that you really do not get faster throughput when using the disk drive with the CPU running at 2 MHz because the speed tends to be limited by the speed of disk transfer and not the speed of the CPU. Therefore, in most cases, if you do not use *ADOS 3* and need to run a BASIC program at 2 MHz, be sure to slow the computer to 1 MHz for disk operation and speed it up again after the disk operation is completed.

Keyboard Extender Cables

Q *How far can I extend the 15 wires of the Color Computer's keyboard? Do I need any special circuitry (terminator resistors or such), or can I just extend them via a ribbon cable without any special tricks?*

Phillip Brown (THEFERRET)
Berkeley, California

A No added circuitry is needed to extend the Color Computer keyboard via a ribbon cable. The only "trick" to it is obtaining or making the proper connectors to hook up everything. Edge connectors that fit the socket on the CoCo motherboard and normally accept the mylar ribbon cable from the keyboard itself are hard to find, as are those female connectors to plug onto the mylar cable. That's why I made my keyboard extender cable as a commercial product for the CoCo 1, 2 and 3. This product, sold by Microcom and soon by CoCo PRO!, allows extension of the CoCo keyboard six feet from the computer. It has all the required adaptors and connectors for a solderless installation. I've extended the keyboard up to about 15 feet without obvious problems, but frankly I advise you to keep the cable as short as possible — preferably six feet or less.

Printers and PC-Compatibles

Q *I'm trying to hook a Tandy DMP-130 printer to a PC-compatible using a standard PC-compatible printer cable. The printer just locks up and doesn't work. Can you help me fix this problem?*

Rex Falkenberg (REXFALK)
Hillsboro, Texas

A Open the shell of the 36-pin connector and cut the wire that goes to Pin 33. The problem is that Tandy used Pin 33 for the printer *INIT line, but on all other printers that is a grounded line. If you want to get really fancy, remove the wire from Pin 31 of that connector and solder it to Pin 33 thereby hooking up the *INIT line for your DMP-130. But frankly I have found no PC-compatible software that actually uses this line. Be sure to label your modified cable as "For Use With DMP-130 Printer Only!" Note, this fix may be required for other printers in the Tandy DMP-130 line, such as the 130A, 132 and 133

X-Pad Resurrection

Q *I have a friend who wants to use a Tandy Color Computer X Pad with his CoCo 3. Does the X Pad require a source of +12 volts?*

Wayne Montague (MILLWAY)
Mississauga, Ontario
Canada

A The Tandy X Pad is unique among devices made by Tandy for the CoCo. It requires sources of both +12 volts and -12 volts. Your friend must either use a Multi-Pak Interface or arrange to provide both of these extra voltages by some other means. These voltages were originally supplied by the CoCo 1 on Pins 1 (-12) and 2 (+12) of the system bus. But in the CoCo 2 and 3, these two pins are not connected. In the Multi-Pak, the required +12 and -12 voltages *are* supplied on those pins.

Disk-Drive Cable Extender

Q *How long of a disk drive cable can I use? Can I use one of about five feet?*

Leonard Litberg (RADICAL)
Fox Lake, Illinois

A You will have no problems with a 5-foot cable. I've used cables up to 12 feet long with no problems, but I don't really recommend others try this — they may not be as lucky as I was. The best rule is to keep the cable between the controller and disk drives as short as possible, but any cable under eight feet should work reliably.

Your technical questions are welcomed. Please address them to CoCo Consultations, THE RAINBOW, P. O. Box 385, Prospect, KY 40059.

We reserve the right to publish only questions of general interest and to edit for brevity and clarity. Due to the large volume of mail we receive, we are unable to answer letters individually.

Questions can also be sent to Marty through the Delphi CoCo SIG. From the CoCo SIG> prompt, pick Rainbow Magazine Services. Then at the RAINBOW> prompt , type ASK (for Ask the Experts) to arrive at the EXPERTS> prompt , where you can select the "CoCo Consultations" online form, which has complete instructions.

Make OR Break

by Dennis Hoin

$

Join the CoCo in a mad race to get ahead

Have you ever wanted to have *more* than the Joneses? Well now's your chance with *Make or Break*, a money-strategy game that works on any Color Computer with 32K or more. The objective is to be the only player with any cash left to his name. You have several chances to make more dough, but there are plenty of risks, and you'll need a little luck in order to win. *Make or Break* allows from two to four players, and CoCo can play along, too.

Getting Started

Make Or Break uses the 51-column screen program presented years ago in "Byte Master" (THE RAINBOW, February 1985). If you already have this program, it is not necessary for you to enter Listing 1. If you don't have the original, enter Listing 1 and save it to tape or disk. Now run the program.

Dennis Hoin is employed by Pennsylvania Power and Light. He has been writing programs for his Color Computer for approximately eight years, and he works with other computers as well.

If everything is correct, the CoCo should be in a 51-column mode. Save the machine-language 51-column driver with

```
(C)SAVEM"51",&H7C22,&H7FFF,&H7CC2
```

Now enter the main program, `MAKEBRAK`, shown in Listing 2. When run, this program first checks to see if you are in the 51-column mode. If not, it automatically loads the 51-column driver and executes it. As written, `MAKEBRAK` looks for this driver on a disk. If you are using a tape-based system, simply change `LOADM` in Line 40 to `CLOADM`. When you have an error-free listing, you are ready to play.

Let the Game Begin

To begin play, first enter the number of players and their names. If you want a computer-generated player, enter `COMP` as the player's name. You can have more than one computer-generated player in each game, but remember that all computer-generated players' names must begin with COMP. For example, you might name one player COMP1 and another COMPUTER.

32K Disk (or Cassette Modification)

60	55
120	202
180	223
240	68
300	66
360	29
420	199
480	116
END	94

Listing 1: `SCREEN51`

```
1 'MAKE OR BREAK
2 'WRITTEN BY CHRIS BONE
3 'COPYRIGHT JUNE 1991
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 'LISTING FOR 51 CHARACTER
  SCREEN
20 CLEAR301,&H7CC1:PMODE4,1
30 X=31933
40 READ A:IF A=999 THEN 50 ELSE
POKE X,A:X=X+1:GOTO40
50 PMODE4,1:CLS:EXEC&H7CC2:CLS:S
CREEN1,0:PRINT"YOU ARE NOW IN SC
REEN 51 MODE"
60 PRINT"TO SAVE TO DISK <SAVEM'
'51'',&H7CC2,&H7FFF,&H7CC2>":END
70 DATA82,69,69,78,255,142,125,8
8,252,1,104,191,1,104,253,125,18
3,142,124
80 DATA225,220,169,159,169,253,1
24,248,158,186,191,127,252,127,1
27,250,57,129,158
90 DATA38,21,52,22,158,186,191,1
27,252,204,255,255,237,129,156,1
83,38,250,53
100 DATA22,126,170,26,129,135,38
,5,115,127,255,32,244,129,64,39,
9,129,32
110 DATA39,236,127,127,255,32,23
1,125,127,255,39,226,127,127,255
,52,22,134,32
120 DATA23,0,154,189,179,228,16,
131,4,200,16,36,55,32,16,131,0,5
0,35
130 DATA8,131,0,51,124,127,255,3
2,242,134,5,61,52,4,196,7,247,12
7,250
140 DATA182,127,255,53,4,84,84,8
4,211,186,253,127,252,127,127,25
5,53,22,134
```

During play, *Make or Break* rolls the dice and moves your game piece around the board for you. If the square on which you land is not owned by another player, you are shown the name of the square, its cost, and given the option to either purchase the square or pass. If the square on which you land is owned by another player, you are shown the owner's name and the amount you owe that player for temporarily using his square. This amount is based on the height (in floors) of any building currently erected on the square, and it is automatically withdrawn from your cash balance. If you already own a square on which you land, you are given the option to either add another story to the existing structure or pass.

There are three free spaces on the board. If your game piece lands on one of these, you lose no money and you gain no money. In addition, there are four bonus/bad luck squares: Lucky Day, General Store, Pay Day and the Bonus Pot. Landing on Lucky Day earns you anywhere from $100 to $200, free and clear. If you land on General Store, it'll *cost* you from $100 to $200. All players receive $400 each time they make it past another Pay Day. The Bonus Pot starts at $200 and increases by $75 every time a player passes the Bonus Pot square. Any player who lands on this square receives the cash currently in the pot, and the Bonus Pot starts at $200 again.

When you run out of money, you are removed from the game and your squares are freed for sale to the remaining players. Play continues until there is only one player with a cash balance. Good luck!

❑

```
150 DATA32,32,159,13,111,16,38,Ø
,88,52,22,129,8,38,19,23,Ø,253,1
34
160 DATA32,141,78,134,32,141,74,
23,Ø,242,23,Ø,239,32,54,129,12,3
8,19
170 DATA158,186,191,127,252,127,
127,25Ø,2Ø4,255,255,237,129,156,
183,38,25Ø,32,31
180 DATA141,41,19Ø,127,252,156,1
83,37,22,48,137,255,Ø,191,127,25
2,158,186,236
190 DATA137,1,Ø,237,129,188,127,
252,38,245,32,214,134,95,141,8,2
3,Ø,176
2ØØ DATA53,15Ø,126,2Ø4,28,129,13
,38,17,134,32,141,248,23,Ø,16Ø,1
27,127,253
21Ø DATA124,127,252,127,127,25Ø,
57,198,4,247,127,254,128,32,61,1
95,126,14Ø,31
22Ø DATA2,19Ø,127,252,166,16Ø,52
,2,138,15,141,22,53,2,72,72,72,7
2,138
23Ø DATA15,141,12,122,127,254,38
,233,48,137,255,Ø,141,66,57,246,
127,25Ø,92
24Ø DATA247,127,251,52,2,2Ø4,248
,Ø,138,8,253,127,248,53,2,198,25
5,122,127
25Ø DATA251,39,14,26,1,7Ø,86,28,
254,118,127,248,118,127,249,32,2
37,52,6
26Ø DATA252,127,248,17Ø,132,234,
1,167,132,231,1,53,6,164,132,228
,1,237,132
27Ø DATA48,136,32,57,246,127,25Ø
,2Ø3,5,193,7,35,4,48,1,192,8,193
,7
28Ø DATA38,15,3Ø,16,193,31,38,7,
76,95,3Ø,1,95,32,2,3Ø,1,247,127
29Ø DATA25Ø,191,127,252,57,52,4,
246,127,25Ø,19Ø,127,252,192,5,42
,2Ø,2Ø3,8
3ØØ DATA48,31,3Ø,16,193,255,38,8
,198,31,3Ø,16,198,2,32,2,3Ø,16,1
91
31Ø DATA127,252,247,127,25Ø,53,1
32,255,255,255,255,221,221,223,2
23,85,255,255,255
32Ø DATA153,9,9,159,177,123,2Ø9,
191,243,45,18Ø,2Ø7,181,91,37,175
,219,255,255
33Ø DATA255,219,119,123,223,189,
238,237,191,246,144,15Ø,255,251,
177,187,255,255,255
34Ø DATA157,191,255,24Ø,255,255,
255,255,253,223,253,219,183,127,
15Ø,64,38,159,217
35Ø DATA221,221,143,15Ø,237,183,
15,15Ø,233,23Ø,159,217,8Ø,221,22
3,7,3Ø,23Ø,159
36Ø DATA219,113,1Ø2,159,14,237,1
83,127,15Ø,1Ø5,1Ø2,159,15Ø,1Ø4,2
37,191,253,223,221
37Ø DATA255,253,223,221,251,237,
183,189,239,255,15,15,255,123,22
2,219,127,15Ø,237
38Ø DATA191,191,15Ø,66,71,143,15
Ø,96,1Ø2,111,53,81,1Ø2,31,15Ø,11
9,118,159,26
39Ø DATA17Ø,17Ø,31,7,113,119,15,
7,113,119,127,15Ø,116,1Ø2,159,1Ø
2,96,1Ø2,111
4ØØ DATA141,221,221,143,2Ø6,238,
23Ø,159,1Ø1,51,53,111,119,119,11
9,15,96,6,1Ø2
41Ø DATA111,98,36,68,111,15Ø,1Ø2
,1Ø2,159,22,97,119,127,15Ø,1Ø2,3
7,175,22,97
42Ø DATA53,111,15Ø,121,23Ø,159,1
3,221,221,223,1Ø2,1Ø2,1Ø2,159,1Ø
2,1Ø2,1Ø5,159,1Ø2
43Ø DATA1Ø2,Ø,111,1Ø2,153,15Ø,11
1,1Ø2,96,221,223,14,2Ø1,55,15,13
9,187,187,143
44Ø DATA247,123,189,223,29,221,2
21,31,181,95,255,255,255,255,255
,15,189,255,255
45Ø DATA255,255,3Ø,134,143,119,1
13,1Ø2,31,255,135,119,143,238,23
2,1Ø2,143,255,15Ø
46Ø DATA7,159,218,177,187,191,24
9,1Ø2,142,143,119,22,1Ø2,111,223
,157,221,143,239
47Ø DATA238,23Ø,159,247,1ØØ,22,1
11,59,187,187,31,255,96,1Ø2,111,
255,22,1Ø2,111
48Ø DATA255,15Ø,1Ø2,159,241,1Ø2,
23,127,248,1Ø2,142,239,255,22,11
9,127,255,7,14
49Ø DATA15,187,27,187,191,255,1Ø
2,1Ø2,159,255,1Ø2,1Ø5,159,255,1Ø
2,96,111,255,1Ø5
5ØØ DATA15Ø,111,246,1Ø2,142,159,
255,14,216,15,248,Ø,Ø,Ø,37,Ø,Ø,Ø
,69,999
```

7Ø	149	57Ø	71	115Ø	9Ø
16Ø	222	64Ø	14	124Ø	222
26Ø	219	71Ø	214	132Ø	191
32Ø	61	79Ø	128	14ØØ	85
38Ø	35	84Ø	8Ø	146Ø	215
43Ø	75	91Ø	2Ø3	152Ø	82
51Ø	47	1Ø4Ø	9	END	67

Listing 2: MAKEBRAK

```
1 'MAKE OR BREAK
2 'WRITTEN BY DENNIS HOIN
3 'COPYRIGHT JUNE 1991
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
1Ø '******LISTING FOR MAKE OR BR
EAK****
2Ø '********LOAD 51 CHR.SCREEN &
 DIM******
3Ø IF PEEK(&H7CC2)<>142 THEN 4Ø
ELSE 7Ø
4Ø CLEAR1,&H7CC1:CLEAR3ØØ:LOADM"
51":EXEC&H7CC2
5Ø PMODE4,1:PCLS1:SCREEN1,Ø
6Ø CLS
7Ø DIM PP(318),P1(318),P2(318),P
3(318),BB(318),A$(26),B$(26),C(2
6),D1(17),D2(17),D3(17),D4(17),N
$(32),NN$(32),N(32),OW(32),O(32)
8Ø M(1)=1:M(2)=18Ø:M(3)=1:M(4)=1
8Ø:M(5)=1:M(6)=18Ø:M(7)=1:M(8)=1
8Ø:P$="#####.##"
9Ø FOR X=1TO4:P(X)=1:PAY(X)=15ØØ
:NEXT
1ØØ PL$(Ø)="None":POT=2ØØ
11Ø COLORØ
12Ø GOSUB 69Ø
13Ø '*****************GET NUM.OF
PLAYERS & NAMES***
14Ø GOSUB123Ø:PRINT@368,"HOW MAN
Y"::PRINT@419,"PLAYERS"::GOSUB13
2Ø:INPUTZZ:PRINT@419,"";
```

Yes! They're still available

Rainbow Back Issues

BACK ISSUES STILL AVAILABLE

Have you explored the wealth of information in our past issues? From our very first, four-page issue to many with more than 300 pages of material, it's all just for CoCo users — a great way to expand your library!

A WORLD OF INFO AT A BARGAIN PRICE

All back issues sell for the single issue cover price. In addition, there is a $3.50 charge for the first issue, plus 50 cents for each additional issue for postage and handling if sent by United Parcel Service. There is a $5 charge for the first issue, plus a $1 charge for each additional issue on orders sent by U.S. Mail. UPS will not deliver to a post office box or to another country.

MOST ISSUES STILL AVAILABLE

Available issues through June 1982 are provided on white paper in a reprint form. All others are in regular magazine form. VISA, MasterCard and American Express accepted. Kentucky residents please add 6 percent sales tax; Canadian residents, 7 percent GST. In order to hold down costs, we do not bill, and no C.O.D. orders are accepted.

Due to heavy demand, we suggest you order the back issues you want now while supplies last. To order, review and fill out the form below and mail it with your payment.

For greater convenience, order through the Rainbow Magazine Services area of our Delphi CoCo SIG.

RAINBOW INDEX

A complete index for, July 1981 through June 1984, is printed in the July 1984 issue. Separate copies are available for $2.50 plus 50¢ handling. Indexes for subsequent years are published annually in the July issues of THE RAINBOW.

TOTAL	______
KY RESIDENTS ADD 6%	______
CANADIAN RESIDENTS ADD 7% GST	______
U.S. MAIL CHARGE	______
SHIPPING & HANDLING	______
U.P.S. CHARGE	______
TOTAL AMOUNT	______
ENCLOSED	______

Article Reprints

In instances where a given issue is now out of print and not available for purchase, we do provide photocopies of specific articles. The cost for this service is $1.50 plus 50 cents S/H per article. This service is provided *only* in the case of out-of-stock issues.

Name ______________________

Address ______________________

City __________ State ______ Zip ______

❑ Payment Enclosed, or

Charge to my: ❑ VISA ❑ MC ❑ AE

CARD # ______________________

EXPIRATION DATE __________ PHONE ( ) ______

SIGNATURE ______________________

TO ORDER BY PHONE (credit card orders only) call (800) 847-0309, 9 a.m. to 5 p.m. EST. All other inquiries call (502) 228-4492.

send to: THE RAINBOW, The Falsoft Building, P.O. Box 385, Prospect, KY 40059

Please send me the following back issues:

	VOLUME 1		
JUL 81	Premier Issue	$2.00	❑
AUG 81		$2.00	❑
NOV 81		$2.00	❑
FEB 82		$2.00	❑
	VOLUME 2		
JUN 83	Printers	$2.95	❑
	VOLUME 3		
AUG 83	Games	$2.95	❑
SEP 83	Education	$2.95	❑
OCT 83	Graphics	$3.95	❑
MAR 84	Business	$3.95	❑
APR 84	Gaming	$3.95	❑
MAY 84	Printer	$3.95	❑
JUN 84	Music	$3.95	❑
JUL 84	Anniversary	$3.95	❑
	VOLUME 4		
AUG 84	Games	$3.95	❑
SEP 84	Education	$3.95	❑
OCT 84	Graphics	$3.95	❑
NOV 84	Data Comm.	$3.95	❑
DEC 84	Holiday	$3.95	❑
JAN 85	Beginners	$3.95	❑
FEB 85	Utilities	$3.95	❑
MAR 85	Business	$3.95	❑
APR 85	Simulations	$3.95	❑
MAY 85	Printer	$3.95	❑
JUN 85	Music	$3.95	❑
JUL 85	Anniversary	$3.95	❑
	VOLUME 5		
AUG 85	Games	$3.95	❑
SEP 85	Education	$3.95	❑
OCT 85	Graphics	$3.95	❑
NOV 85	Data Comm.	$3.95	❑
DEC 85	Holiday	$3.95	❑
JAN 86	Beginners	$3.95	❑
FEB 86	Utilities	$3.95	❑
MAR 86	Business	$3.95	❑
APR 86	Home Help	$3.95	❑
MAY 86	Printer	$3.95	❑
JUN 86	Music	$3.95	❑
JUL 86	Anniversary	$3.95	❑
	VOLUME 6		
AUG 86	Games	$3.95	❑
SEP 86	Education	$3.95	❑
OCT 86	Graphics	$3.95	❑
NOV 86	Data Comm.	$3.95	❑
DEC 86	Holiday	$3.95	❑
JAN 87	Beginners	$3.95	❑
FEB 87	Utilities	$3.95	❑
MAR 87	Business	$3.95	❑
APR 87	Home Help	$3.95	❑
MAY 87	Printer	$3.95	❑
JUN 87	Music	$3.95	❑
JUL 87	Anniversary	$3.95	❑
	VOLUME 7		
AUG 87	Games	$3.95	❑
SEP 87	Education	$3.95	❑
OCT 87	Graphics	$3.95	❑
NOV 87	Data Comm.	$3.95	❑
DEC 87	Holiday	$3.95	❑
JAN 88	Beginners	$3.95	❑
FEB 88	Utilities	$3.95	❑
MAR 88	Business	$3.95	❑
APR 88	Home Help	$3.95	❑
MAY 88	Printer	$3.95	❑
JUN 88	Music	$3.95	❑
JUL 88	Anniversary	$3.95	❑
	VOLUME 8		
AUG 88	Games	$3.95	❑
SEP 88	Education	$3.95	❑
OCT 88	Graphics	$3.95	❑
NOV 88	Data Comm.	$3.95	❑
DEC 88	Holiday	$3.95	❑
JAN 89	Beginners	$3.95	❑
FEB 89	Home Help	$3.95	❑
MAR 89	Hardware	$3.95	❑
APR 89	Business	$3.95	❑
MAY 89	Printer	$3.95	❑
JUN 89	Summer Fun	$3.95	❑
JUL 89	Anniversary	$3.95	❑
	VOLUME 9		
AUG 89	Beyond BASIC	$3.95	❑
SEP 89	Education	$3.95	❑
OCT 89	Graphics	$3.95	❑
NOV 89	Data Comm.	$3.95	❑
DEC 89	Holiday	$3.95	❑
JAN 90	Beginners	$3.95	❑
FEB 90	Home Help	$3.95	❑
MAR 90	Hardware	$3.95	❑
APR 90	Business	$3.95	❑
MAY 90	Printer	$3.95	❑
JUN 90	Summer Fun	$3.95	❑
JUL 90	Anniversary	$3.95	❑
	VOLUME 10		
AUG 90	0S-9	$3.95	❑
SEP 90	Education	$3.95	❑
OCT 90	Graphics	$3.95	❑
NOV 90	Data Comm.	$3.95	❑
DEC 90	Holiday	$3.95	❑
JAN 91	Beginners	$3.95	❑
FEB 91	Home Help	$3.95	❑
MAR 91	Hardware	$3.95	❑
APR 91	Music	$3.95	❑
MAY 91	Printer	$3.95	❑
JUN 91	Summer Fun	$3.95	❑

```
15Ø IF ZZ<Ø OR ZZ>4 THEN GOTO14Ø
 ELSE GOSUB 122Ø
16Ø FOR X=1TOZZ:GOSUB123Ø:PRINT@
368,"ENTER NAME";:PRINT@419,"OF
PLAYER";X
17Ø PRINT@47Ø," ";:GOSUB132Ø:INP
UTPL$(X):NEXT:GOSUB122Ø
18Ø '************PLAY GAME******
******************
19Ø FOR T=1TO ZZ
2ØØ IF PAY(T)=Ø THEN 44Ø
21Ø PRINT@726,STRING$(27,"*");
22Ø GOSUB134Ø:PRINT@729,"IT'S YO
UR TURN ";PL$(T);: PRINT@386,"Pl
ayer ";T;:PRINT@485,"You Have";:
PRINT@536,"$";:PRINTUSING P$;PAY
(T);
23Ø LINE(Ø,Ø)-(256,191),PSET,B
24Ø GOSUB122Ø:GOSUB132Ø:GOSUB83Ø
25Ø PRINT@1114,"$";POT;
26Ø IF T=1 THEN TT=1 ELSE IF T=2
 THEN TT=3 ELSE IF T=3 THEN TT=5
 ELSE IF T=4 THEN TT=7
27Ø GOSUB45Ø
28Ø IF P(T)=8 THEN GOSUB141Ø:GOT
O38Ø
29Ø GOSUB123Ø:PRINT@ 368,N$(P(T)
);:PRINT@419,NN$(P(T));:PRINT@47
Ø,"$";:PRINTUSING P$;N(P(T));:PR
INT@521,"OWNER ";PL$(OW(P(T)));
3ØØ IF O(P(T))>Ø THEN SOUND 1ØØ,
1:PRINT@623,O(P(T));" STORIES";
31Ø IF OW(P(T))>Ø AND OW(P(T))<>
T AND O(P(T))<4THEN N=N(P(T))*(O
(P(T))*1Ø)/1ØØ:GOSUB136Ø:GOSUB13
7Ø:GOSUB14ØØ:GOSUB134Ø:GOSUB135Ø
:GOTO38Ø
32Ø IF OW(P(T))>Ø AND OW(P(T))<>
T AND O(P(T))>3 THEN N=N(P(T))*(
O(P(T))*1Ø)/1ØØ+1ØØ*O(P(T)):GOSU
B136Ø:GOSUB137Ø:GOSUB 14ØØ:GOSUB
134Ø:GOSUB135Ø:GOTO38Ø
33Ø IF P(T)=24 THEN PAY(T)=PAY(T
)+POT:POT=2ØØ:SOUND23Ø,5:GOSUB13
5Ø:GOTO38Ø
34Ø IF P(T)=1 OR P(T)=6 OR  P(T)
=19 OR P(T)=24 OR P(T)=26 THEN G
OSUB134Ø:GOSUB135Ø:GOTO38Ø
35Ø IF P(T)=17 THEN GOSUB 142Ø:G
OSUB135Ø:GOTO38Ø
36Ø GOSUB 134Ø:PRINT@383,"1=Buy"
;:PRINT@434,"2=Build";:PRINT@485
,"Space=Pass";:PRINT@536,"$";:PR
INTUSING P$;PAY(T);:GOSUB139Ø
37Ø IF MID$(PL$(T),1,4)="COMP" T
HEN GOTO 148Ø
38Ø IF MID$(PL$(T),1,4)="COMP" T
HEN A$=" ":GOTO 39Ø ELSE A$=INKE
Y$:IF A$="" THEN 38Ø
39Ø IF P(T)=1 OR P(T)=6 OR P(T)=
8 ORP(T)=19 OR P(T)=17 ORP(T)=19
 OR P(T)=24 OR P(T)=26 THEN GOTO
43Ø
4ØØ IF A$="1"AND O(P(T))=Ø AND P
AY(T)>N(P(T)) THEN O(P(T))=1:OW(
P(T))=T:PAY(T)=PAY(T)-N(P(T)):PL
AY"T2ØL2ØØO3;1;5;12":PRINT@588,"
+++BUY+++";:GOTO43Ø
41Ø IF A$="2" AND OW(P(T))=T AND
 PAY(T)>N(P(T)) THEN PAY(T)=PAY(
T)-N(P(T)):O(P(T))=O(P(T))+1:PLA
Y"T2ØØL2ØØO4;1;5;12":PRINT@588,"
###BUILD###";:GOTO43Ø
42Ø GOSUB139Ø
43Ø IF D3=1 THEN SOUND 2ØØ,3:PLA
Y"P8":SOUND2ØØ,3:GOTO2ØØ
44Ø NEXT T:GOTO19Ø
45Ø '***********************MOVE
PIECES*********
46Ø FOR X=1 TO D1
47Ø IF OW(P(T))<>T THEN GOSUB 13
8Ø
48Ø IF OW(P(T))=T THEN ON T GOSU
B125Ø,127Ø,129Ø,131Ø
49Ø IF M(TT)<36 THEN M(TT+1)=M(T
T+1)-15:IF T=1THEN M(TT)=1 ELSE
IF T=2 THEN M(TT)=13 ELSE IF T=3
 THEN M(TT)=24 ELSE IF T=4 THEN
M(TT)=35
5ØØ IF M(TT+1)>141 AND M(TT)<5Ø
AND M(TT+1)<179 THEN M(TT+1)=128
51Ø IF M(TT+1)=38 AND M(TT)<36 T
HEN M(TT)=16:M(TT+1)=1
52Ø IF M(TT+1)<36 THEN M(TT)=M(T
T)+2Ø:IF T=1 THEN M(TT+1)=1 ELSE
 IF T=2 THEN M(TT+1)=13 ELSE IF
T=3 THEN M(TT+1)=24 ELSE IF T=4
THEN M(TT+1)=35
53Ø IF M(TT)=56 THEN M(TT)=54
54Ø IF M(TT)>2Ø5 AND M(TT+1)<36
THEN M(TT+1)=23
55Ø IF M(TT)>2Ø6 THEN M(TT+1)=M(
TT+1)+15:IF T=1 THEN M(TT)=2Ø7 E
LSE IF T=2 THEN M(TT)=218 ELSE I
F T=3 THEN M(TT)=229 ELSE IF T=4
 THEN M(TT)=24Ø
56Ø IF M(TT+1)>141 AND M(TT)>2Ø6
 THEN M(TT)=226
57Ø IF M(TT+1)>141 AND M(TT)>5Ø
THEN M(TT)=M(TT)-2Ø:IF T=1 THEN
M(TT+1)=142 ELSE IF T=2 THEN M(T
T+1)=153 ELSE IF T=3 THEN M(TT+1
)=164 ELSE IF T=4 THEN M(TT+1)=1
75
58Ø IFM(TT)=186 THEN M(TT)=193
59Ø IF M(TT)<5Ø AND M(TT+1)>141
THEN M(TT+1)=18Ø:IFT=1 THEN M(TT
)=1 ELSE IF T=2 THEN M(TT)=13 EL
SE IF T=3 THEN M(TT)=24 ELSE IF
T=4 THEN M(TT)=35
6ØØ IF P(T)=1 AND PAY(T)>Ø THEN
PAY(T)=PAY(T)+4ØØ:PLAY"T2ØØL2ØØO
5;1;3;6;9"
61Ø IF P(T)=24 THEN POT=POT+75
62Ø P(T)=P(T)+1:IF P(T)=33 THEN
P(T)=1
63Ø ON T GOSUB 124Ø,126Ø,128Ø,13
ØØ
64Ø PLAY"T2ØØ;L2ØØ;O1;5;O3;5;O5;
5"
65Ø NEXTX
66Ø RETURN
67Ø CLS'********************GET
STATEMENTS******
68Ø LINE(5Ø,5Ø)-(1ØØ,1ØØ),PSET,B
:RETURN
69Ø CLS:PRINT@367,"   PAY":PRINT
@418,"   DAY":PRINT@469,"  $4ØØ"
:GOSUB68Ø:GET(5Ø,5Ø)-(1ØØ,1ØØ),P
P,G
7ØØ CLS:PRINT@367,"  GENERAL";:P
RINT@418,"   STORE":GOSUB68Ø:GET
(5Ø,5Ø)-(1ØØ,1ØØ),P2,G
71Ø CLS:PRINT@367,"   BONUS":PRI
NT@418,"   POT":GOSUB68Ø:GET(5Ø,
5Ø)-(1ØØ,1ØØ),P3,G
72Ø CLS:PRINT@367,"  LUCKY":PRIN
T@418,"   DAY":GOSUB68Ø:GET(5Ø,5
Ø)-(1ØØ,1ØØ),P1,G
73Ø PRINT@52,"1":LINE(2,6)-(12,1
6),PSET,B:GET(2,6)-(12,16),D1,G
74Ø PRINT@52,"2":LINE(2,6)-(12,1
6),PSET,B:GET(2,6)-(12,16),D2,G
75Ø PRINT@52,"3":LINE(2,6)-(12,1
6),PSET,B:GET(2,6)-(12,16),D3,G
76Ø PRINT@52,"4":LINE(2,6)-(12,1
6),PSET,B:GET(2,6)-(12,16),D4,G
77Ø '*****************DATA******
****************
78Ø GOSUB143Ø
79Ø DATA PAY,DAY,4ØØ,Skid,Road,2
ØØ,City,Dump,22Ø,Train,Station,3
ØØ,Creek,Side,25Ø,FREE,SPACE,Ø,R
iver,View,275,LUCKY,DAY,,Park,St
reet,3ØØ,Burbon,Ave.,32Ø,Apple,B
lvd.,33Ø,Power,House,3ØØ
8ØØ DATA Subway,Terminal,3ØØ,Gra
nd,View,35Ø,Mountain,Side,375,La
ke,Side,39Ø,GENERAL,STORE,,Magno
la,Drive,4ØØ,FREE,SPACE,Ø,Rosewo
od,Ave.,44Ø,Bus,Station,3ØØ,Colo
nial,Park,47Ø,Ranch,Drive,49Ø,BO
NUS,POT,,Thunderbird,Terace,5ØØ,
FREE,SPACE,Ø
81Ø DATA Bently,Drive,52Ø,Hudson
,Blud.,54Ø,Airline,Terminal,3ØØ,
Platinum,Place,56Ø,Silver,Lake,5
8Ø,Diamond,Aces,6ØØ
82Ø GOTO88Ø
83Ø '***************** ROLE DIE**
*************
84Ø COLORØ:LINE(8Ø,127)-(154,137
),PSET,BF:DRAW"C1BM52,127E5R1ØG5
BM64,127E5R1ØG5E5D1ØG5":FOR Z=1T
O RND(5)+5:LINE(52,127)-(62,137)
,PSET,BF:LINE(64,127)-(74,137),P
SET,BF:D1=Ø:Y=Ø
85Ø FORX=54TO66STEP12:A=RND(6):Y
=Y+1:D(Y)=A:D2=7-A:D1=D1+D2:B=A>
1:C=A>3:D=A=6:E=A/2=INT(A/2):PSE
T(X,128,-D):PSET(X+6,128,-C):PSE
T(X,132,-B):PSET(X+3,132,1-E):PS
ET(X+6,132,-B):PSET(X,136,-C):PS
ET(X+6,136,-D):NEXTX:EXEC43345
86Ø NEXTZ:PRINT@833,"="D1;:IF D(
1)=D(2)THEN D3=1:PRINT" DOUBLES"
; ELSE D3=Ø
87Ø RETURN
88Ø CLS'***************DRAW BOARD
***************
89Ø PRINT@Ø,""
9ØØ LINE(Ø,Ø)-(256,191),PSET,B
91Ø LINE(5Ø,5Ø)-(2Ø6,141),PSET,B
F
92Ø LINE(Ø,Ø)-(5Ø,5Ø),PSET,B
93Ø LINE(2Ø6,141)-(256,191),PSET
,B
94Ø LINE(Ø,141)-(5Ø,191),PSET,B
95Ø LINE(2Ø6,5Ø)-(256,Ø),PSET,B
96Ø LINE(Ø,5Ø)-(5Ø,65),PSET,B
97Ø LINE(2Ø6,5Ø)-(256,65),PSET,B
98Ø LINE(Ø,65)-(5Ø,8Ø),PSET,B
99Ø LINE(2Ø6,65)-(256,8Ø),PSET,B
1ØØØ LINE(Ø,8Ø)-(5Ø,95),PSET,B
1Ø1Ø LINE(2Ø6,8Ø)-(256,95),PSET,
B
1Ø2Ø LINE(Ø,95)-(5Ø,11Ø),PSET,B
1Ø3Ø LINE(2Ø6,95)-(256,11Ø),PSET
,B
1Ø4Ø LINE(Ø,11Ø)-(5Ø,125),PSET,B
1Ø5Ø LINE(2Ø6,11Ø)-(256,125),PSE
T,B
1Ø6Ø LINE(7Ø,Ø)-(9Ø,5Ø),PSET,B
1Ø7Ø LINE(7Ø,141)-(9Ø,191),PSET,
B
1Ø8Ø LINE(11Ø,Ø)-(13Ø,5Ø),PSET,B
1Ø9Ø LINE(11Ø,141)-(13Ø,191),PSE
T,B
11ØØ LINE(15Ø,Ø)-(17Ø,5Ø),PSET,B
111Ø LINE(15Ø,141)-(17Ø,191),PSE
T,B
```

```
1120 LINE(190,Ø)-(190,5Ø),PSET,B
1130 LINE(190,141)-(190,191),PSE
T
1140 PUT(Ø,141)-(5Ø,191),PP,PSET
1150 PUT(205,Ø)-(255,5Ø),P2,PSET
1160 PUT(205,141)-(255,191),P3,P
SET
1170 PUT(Ø,Ø)-(5Ø,5Ø),P1,PSET
1180 X=Ø
1190 EXEC43345:X=X+1:READ N$(X),
NN$(X),N(X):IF X=32 THEN 12ØØ EL
SE 119Ø
12ØØ GOTO14Ø
121Ø '*****************GOSUB STA
TEMENTS***********
122Ø LINE(52,52)-(124,11Ø),PSET,
BF:COLORØ:RETURN
123Ø COLOR1:LINE(52,52)-(12Ø,11Ø
),PSET,BF:COLORØ:RETURN
124Ø PUT(M(1),M(2))-(M(1)+1Ø,M(2
)+1Ø),D1,PSET:RETURN
125Ø PUT(M(1),M(2))-(M(1)+1Ø,M(2
)+1Ø),D1,PRESET:RETURN
126Ø PUT(M(3),M(4))-(M(3)+1Ø,M(4
)+1Ø),D2,PSET:RETURN
127Ø PUT(M(3),M(4))-(M(3)+1Ø,M(4
)+1Ø),D2,PRESET:RETURN
128Ø PUT(M(5),M(6))-(M(5)+1Ø,M(6
)+1Ø),D3,PSET:RETURN
129Ø PUT(M(5),M(6))-(M(5)+1Ø,M(6
)+1Ø),D3,PRESET:RETURN
13ØØ PUT(M(7),M(8))-(M(7)+1Ø,M(8
)+1Ø),D4,PSET:RETURN
131Ø PUT(M(7),M(8))-(M(7)+1Ø,M(8
)+1Ø),D4,PRESET:RETURN
132Ø PLAY"T1ØØL1ØØO3;1;12;1":RET
URN
133Ø LINE(125,52)-(2Ø4,11Ø),PSET
,BF:RETURN
134Ø COLOR1:LINE(125,52)-(2Ø4,11
Ø),PSET,BF:COLORØ:RETURN
135Ø GOSUB139Ø:PRINT@638,"PRESS
ANY KEY";:RETURN
136Ø PRINT@521,"OWNER ";PL$(OW(P
(T)));:RETURN
137Ø PRINT@572,"RENT$";:PRINTUSI
NG"####.##";N;:RETURN
138Ø COLOR1:LINE(M(TT),M(TT+1))-
(M(TT)+1Ø,M(TT+1)+1Ø),PSET,BF:RE
TURN
139Ø PLAY"T2ØØL2ØØO5;1;12":RETUR
N
14ØØ PAY(T)=PAY(T)-N:PAY(OW(P(T)
))=PAY(OW(P(T)))+N:IF PAY(T)<=Ø
THEN 15ØØ ELSE RETURN
141Ø N=RND(2ØØ):PAY(T)=PAY(T)+N:
GOSUB134Ø:PRINT@383,"YOU WIN $";
N;:GOSUB 135Ø:RETURN
142Ø N=RND(2ØØ):PAY(T)=PAY(T)-N:
GOSUB134Ø:PRINT@383,"YOU PAY $";
N;:IF PAY(T)=<ØTHENGOTO15ØØELSE
GOSUB135Ø:RETURN
143Ø CLS:PMODE2,1:SCREEN1,Ø:PRIN
T@61,"Make":PRINT@113,"or":PRINT
@163,"Break"
144Ø COLORØ:FOR X=1TO9Ø: EXEC433
45:LINE(X,12)-(255-X,62),PSET,B:
NEXT
145Ø PRINT@5ØØ,"By  Dennis";:PRI
NT@557,"Hoin";
146Ø LINE(Ø,Ø)-(255,19Ø),PSET,B:
FOR X=1TO9ØØ:NEXT
147Ø CLS:PMODE4,1:SCREEN1,Ø:COLO
RØ,1:RETURN
148Ø IF PAY(T)-N(P(T))>6ØØ AND O
(P(T))=Ø THEN A$="1":PRINT@588,"
+++BUY+++";:GOTO39Ø ELSE IF PAY(
T)-N(P(T))>6ØØ THEN A$="2": PRIN
T@588,"###BUILD###";:GOTO39Ø ELS
E A$=" ":PRINT@588,"%%%PASS%%%";
:GOTO 39Ø
149Ø '************* CHECK IF GAME
 OVER********
15ØØ PAY(T)=Ø:FOR X=1TO32:IF OW(
X)=T THEN OW(X)=Ø:O(X)=Ø
151Ø NEXT
152Ø N=Ø:GOSUB122Ø:GOSUB133Ø:GOS
UB123Ø:GOSUB134Ø:PRINT@419,"   S
ORRY";:PRINT@433,"  ";PL$(T);:PR
INT@47Ø,"    YOUR";:PRINT@484,"
 OUT";:SOUND1,5Ø:SOUND1ØØ,1:FOR
X= 1TO ZZ:IF PAY(X)>Ø THEN N=N+1
153Ø NEXT X
154Ø IF N>1 THEN D1=32:GOSUB46Ø:
GOSUB138Ø:RETURN  ELSE CLS:PMODE
2,1:SCREEN1,Ø: FOR X=1 TO ZZ:IF
PAY(X)>Ø THEN  SOUND 2ØØ,1Ø:PRIN
T@61,"WINNER ";:PRINT@163,PL$(X)
 ELSE NEXT X
155Ø FOR X=1TO 2ØØ STEP3:SOUNDX,
1:SOUND 21Ø-X,1:NEXT:PMODE4,1:SC
REEN1,Ø:END
156Ø '*************************
157Ø '***   MAKE OR BREAK     ***
158Ø '***   BY DENNY HOIN     ***
159Ø '***      Ø2/27/85       ***
16ØØ '*************************
```

JWT Enterprises

Attention! MM1 programmers wanted. Many prospective projects already planned. Write for more information.

Optimize Utility Set 1 *NEW!*

Optimize your disks by eliminating fragmented files and compacting your directories for faster file access. Also includes a utility to assess file fragmentation and directory fragmentation as well as excess directory padding.

Optimize Utility Set 1 $29.95 Foreign postage, add $3.00

Nine-Times: Each issue contains: 9 helpful and useful programs to help build your OS-9 library • Instructions, examples, and samples of Basic09 procedures and subroutines to help with your own programs and your understanding of Basic09 • C programs and programming examples • Hints, Help columns, and informative articles to advance your knowledge of OS-9 • Supplied totally of 5.25" disk • Bound manual sent to each new subscriber for help in getting *Nine-Times* up and running, as well as tips on using it with a ram disk or hard disk • All graphic/joystick interface for ease of use.

1-Year Subs, $34.95 Canadian postage, add $1.00
Article/programming submissions wanted Foreign postage, add $7.00

Back Issues: Available for the *May1989* through *March 1991* issues. Please write for information on Back Issue contents.

Back Issue, ea. $7.00 Foreign postage, add $1.50 ea.

Magazine Source: Due to many inquiries, the source code for the magazine graphic shell is being provided as an informational tool. Included is the actual Basic09 source code and compiled modules on disk, as well as documentation and a printed copy of the source code.

Source, $24.95 Foreign postage, add $3.00

To order, please send U.S. check or money order to:

JWT Enterprises
5755 Lockwood Blvd.
Youngstown, OH 44512
All products for Level 2 OS-9

Technical Assistance: (216)-758-7694

RAINBOW CERTIFICATION SEAL

Sorry, no C.O.D.'s or credit cards; Foreign & Canadian orders, *please* use **U.S. money orders.** U.S. checks, allow 3-4 weeks for receipt of first issue/back issue.

Copyright (C) 1991 OS-9 is a trademark of Microware Systems Corp. and Motorola, Inc.

GRANITE COMPUTER SYSTEMS

MODEMS

NEW 9600 Baud V.32/V.42/V.42bis data modems. MNP1-5LAPM.
Error Correction and data compression (much higher effective throughput — as much as 38400 Baud). External — call for price

NEW PRICE 2400 Baud V.42/V.42bis data modems. MNP1-5/LAPM.
Error correction and data compression (much higher effective throughput — as much as 9600 Baud). External $189 (+$7 S&H).

NEW PRODUCT 9600 Baud Send/Receive Fax modems.
Send/Receive text/graphics files from/to your computer/any Fax machine in the world. Full 2400 Baud data modem capabilitiy.
Includes software. External $149/Internal $139 (+$5 S&H)

NEW PRICE 9600 Baud SendFax modems.
Send text/graphics files from your computer to any Fax machine in the world. Full 2400 Baud data modem capability. Includes software. External $129/Internal $119 (+$5 S&H)

NEW PRICE 2400 Baud Data modems External $109/Internal $99 (+$5 S&H)

These are all high quality modems made by Zoom Telephonics in the USA. Fully Hayes compatible. Two year mfg warranty. Software available.

S&H Canada (Air PP and Ins) V.42/v.42bis $12.00 Sendfax/Data $8.00

GCS FILE TRANSFER UTILITIES — Version 3.0

The GCS File Transfer Utilities provide a simple and quick method to transfer text/binary files from/to a variety of floppy disk formats.

Commands PC, RS, FLEX disks: Dir, Dump, Read, Write
PC disks: Rename, Delete, Format

Handles most 5.25 and 3.5 formats. Any level sub-directories (PC). Binary files. Use pipes for multiple file transfers. Multi-Vue version can be used under Multi-Vue or as stand alone Shell commands.

Requires OS-9 L2 for COCO 3, L1 for COCO 1 or 2. 2 drives (one can be hard/ramdisk, one floppy 40 T DD DS). Multi-Vue for Multi-Vue version. SDISK3 for COCO3 - SDISK for COCO 1 or 2. Multi-Vue version $54.95 Standard version $44.95
V3.0 updates (provide disk number) $15.00

D.P. Johnson Software SDISK or SDISK3 $29.95 L1+L2 Utils $75.00
Shipping and handlling — any software $2.00

Orders must be prepaid or COD. VISA/MC accepted. COD is additional.

571 Center Road, Hillsboro, NH 03244 USA
(603) 464-3850

OS-9 is a trademark of Microware Systems Corporation and Motorola, Inc.
MS-DOS is a trademark of Microsoft Corp. FLEX is a trademark of TSC, Inc.

Graphics CoCo 1, 2 & 3

Super Comics+

E.Z. Friendly has released *Super Comics+*, an upgrade to its *Comics+* program for the CoCo 2 and 3. (*Comics +* was reviewed in THE RAINBOW, February 1990, Page 106.) As the previous review thoroughly covered the original program, the focus of this review highlights the additions in the new version. I have not seen the original program, so my comparison information comes from the previous review.

Super Comics+ is specifically designed for the CoCo 2 and 3. It now takes advantage of the CoCo 3's extra memory by supporting a RAM disk (you must provide the RAM disk software). The program is disk-based and still requires a spring-centered joystick. E.Z. Friendly warns against using a mouse. I find this very disappointing since I work with various graphics programs for the Macintosh and IBM clones, and I have become attached to using the mouse. I think it's a shame that many programs for the CoCo, especially graphics programs, still ignore the mouse.

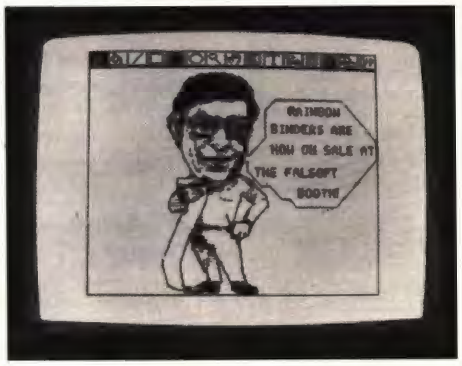
Printer support for *Super Comics+* has been expanded to include the Tandy DMP-130 series. Only two printer rates are supported: 600 and 2400. The manual indicates that if your printer is in IBM mode, you must edit the C+ loader program and insert a command to switch your printer to the Tandy mode. The manual is not very clear on this process. There are no remark statements in the program to indicate where the command should go. This type of change would require a call to E.Z. Friendly. I suggest these instructions be more clear. Also, E.Z. Friendly needs to put their address and a support phone number in the manual. If you do not have a Tandy printer you should call E.Z. Friendly and ask if other printers can also be used.

Super Comics+ has added a `COPIER` program to create a backup on a new disk, leaving out the various utilities, so you can keep the program and your creations on one disk. If you use the CoCo 3 with a RAM disk, you will not need this feature since the RAM disk holds the entire program.

This updated program contains 15 icons: Pencil (free-hand draw), Paint Can (black and white fills), Line, Box, Filled Box, Circle, Magnifying Glass (zoom edit), Hand (move object), Eraser, Text (enter text), Clear Screen, Undo, Disk (save/load), Printer and Clip Art. The Pencil icon works just as in the previous version, with one addition: Pressing W allows precision control in a 64-by-64 pixel window. The manual suggests you draw with a spring-centered joystick. I find this extremely difficult. The W function gives you an opportunity to use free-floating control, which I find easier in a small area. The Erase function can also use the W function (window) for fine detail.

The Paint, Line, Box, Circle and Magnifying Glass icons all function the same as in the earlier version. The Filled-Box function has been changed to an inverted box. The Hand icon still allows you to "get and put" parts of your drawing. You can still stamp with it, but a few nifty features have been added. You can cut a section of the drawing and move the section to the top of the screen. The cut-out section of the drawing disappears. It is now temporarily loaded in memory and is pasted when you press ENTER. The cut portion stays in memory while using other program functions, with exceptions being magnify and print. If you double click on the hand, a Pattern Cut and Stamp menu appears. There are 20 available patterns — from bricks to bar-code lines. Select the pattern you want, then stamp it into your drawing. Press the space bar while double clicking on the hand icon to see a menu of capital letters for pasting. You can also flip selected objects vertically or horizontally.

As with the previous version, the manual includes a Tricks and Hints section. Also included on the disk are several picture files for experimenting. You can see the available pictures by clicking on the Disk icon and bringing up a disk directory. It would be nice here to also have the ability to choose files with the joystick.



I believe software reviews are written for the designer as well as for the consumer. A review is one place where shortcomings or annoyances in a software package are revealed. A concerned developer considers a reviewer's criticisms, along with user's suggestions, when developing upgrades to programs. With this in mind, I checked to see if E.Z. Friendly has made any effort to correct the criticisms from the review of *Comics+*.

The first criticism from the previous review was that pressing the ESC key dumps you out of the program. The reviewer was used to the ESC key as being a way to back out of menus and found it difficult to break his old habit. E.Z. Friendly addressed this issue by disabling the ESC key. Now this key has no programming function. Actually, I wish it would allow you to back out of a menu. I found myself forced into making choices that led into unwanted menus. I was forced to choose when, in many programs, I can press the ESC key to return to a previous location.

The second criticism from the earlier review was the reviewer felt it would be nice if you could add to the clip art library. E.Z. Friendly addressed this complaint, too, and added a Custom option under the Clip icon. Now there is room to load clip art created by the *Art Util* program provided in the *Super Comics+* package. Also, the ability to cut and stamp allows you to load a picture, cut the clip art you want, reload the destination picture, and paste in the object.

The Clip Art utility allows you to add six clip art drawings to *Super Comics+*. Running *Art Util* brings up two sections: A drawing area, and a diagram showing the keys used for precision drawing. Press U for undo, E to erase the entire picture and S to save the picture. A bar graph at the bottom estimates whether or not your picture fits the limitations of the clip-art file. Keep the bar as small as possible for best results. Unfortunately, I'm not sure this is so easy to do. It seems as though each pixel movement adds to the amount of information, whether or not you are actually drawing. It is important, then, to begin at the cursor's initial starting point and draw from there. The manual offers an explanation for this.

After you finish drawing and press S for save, *Art Util* converts your drawing into "absolutely the most compact `DRAW` statement that can be written." If *Art Util* can't make a "short enough" `DRAW` statement, you are given the option of returning to the drawing and shortening it with the Undo function. *Art Util* actually builds a `DRAW` statement, taking into account each pixel move of the joystick. I think this makes drawing much more difficult. If *Art Util* is successful in creating a small enough `DRAW` statement, you are asked for a file number (numbers range from one through six) to add to the custom selection under the Clip icon. You can change the custom clips by renaming the custom clip files.

The last criticism from the earlier review was the desire that the program support more printers than the Tandy DMP-105 and 106. E.Z. Friendly gets good marks for listening. They have addressed each major criticism from the earlier review of their original product. Also, at the end of the manual they request users contact them with any suggestions or examples of other possible uses for the program. This support goes a long way toward ensuring the quality and usefulness of the package.

I don't really find the drawing abilities of the program strong when compared to *CoCo Max* and similar graphics programs, but you can incorporate pictures into *Super Comics+* by loading in any `PMODE4` picture. The picture must have a `.BIN` extension. This makes *Super Comics+* a good add-on utility for *CoCo Max* users. For *Comics+* owners, this upgrade is justified if you own a CoCo 3 with 512K. The major advantage to the upgrade is its ability to use a RAM disk. For owners with a CoCo 2, there are fewer reasons to purchase this upgrade. For new users, as the previous reviewer mentioned, "it will be a sure source of amusement. It's just plain fun to use."

[*Editor's Note: E.Z. Friendly has informed us of a newer version of* Super Comics+ *that supports both the IBM and Tandy modes. The IBM mode can also be used for other printers that support IBM emulation.*]

(E.Z. Friendly, 118 Corlies Ave., Poughkeepsie, NY 12601; 914-485-8150; upgrade $14.45, new $21.45; plus $1.50 S/H)

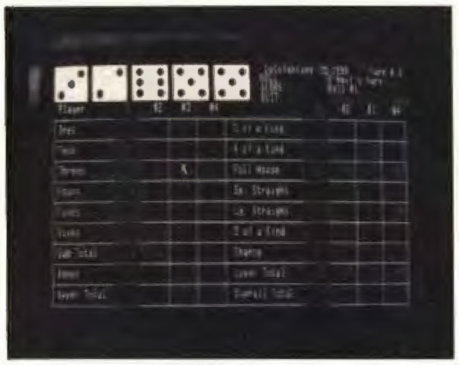
— Kay Cornwell

Game | OS-9 Level II

CoCoYahtzee

CoCoYahtzee from ColorSystems is played just like the *Yahtzee* game from Milton Bradley. It requires a CoCo 3 with a minimum 256K RAM, the `windint` module from *Multi-Vue* and a monitor. *CoCoYahtzee* cannot be played on a television. The game is not copy-protected, and the distribution disk contains a directory with the *CoCoYahtzee* documentation. I received 10 pages of printed documentation with the review disk.

CoCoYahtzee comes ready-to-run in the *Multi-Vue* environment. The author has included an icon file, three AIF files and the program module itself. The documentation helps you get set up to run the game by showing you how to copy the appropriate files to the right directories. The game can also be called from the OS-9 command line as long as the `windint` module is in your boot file. The game supports the Hi-Res



joystick interface outside of the *Multi-Vue* environment, however, *CoCoYahtzee* uses the default system settings so you may need to set up the Hi-Res mouse with the `SS.GIP` set status call. Alternatively, run `control-e` from your *Multi-Vue* disk to set up the mouse defined in the environment file.

CoCoYahtzee is designed for one to four players. Throughout the process of my review, the game operated smoothly and caused no problems, even with four players and the operation of other processes. The only difficulty I noticed was that during other processes, the mouse pointer took some time to update when *CoCoYahtzee* was not the current process. The only other confusing item about the game was in the AIF files. The author had the name of the program called by the AIFs as `yahtzee`. But the program name in the `CMDS` directory was `cocoyahtzee`. This caused an error when I first tried to run the game by clicking on the icon under *Multi-Vue*. I renamed the `cocoyahtzee` program in my `CMDS` directory to `yahtzee`. I also could have edited the AIF files and changed the first line to `cocoyahtzee`.

Support is provided by the author through the address and telephone number listed in the documentation. Support is also available through electronic services such as CompuServe, Delphi and GEnie.

[*Editor's Note: According to ColorSystems, the error with the CoCoYahtzee AIFs has been corrected. Current copies of the game do not exhibit this error.*]

(ColorSystems, 4616 Castle Hayne Rd., P.O. Box 540, Castle Hayne, NC 28429; 919-675-1706; $10)

— John Butler

Disk Utility | CoCo 3

Diskette File Protector

Do you work with confidential data files on your Color Computer? Or perhaps you are working on a hot new program and want to lock your executable file so no one else can run it. For whatever reason, there comes a time when you want to keep files from prying eyes. Computers can make privacy difficult — unless you have some kind of file locking utility. Datatech Micro System's first foray into the Color Computer market is just such a utility.

Diskette File Protector comes on a disk and includes a 2-page manual. The program consists of two binary files: `PROTECTX.BIN` and `PROTECTR.BIN`.

The manual explains the loading procedure to run the utility. Also covered is the process of making a backup of the original disk. (Because of the nature of this type of utility, I would have more than one backup lying around.) This is followed by an explanation of the program itself.

Diskette File Protector presents you with a directory of files — up to 65 — to protect as well as an All Files option. An underline indicates protected files (protected by this utility). If a file is not underlined, it has not been protected, but can be protected by placing a P beside the filename. If the file is underlined, meaning it has previously been protected, you can remove the protection by placing a U beside the filename. You can protect as many files as you want with the Protection routine. Use the All Files option to place protection on, or remove protection from, all files on the disk.

When all choices have been made, press ENTER and you are returned to the beginning screen. You may abort the protection process by pressing CLEAR instead of ENTER. Exit *Diskette File Protector* by pressing CLEAR from the beginning screen, which produces a reset and removes the program from memory so your files' protection cannot be removed by someone else.

Looking at the disk directory shows that the files protected by *Diskette File Protector* have been changed by renaming the file extension to `P::`. When a file is unprotected, the extension is returned to its original name. I tried to kill, rename and copy these protected files and received a FN error (Bad File Name or an unacceptable filename format). Once a file is protected, it cannot be opened, closed, loaded, killed, merged, renamed or copied. I protected an ASCII text file I had created in Microcom's

Word Power 3.2 word processor and then tried to edit the file, but a protected file cannot be loaded. You may add files to a protected disk. You may also change, rename and delete all unprotected files.

Under normal circumstances, protected files are impervious to snooping. I do not have any utilities to read and edit tracks or the File Allocation Table as you can with the *Norton Utilities* in the MS-DOS world. Therefore, I cannot verify if such a utility could be used to unlock these files. The only thing you can normally do with protected files is to unprotect them or reformat the disk upon which they are located.

What exactly does *Diskette File Protector* do? Does this utility make it impossible to delete or read the file? Does it prevent a reformat of the floppy disk? Can the file still be loaded into its application? The manual desperately needs an introduction to the functions of the package and its intended audience. An explanation of functions not performed by the program would also be useful.

The manual does not state whether or not *Diskette File Protector* is designed specifically for the CoCo 3, but the disk label specifies CoCo 3. I'm not sure if that means *Diskette File Protector* only works on the CoCo 3 or if there is a version for the older machines. This information should be clearly stated in the manual along with Datatech's address and phone number.

I spend most of my time in the MS-DOS world and am used to using utilities such as *Diskette File Protector* in the form of a comprehensive utility package like *Norton Utilities* or *PC Tools*. I'm not sure of Datatech Micro's intentions, but I suggest bundling *Diskette File Protector* as part of a CoCo Disk utilities package. As part of such a package *Diskette File Protector* would certainly be helpful. If you need a file protector, *Diskette File Protector* accomplishes this with minimum fuss.

[Editor's Note: The manual now reflects this utility as for the CoCo 3 only. There is no CoCo 1 or 2 version available. Also, the company address is clearly printed on each page of the manual.]

(Datatech Micro Systems, 4612 Arden, Lansing, MI 48917; $10)

— Kay Cornwell

Plug-N-Go...

NX-1001
MULTI-FONT

RAINBOW
NX-1020

This next generation color printer sets new standards in color printer performance... 225 cps, 4 NLQ fonts including Script, plus a new high speed draft font; but the enhancements don't stop there. Add a 16k buffer, a special quiet mode, top feed, bottom and rear tractor, and the list goes on. Seven on-demand colors, 8 color graphics modes, Epson and IBM emulation for maximum software compatibility. Virtually everything desired in a printer is here - *speed, color and versatility* at an affordable price with a 2 year warranty.

OUR *PLUG 'N GO FOR THE COCO* SYSTEM INCLUDES:
- **NX-1020 Rainbow Multi-Font Printer**
- **Blue Streak Ultima**
- **Software Support Disk**
- **Color Super Gemprint**

$249.95
+$10 Shipping & Insurance

This enhanced version of the legendary NX-1000II is the latest low-cost high-performance printer from Star Micronics. Fully featured with 4 NLQ plus a draft font. 10 character sizes from subscript to quadruple size, 4k buffer, 180 cps, friction and tractor feed, and much much more. Backed by a 2 year warranty. Epson and IBM emulation modes for maximum software compatibility. A performer so versatile you may never exhaust it's creative possibilities.

OUR *PLUG 'N' GO FOR THE COCO* SYSTEM INCLUDES:
- **NX-1001 Multi Font Printer**
- **Blue Streak Ultima**
- **Software Support Disk**

$188.88
+$10 Shipping & Insurance

Order Your System Today... Call (513) 885-5999
All Dayton Associate's products have a 30 day moneyback guarantee.

DAYTON ASSOCIATES of W.R. Hall, INC.
9644 Quailwood Trail • Spring Valley, Ohio 45370
Shipping charges to Canada, P.R., HI, AK, APO, FPO are double. Triple charge to all other countries

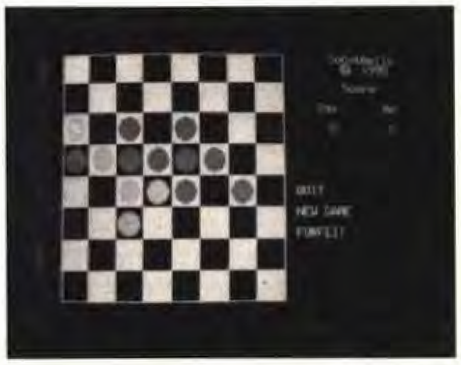
Visa & Master accepted within the continental U.S.
Ohio residents add 6.5% sales tax
COD add $4.00

Epson is a registered trade mark of Seiko Epson Corp. IBM is a registered trade mark of International Business Machines Corp. Price and Specifications are subject to change without notice.

Game	OS-9 Level II

CoCothello

I have always enjoyed playing the game *Othello*. Now, with the introduction of *CoCothello*, Version 1.4, I can enjoy this game on my Color Computer 3 under OS-9 Level II and *Multi-Vue*. *CoCothello* system requirements include a Color Computer 3, a joystick or mouse, a monitor (TV supported), OS-9 Level II and the `windint` module from the *Multi-Vue* disk. The `windint` module is used to provide joystick or mouse support, including the keyboard mouse. The game can be run from the OS-9 command line, but the `windint` module must be in your OS-9 boot file. (This is no problem for me because I use the `windint` module in my standard OS-9 boot file.) If the game is run from the command line, the Tandy Hi-Res interface is supported, but you may need to use the `SS.GIP` set status call to set up the Hi-Res mouse. Or, execute `control -e` from the *Multi-Vue* disk to set up the mouse.



CoCothello comes into its own when run under *Multi-Vue*. The author, Zack Sessions, has included an icon file and an AIF file. The instructions take you through the steps needed to set up the game under *Multi-Vue*. There are four AIF files supplied with the game. One AIF prompts you for your monitor type before starting the game. The other three support a default monitor type upon starting the game — one for RGB, one for composite (TV) and one for monochrome. Also, under *Multi-Vue* the Hi-Res interface can be used. The instructions are included as a text file on the game disk, but I received a four-page instruction document with the review disk.

I ran the game from both *Multi-Vue* and the OS-9 command line and experienced no problems at all. *CoCothello* is played against the computer. You are given a choice of two levels of play, your choice of color, and whether or not you want to go first. There are also two modes of play — the fast mode and the slow mode. In the fast mode the computer moves immediately and in the slow mode the computer waits for a mouse click before moving. *CoCothello* represents a good value in game software. Support is provided by the author by phone from 5 p.m. to 11 p.m. weekdays or on CompuServe, Delphi and GEnie. The information necessary to contact the author is provided in the documentation.

(ColorSystems, 4616 Castle Hayne Rd., P.O. Box 540, Castle Hayne, NC 28429; 919-675-1706; $10)

— John Butler

Utility CoCo 3

Nib Compressor and *Gallery Maker*

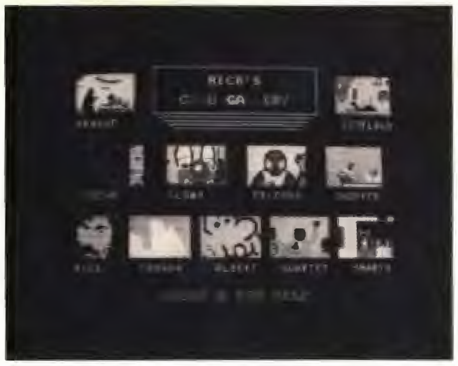
Rick's Computer Enterprise brings two great utility disks to the CoCo market to help view and store your graphics files. All of the programs reviewed here require a CoCo 3 and a disk drive. The programs are set up to allow the use of either a composite or RGB monitor.

The first utility is called *Nib Compressor*. This collection of programs allows you to save graphics screens to disk in a compressed format, thereby reducing the number of granules needed for such a file.

Many graphics compression schemes work on one byte (eight bits) of picture data at a time. *Nib Compressor* works on a nib (one half-byte) at a time.

The *Nib Compressor* disk I received came with four programs: `NIBLOADR.BAS`, `NIBLOADR.BIN`, `NIBSAVER.BAS` and `NIBSAVER.BIN`. Also on the disk is a sample "gallery" of pictures made with *Gallery Maker*, a program I will discuss later.

The `NIBSAVER` programs allow you to



compress and then save to disk any `HSCREEN` picture in the Hi-Res graphics screen memory of your CoCo. You have the option of using 192 or 200 lines of graphics data. The `NIBLOADR` programs allow you to load these files from disk to view on your monitor.

Although all graphics are saved in `HSCREEN2` mode, `NIBLOADR` has special keys that allow you to view the picture in the `HSCREEN` mode in which it was created, with 192 or 200 lines-per-screen.

The compression and saving, or loading and subsequent uncompression is quite fast — about five seconds to perform either function. An interesting side effect of these programs allows you to watch the graphics screen as it is compressed or uncompressed.

Gallery Maker is another fine program from Rick's Computer Enterprise. The *Gallery Maker* disk has five programs, in addition to the four `NIB` programs mentioned earlier, with which you can make

...For the CoCo

The Smallest, Sleekest, Fastest Serial To Parallel Converter You Can Buy!

7 Switchable Baud Rates
300 • 600 • 1200 • 2400 • 4800 • 9600 • 19200

Use this "smart" cable to connect a Centronics parallel printer to any version CoCo or use it to improve performance of your current printer. The cables are long-life, high quality shielded cables with moulded plugs for extra durability.

$34.95 +$2 Shipping

Software Support Disk $19.95
(free with purchase of printer)

TYPE SELECTION/ TUTORIAL	HI-RES SUPER GEMPRINT	SUPER GEMPRINT	COLOR SUPER GEMPRINT
Online instructional program that will select 24 special features of your printer or display methods to incorporate them into your programs	Disk software that will transfer a Hscreen 1,2,3 or 4 picture screen to printer. Grey level shading for color.	Will transfer a Pmode 0, 1, 2, 3, or 4 picture screen to printer 8"x11" hardcopy. Black/white, white/black or grey level shading for color	Transfers color screens to Color NX-1020. 8"x11" hardcopy in any Pmode or Hscreen mode

Order Your System Today... Call (513) 885-5999
All Dayton Associate's products have a 30 day moneyback guarantee.

DAYTON ASSOCIATES of W.R. Hall, INC.
9644 Quailwood Trail • Spring Valley, Ohio 45370
Shipping charges to Canada, P.R., HI, AK, APO FPO are double. Triple charge to all other countries

Visa & Master accepted within the continental U.S.
Ohio residents add 6.5% sales tax
COD add $4.00

your own personalized CoCo Gallery on Disk from the picture files you have in your collection.

Gallery Maker converts 11 pictures to NIB format and saves them to disk as a gallery. Then you can provide your name to personalize the title page of the CoCo Gallery. Next is an option to supply information about the graphics files, such as the name of the artist who created the pictures, how the pictures were created, or other data which can be accessed from the CoCo Gallery menu at a later time.

The menu is an outstanding graphics screen containing reduced images captured from each graphics picture. Use the arrow keys to place a box over an image and press ENTER to load the file for viewing. Press A to access the file's information.

I noticed a few bugs during reviewing of these programs. All of these bugs are with *Gallery Maker*.

When capturing an image from a picture, I could not move the cursor to the right third of the screen. This is perhaps a minor bug, but I could not capture details from the right side of the screen.

Another minor bug occurs when entering background information for a file. The manual and program states that you have about 240 characters for this entry, but entering as few as 128 characters causes the program to cease operation with an FC error. This should not, however, be a great problem since this number of characters is rarely used anyway, but it is something the author should consider.

The other error, also minor, occurs when attempting to run the CoCo Gallery menu immediately after the CoCo Gallery disk is created. This problem is solved by turning off the CoCo for a while and then running the menu again. This could be a problem associated with hardware memory.

I know it might sound as though I'm nit picking, but it is my duty as a reviewer to point out such problems to both you and the author.

Finally, let me say that these programs were fun. It was great to see my personal CoCo Gallery right on my monitor.

With the exception of the minor bugs, the program performs as the manual states. And the manual is not difficult to follow. A hint sheet is now included with all *Gallery Maker* orders. The hint sheet offers solutions to the minor bugs addressed in this review.

For the price, you cannot find a better deal for graphics utilities.

(Rick's Computer Enterprise, P.O. Box 276, Liberty, KY 42539; 606-787-5783; *Nib Compressor* disk, $2.50; *Gallery Maker* disk, $10; plus $2 S/H)

— Richard McNabb

Utility | CoCo 3

SmartWatch Drivers

Over the years, some members of the CoCo Community have found ways to interface various hardware items with the CoCo (many of which were never expressly intended for this purpose). For example, who in Ft. Worth would have ever dreamed that someone would interface robotic hardware with a CoCo 2? A friend of mine named Ray did just this. Now he uses it to control a carving machine that makes beautifully carved wooden signs, which he sells as his sole source of income.

Another piece of hardware not intended for the CoCo is the SmartWatch clock chip. Radio Shack and other electronics supply houses, including at least one RAINBOW advertiser, sell the SmartWatch. Until now, most Disk BASIC users have been unaware of its existence since it had little practical use with a CoCo in a standard setup. That has changed.

Robert Gault has come up with drivers that allow CoCo users to set and read, using Disk BASIC, the time on the SmartWatch clock in their system. When I first received the software for review, I was pretty disappointed. You see, when you buy a SmartWatch from a retail outlet (like Radio Shack) there are no installation instructions. And Mr. Gault's instructions assume the clock has already been installed. Another disappointment was with the inflexibility of the clock drivers. If the clock is not in the disk controller (something I cannot do with my unique setup) the drivers are useless.

I called Robert Gault with my observations and found him exceptionally open to new ideas and suggestions. He immediately sent me a new version of the drivers. I was very impressed the second time around. Not only did the new drivers search for the clock in the system, but there were also very detailed instructions for installation of the SmartWatch in the disk controller, RS-232 Pak or ROM Pak, with pictures for the novice. I have the clock in my RS-232 Pak plugged into Slot 3 of my Multi-Pak Interface, which is the reason I could not use the first set of drivers.

The SWSET program allows you to set the date and time in 12- or 24-hour mode, completely disable the clock interface when not in use to conserve battery life (10-year Lithium battery), or use the clock as an elapsed timer. After the clock is set, you can use the SWREAD routine to read the clock's date and time, or elapsed time. Remember, though, if you are using the elapsed-time mode, it stays that way until reset with the proper date and time. I should note that once the correct time is set, you won't have to reset it again until the time change in the spring or fall, or unless you want to use the elapsed timer. Also, because of its internal battery, powering down and power failures have no effect on the SmartWatch once you have set the unit.

When the SWREAD program is executed, DATE$ returns the date and time information, or you can assign it a variable, such as A$=DATE$:PRINT A$. I think anyone running a BBS could make excellent use of this feature for time-stamping calls. Unlike routines using the timer statement, where disk I/O suspends the timing for the duration of disk access, the SmartWatch is unaffected so the driver returns the correct time every time. With the clock in my RS-232 Pak, the driver found it no matter which slot of my Multi-Pak Interface held the Pak. Robert Gault indicated on the phone that the "slot search" of the driver slowed it down. I guess in a computer's world of nanoseconds it probably is slow, but to humans it appears instantaneous.

Also included on the disk is a driver for users of *Ultimaterm*. When using *Ultimaterm* with the clock installed and Robert Gault's driver, the time is constantly displayed in the upper-right corner of the screen. This is handy if you have no other clocks at which to look. You could set the clock to elapsed time just before a session to see approximately how long you've been in a BBS session, but, unfortunately, *Ultimaterm*'s clock-reset function (for its native elapsed timer) doesn't work with this driver. I was pleasantly surprised to find that this driver returned the correct time when running a patched version of *Ultimaterm* with *Extended ADOS 3*, which is my most-used DOS. In fact, if you are presently using *Extended ADOS 3* without previously incorporating a clock driver, SWREAD works with it as well, returning the correct time when you enter PRINT DATE$.

Except for the printed installation instructions, the documentation is contained in a text file on the disk. I feel it is now quite adequate for the novice to comprehend, although the RUNME program that displays the instructions on your screen does so in an 80-column mode, which may be difficult to read on some monitors or TVs. If this is the case, just load the INSTRUC.TXT file into your favorite word processor and print it to get a hardcopoy.

The price is very fair, and if you have any intention of buying a SmartWatch for use in your CoCo 3, I highly recommend this software to access it. Please note, the SmartWatch is not included.

(Robert Gault, 832 N. Renaud, Grosse Pointe Woods, MI 48236; $12, plus $4 S/H)

— Steve Ricketts

Received and Certified

The following products have recently been received by THE RAINBOW, examined by our magazine staff and issued the Rainbow Seal of Certification, your assurance that we have seen the product and have ascertained that it is what it purports to be.

DSDISK #4. This disk contains four programs written by Mr. David Sham. `MAC2.BAS` is a `.MAC`-to-`HSCREEN2` transfer (half-size) requiring 512K and a RAM disk. `MAC2.128` is the 128K version. `H1H2.BAS` is an `HSCREEN1`-to-`HSCREEN2` that requires a disk drive. `H2AR.BAS` is an `HSCREEN2` artifact color conversion program that also requires a disk drive. All programs require a CoCo 3. Five `.MAC` pictures, one `.CM3` image and one `HSCREEN1` shot are included on the disk to use for demos and testing the programs. *Mr. David Sham, 1155 E. 33rd Ave., Vancouver, BC V5V 3B4, Canada; $22 U.S. plus $3 S/H.*

◆ **The Magic Mortgage**, uses a simple but very powerful technique that most people wouldn't think could save such a large sum of money on their mortgages or long-term loans. Using *Magic Mortgage*, the average homeowner can save about $80,000 — some more, some less. The program comes on a cassette tape and is ready for loading into your CoCo 3. The program lets you design your own repayment plan — of course your lender must accept it. *Magic Mortgage* has a money-back guarantee. *Advanced Mortgage Systems, 111 Lake Street, Halifax, MA 02338, (617) 293-7610; $34.98 plus $4 S/H.*

CIII PagesE v2.5, enhancements and additions to *CIII Pages* V2.0. Desktop publisher, greeting card designer, form maker, calligrapher, or CAD for the CoCo 3. Includes pull-down menus, icons and dialog boxes. Allows you to mix text with graphics. Magnify, rotate, flip, reduce, stretch and slide screen in seconds. Requires a CoCo 3, Tandy Hi-Res interface, monitor, joystick/mouse. Compatible with several brands of printers. *Coless Computer Design, 1917 Madera St. #8, Waukesha, WI 53186, (414) 549-0750; $49.95 plus $3 S/H.*

CIII ClipartE, 1500+ pieces of clip art for *CIII PagesE* that comes on five double-sided disks. A viewing utility is included on Side A of each disk to view the clip art outside the *CIII PagesE* environment. Clip art includes: Holidays, cartoon characters, astrological signs, education, medals, vehicles, sports, signs, animals, entertainment, seasonal, fantasy, states, ornamental capital letters, and other miscellaneous artwork. *Coless Computer Design, 1917 Madera St. #8, Waukesha, WI 53186; $29.95 plus $3 S/H.*

Window Master Desk Accessory Pak, consists of several programs that can be loaded into the Window Master Operating System as resident desk accessory programs. *Window Master* has always supported this type of programs, but they haven't been used to their fullest until now. These accessory programs require *Window Master* V3.0. They can be used at any time without disturbing the current program that is running on the system. Programs include: Font and Icon editors, function keys, terminal program, graphics editor and a calendar. *Cer-Comp Ltd., 5566 Richochet Avenue, Las Vegas, NV 89110, (720) 452-0632; $39.95 plus $4 S/H.*

◆ **Power Pieces**, a two-player strategy game using a total of 16 pieces. Players maneuver these pieces around the board trying to eliminate the opposing players' pieces or capture enemy territories. Three different sets of army pieces to choose from: Civil War, WWI and WWII. Requires OS-9 Level II, 512K CoCo 3 and a disk drive. *Game Gem Games, P.O. Box 487, Clarksburg, WV 26301; $19.95.*

Quick Stats 3, a file-oriented baseball stats program designed especially for rotisserie fans. The program is also useful for recording major and minor league stats, as well as for softball. Keep track of over a dozen stats, including at bats, hits, home runs, RBIs, stolen bases, averages, innings pitched, earned runs, walks, wins, saves, ERA and ratio. Files can be viewed onscreen or printed. *Trading Post, P.O. Box 3453, Carbondale, IL 62902-3453, (618) 457-5258; $24.95 plus $3 S/H.*

KJV on Disk #20, Psalms 88-150 from the King James version of the Bible in ASCII files for the CoCo 1, 2 and 3. A word processor or text editor is recommended for viewing the files. Requires at least 32K and one disk drive. *BDS Software, P.O. Box 485, Glenview, IL 60025-0485; $3.*

POW/MIA Database, a database of the 2,260 American Prisoners and Missing in Southeast Asia (as of November 1990) from all military services. It was established to honor these personnel and help us remember their sacrifices to keep us free. Data elements include the POW/MIA home state, hometown, name, military service and grade (rank), birth date, and where and when the POW/MIA was recorded as missing. Requires a disk drive. A printer is optional. *Johnson Software, P.O. Box 92, Dayton, OH 45449, (513) 866-2601; $17.50 plus $2 S/H.*

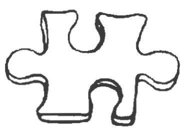
Baseball Card Catalog, a database for all of your baseball cards. Enter, delete and view your pitcher, infielder, record breaker and managing staff cards. Requires two disk drives and a CoCo 3. CoCo 1 and 2 cassette version also available. *The Millsoft Company, Box 2377, Amagansett, NY 11930, (516) 324-7953; $18.95.*

◆ **First product received from this company**

The *Seal of Certification* is open to all manufacturers of products for the Tandy Color Computer, regardless of whether they advertise in THE RAINBOW.

By awarding a *Seal*, the magazine certifies the program does *exist* — that we have examined it and have a sample copy — but this *does not* constitute any guarantee of satisfaction. As soon as possible, these hardware or software items will be forwarded to THE RAINBOW reviewers for evaluation.

The Assembly Line
Searching Word Puzzles



by William P. Nee

Several people have written to me to ask how to process and sort strings in machine language. To make this more interesting, I thought I'd use these ideas in a word-search puzzle — where a certain number of words are hidden within a large grid and surrounded by random letters. In this type of puzzle the words can be positioned horizontally, vertically, backward, forward, or top-to-bottom. This program generates a word-search puzzle from your list of words. Word-search puzzles are very popular at family reunions and children's birthday parties.

Essential to the study of strings in the CoCo is an understanding of the VARPTR function. This routine is like a map of all the strings you're using. It doesn't tell you what the string is, but rather, gives you information about the string. This information is generally five bytes long and consists of the string length, a zero, the string's location (two bytes), and another zero. Using the function V=VARPTR(A$), V is the *location* where these five bytes of information start. The two bytes just before V are the ASCII value of the string's name (128 is added to the value of the second letter of the name so the CoCo knows it's dealing with a string variable and not a numeric variable). Try the following example:

```
10 DIM AB$,V
20 AB$="THE RAINBOW IS A GREAT M
AGAZINE"
30 V=VARPTR(AB$)
40 PRINT CHR$(PEEK(V-2));
41 PRINT CHR$(PEEK(V 1) 128)
42 PRINT PEEK(V)
43 PRINT HEX$(PEEK(V+2));
44 PRINT HEX$(PEEK(V+3))
```

Here's all the information about AB$. If you need to check any of the letters in the string, they start at the location printed by lines 43 and 44. Starting at that location are the ASCII values for each letter in the string.

But what if we want our words in a one-dimensional array? Again, everything is located before and after the VARPTR value. This time, though, we need the initial location of AB$(0). Information about the array is contained in the seven bytes before this location:

V-7 = array's first name
V-6 = array's second name plus 128
V-5, V-4 = location of next array
V-3 = number of dimensions
V-2, V 1 = number of strings

The next group of five bytes contain the information about individual strings in the array:

V = length of AB$(0)
V+1 = 0
V+2, V+3 = location of AB$(0)
V+4 = 0
V+5 = length of AB$(1)
etc.

Armed with all this information, you can now charge through string arrays and change them at will. The only unchangeable information is the string's actual location. Notice that earlier in Line 10 both the array name and Variable V are dimensioned at the beginning of the program. The CoCo has a tendency to bounce string locations around, especially if they are used for different variables. This is commonly called *garbage collection*. Giving variables an immediate "home" ensures that whenever you call VARPTR, you'll always get the same location.

In the puzzle program, a one-dimensional array is used. V-3 is 1, the number of strings is less than 256, so V 2 is always 0, and the array name is WD$. All that is needed for further checking is the total number of strings, the length of each string, and it's location.

Bill Nee reversed the snowbird trend by retiring to Wisconsin from a banking career in Florida. The success of his 13-part series, "Machine Language Made BASIC" (July 1988 to July 1989), prompted him to continue writing articles about Color Computer machine-language programming. You may contact Bill at Route 2, Box 216C, Mason, WI 54856-9302.

The machine-language program in Listing 2 starts by defining its variables (lines 110-260). The `JSR $B3ED` in Line 280 gets the `VARPTR` of `WD$(0)` that has been passed by the BASIC program (Listing 1) and stores it in `VARPTR`. This means that `VARPTR-1` contains the number of words in the puzzle, plus 1 (there is no `WD$(0)`).

It's easier to fit the longest words into the puzzle, so the program sorts all the words by length. The locations of the words won't change, just the information in the five-byte groups. Since `VARPTR` has been put in Register U, increasing U by five is the `VARPTR` for `WD$(1)`. Five bytes away from that is the start of the five-byte information for `WD$(2)`. The first byte of each location is the length of the respective words. If the length of the second word is less than the length of the first word, swap `VARPTR` information entirely (lines 380-440). When the swap is complete, it's back to the beginning. If the first word is the same length or longer than the second word, skip the swap and compare the second and third words. It's not a very graceful routine, but it is quick in machine language and easy to understand.

Next is the bulk of the program. There are three overall attempts to fit a word into the current size puzzle array before the size is increased. After the screen is cleared, the current size is printed (lines 520-590) and the array is cleared to zero (lines 600-630). After the number of words in the puzzle (less 1) is determined, the first word is located and displayed on the screen (lines 650-780). Location $6 is a built-in CoCo flag indicating whether the current operation involves

64K Disk

Listing 1: `WORDPUZL`

```
1 'ASSEMBLY LINE
2 'WRITTEN BY WILLIAM NEE
3 'COPYRIGHT JUNE 1991
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 CLEAR 200,&H6000-1:CLS
20 IF PEEK(&H6011)<>189 THEN LOA
DM"WDPUZZLE":POKE&HFF40,0
30 PRINT"  P U Z Z L E   S E A R
 C H "
40 DIM TI$,N,SZ,V,L,T
50 DEF USR=&H6011
60 V=RND(-TIMER)
70 READ TI$:V=LEN(TI$)
80 PRINT#-2,CHR$(27);CHR$(16);CH
R$(0);CHR$(0):PRINT#-2
90 PRINT#-2,TAB(40-V/2) TI$
100 PRINT#-2,CHR$(27);CHR$(16);C
HR$(9);CHR$(0)
110 PRINT#-2
120 READ N:DIM WD$(N)
130 FOR V=1 TO N:READ WD$(V)
140 IF LEN(WD$(V))>L THEN L=LEN(
WD$(V))
150 IF L>15 THEN PRINTWD$(V);" I
S TOO LONG":STOP
160 T=T+LEN(WD$(V))
170 NEXT
180 T=INT(SQR(T)+.5)+2:L=L+1
190 IF T>L THEN L=T
200 IF L>30 THEN PRINT"TOO MANY
WORDS":STOP
210 PRINT"MINIMUM SIZE IS -";L
220 INPUT"DESIRED SIZE    ";SZ
230 IF SZ=0 THEN SZ=L
240 IF SZ>30 THEN SZ=30
250 POKE &H6009,SZ
260 V=USR(VARPTR(WD$(0)))
270 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(0)
280 PRINT#-2
290 FOR V=1 TO N:L=(V-1)AND 3
300 PRINT#-2,TAB(20*L) WD$(V);
310 IF L=3 THEN PRINT#-2,CHR$(27
);CHR$(16);CHR$(0);CHR$(0)
320 NEXT:PRINT#-2
330 STOP
340 DATA P R E S I D E N T I A L
  Q U I Z,32
350 DATA WASHINGTON,ADAMS,JEFFER
SON,MADISON,MONROE,JACKSON,VANBU
REN,POLK,TAYLOR,PIERCE
360 DATA BUCHANAN,LINCOLN,GRANT,
HAYES,GARFIELD,CLEVELAND,HARRISO
N,TAFT,WILSON,HARDING,COOLIDGE,H
OOVER
370 DATA ROOSEVELT,TRUMAN,EISENH
OWER,KENNEDY,JOHNSON,NIXON,FORD,
CARTER,REAGAN,BUSH
```

Listing 2: `WDPUZZLE.ASM`

```
00050 *WORD SEARCH PUZZLE
00100          ORG    $6000
00110 VARPTR   RMB    2
00120 TOTAL    RMB    1
00130 NUMBER   RMB    1
00140 WDLEN    RMB    1
00150 WDLEN1   RMB    1
00160 RCOUNT   RMB    1
00170 RNUM     RMB    1
00180 RTIMES   RMB    1
00190 SIZE     RMB    1
00200 CX       RMB    1
00210 CY       RMB    1
00220 CX1      RMB    1
00230 CY1      RMB    1
00240 SGNDX    RMB    1
00250 SGNDY    RMB    1
00260 TAB      RMB    1
00270
00280 LENSRT   JSR    $B3ED    GET WD$(0) VARPTR
00290          STD    VARPTR   SAVE IT
00300 LSORT    LDU    VARPTR
00310          LDA    -1,U     MAXIMUM NUMBER OF WORDS
00320          SUBA   #2       DON'T NEED FIRST AND LAST ONE
00330          STA    NUMBER
00340          LEAU   5,U      VARPTR OF WD$(1)
00350 LSRT1    LDA    ,U       IT'S LENGTH
00360          CMPA   5,U      COMPARE TO NEXT WORD'S LENGTH
00370          BHS    LSRT2
00380          LDB    5,U      GET SHORTER WORD LENGTH
00390          STB    ,U        AND SWAP IT
00400          STA    5,U
00410          LDD    2,U
00420          LDX    7,U      GET IT'S LOCATION
00430          STX    2,U       AND SWAP IT TOO
00440          STD    7,U
00450          BRA    LSORT    BACK TO BEGINNING
00460 LSRT2    LEAU   5,U      NEXT WORD VARPTR
```

strings ($6<>0) or numbers ($6=0). Using a print routine usually changes $6 to 255, but RND uses the Floating Point Accumulator, and it must be cleared for operations with numbers.

The puzzle has eight possible directions for word placement; north, east, south, west, and diagonally between each. Depending on the direction, a letter's next location is either -1, 0 or +1 from its present location in both the horizontal and vertical directions. These values are located in RTABLE (lines 2670-2740) and represent the changes in location for the eight directions starting with north and moving clockwise (remember that to the computer -1 = $FF). Even though there are eight directions, they are numbered from 0 through 7. The first direction is randomly selected (lines 890-1000).

Next, a random location is chosen within the array (lines 1020-1130) where the program tries to start the current word. The location must first be checked to see if the word fits in the current direction (lines 1150-1300). Every location where a letter would be must then be checked to see if the location currently is a zero and not occupied (lines 1320-1420) — no words overlap, but you could modify the program to allow for this. The program tries to place a word in the given direction 50 times before giving up and going to the next direction.

When all of these conditions have been met, the word can be put in the array (lines 1510-1660). If there are any more words, they are placed next (lines 1670-1680). Now it's time to print the array. Location $6F is the output device location. If it contains -2 ($FE), output goes to the printer (PRINT#-2) and if it contains zero, output goes to the screen. The DMP-105 has 480 printing positions. Each letter plus a space is 12 positions long, so the TAB setting is (480-SIZE*12)/2 or 240-SIZE*6 (lines 1720-1780). You may have to modify this procedure and the following printer commands for your own printer.

The print head is positioned using four values — 27, 16, 0, and TAB, which start at Location 1,1 in the array (lines 1800-1970). If that location contains a letter, it is printed. If there is no letter at that location, a random letter is generated and printed (lines 1980-2070). The entire array is checked and random letters are printed in place of blank array locations (lines 2070-2170). When this is completed, all the words contained in the puzzle are printed below the puzzle in four columns. But I wanted the words printed alphabetically, so the program sorts them. And to be really correct, they are not just sorted alphabetically, but they are also sorted by length — "act" comes before "action".

Again, VARPTR is used to get the number of words in the array without the first and last words. I used the TOTAL variable as a

```
00470          DEC     NUMBER   CHECKED ALL THE WORDS?
00480          BNE     LSRT1
00490
00500 BEGIN    LDB     #3       THREE TRYS PER WORD
00510          STB     TOTAL
00520 START    JSR     $A928    CLS
00530          LDX     #MSG     SIZE MESSAGE
00540          JSR     $B99C    PRINT TO SCREEN
00550          LDB     SIZE
00560          JSR     $BC7C    PRINT THE CURRENT SIZE
00570          JSR     $BDD9
00580          JSR     $B99C
00590          JSR     $B958     AND A CARRIAGE RETURN
00600          LDU     #ARRAY   CLEAR ARRAY TO ZERO
00610 CLOOP    CLR     ,U+
00620          CMPU    #ARRAY+960
00630          BLO     CLOOP
00640
00650          LDY     VARPTR   START OF WD$
00660          LDA     -1,Y     NUMBER OF WORDS IN WD$
00670          DECA             LESS WD$(0)
00680          STA     NUMBER
00690
00700 LOOP1    LEAY    5,Y      WD$(1)
00710          LDB     ,Y       IT'S LENGTH
00720          STB     WDLEN
00730          LDX     2,Y       AND IT'S LOCATION
00740 CL1      LDA     ,X+      FIRST LETTER
00750          JSR     [$A002]  PRINT IT
00760          DECB
00770          BNE     CL1      PRINT ENTIRE WORD
00780          JSR     $B958     AND A C/R
00790          CLR     $6       SET NUMBER FLAG
00800          LDB     #9       NUMBER OF DIRECTIONS + 1
00810          STB     RCOUNT
00820          LDB     #8
00830          JSR     $BC7C
00840          JSR     $BF1F    GET A RANDOM DIRECTION
00850          JSR     $B3ED
00860          DECB              MAKE IT 0 - 7
00870          STB     RNUM
00880
00890 RLOOP    DEC     RCOUNT   START OFF WITH 8 DIRECTIONS
00900          LBEQ    PRINT1   COULDN'T FIT THIS WORD IN!
00910          LDB     RNUM
00920          INCB             CHANGING DIRECTIONS
00930          ANDB    #7        STILL BETWEEN 0 - 7
00940          STB     RNUM
00950          LSLB             RTABLE IS 2 BYTES EACH
00960          LDX     #RTABLE
00970          LDD     B,X      CHANGES FOR THAT DIRECTION
00980          STD     SGNDX    SAVE THOSE CHANGES
00990          LDB     #50      GET 50 ATTEMPTS PER DIRECTION!
01000          STB     RTIMES
01010
01020 CENTER   LDB     SIZE     GET A RANDOM LOCATION IN ARRAY
01030          JSR     $BC7C
01040          JSR     $BF1F
01050          JSR     $B3ED
01060          STB     CX
01070          STB     CX1
01080          LDB     SIZE
01090          JSR     $BC7C
01100          JSR     $BF1F
01110          JSR     $B3ED
01120          STB     CY
01130          STB     CY1
01140
01150 CHKLEN   LDA     WDLEN    WILL THE WORD FIT?
01160          LDB     SGNDX
01170          MUL
01180          ADDB    CX
01190          LBMI    NOGOOD
01200          LBEQ    NOGOOD
01210          CMPB    SIZE
01220          LBHI    NOGOOD
01230          LDA     WDLEN
```

```
01240        LDB    SGNDY
01250        MUL
01260        ADDB   CY
01270        LBMI   NOGOOD
01280        LBEQ   NOGOOD
01290        CMPB   SIZE
01300        LBHI   NOGOOD
01310
01320 ALLCLR LDA    WDLEN   ALL CLEAR IN THAT DIRECTION?
01330        STA    WDLEN1
01340        LDD    CX
01350        STD    CX1
01360 CLRLP  LDA    CY1
01370        LDB    SIZE
01380        MUL
01390        ADDB   CX1
01400        ADCA   #0
01410        LDU    #ARRAY
01420        TST    D,U     IS LOCATION IN ARRAY ZERO?
01430        LBNE   NOGOOD  CAN'T USE IF IT ISN'T
01440        LDD    CX1
01450        ADDA   SGNDX
01460        ADDB   SGNDY
01470        STD    CX1     NEXT LOCATION TO CHECK
01480        DEC    WDLEN1
01490        BNE    CLRLP
01500
01510 STORIT LDX    2,Y     PUT THE WORD IN THE ARRAY
01520 STORLP LDA    CY
01530        LDB    SIZE
01540        MUL
01550        ADDB   CX
01560        ADCA   #0
01570        LDU    #ARRAY
01580        LEAU   D,U     ARRAY LOCATION
01590        LDA    ,X+     CURRENT LETTER
01600        STA    ,U
01610        LDD    CX
01620        ADDA   SGNDX
01630        ADDB   SGNDY
01640        STD    CX      NEXT ARRAY LOCATION
01650        DEC    WDLEN   ANY MORE LETTERS?
01660        BNE    STORLP
01670        DEC    NUMBER  ANY MORE WORDS?
01680        LBNE   LOOP1
01690
01700 PRINT2 LDA    #-2     SWITCH TO THE PRINTER
01710        STA    $6F     OUTPUT DEVICE LOCATION
01720        LDA    SIZE    CENTER THE PRINTING
01730        LDB    #6
01740        MUL
01750        PSHS   B
01760        LDA    #240
01770        SUBA   ,S+
01780        STA    TAB     LEFT MARGIN AND RIGHT MARGIN
01790        LDY    #ARRAY
01800        LDA    #1      START AT ARRAY(1,1)
01810 P2L1   STA    CY
01820        LDA    #27     PRINTER COMMANDS
01830        JSR    [$A002]
01840        LDA    #16      TO MOVE
01850        JSR    [$A002]
01860        CLRA            TO LEFT
01870        JSR    [$A002]
01880        LDA    TAB      MARGIN
01890        JSR    [$A002]
01900        LDB    #1
01910 P2L2   STB    CX
01920        LDA    SIZE
01930        LDB    CY
01940        MUL
01950        ADDB   CX
01960        ADCA   #0
01970        LDA    D,Y     FIRST LETTER IN ARRAY(1,1)
01980        BNE    P2L3    IS IT A PUZZLE LETTER?
01990        CLR    $6      NUMBER FLAG
02000        LDB    #26     REPLACE 0 WITH RANDOM LETTER
```

Submitting Material To Rainbow

Contributions to THE RAINBOW are welcome from everyone. We like to run a variety of programs that are useful, helpful and fun for other CoCo owners.

WHAT TO WRITE: We are interested in what you want to tell our readers. We accept for consideration anything that is well-written and has a practical application for the Tandy Color Computer. If it interests you, it will probably interest lots of others. However, we vastly prefer articles with accompanying programs that can be entered and run. The more unique the idea, the more the appeal. We have a continuing need for short articles with short listings. These are especially appealing to our many beginners.

FORMAT: Program submissions must be on tape or disk, and it is best to make several saves, at least one of them in ASCII format. We're sorry, but we do not have time to key in programs and debug our typing errors. All programs should be supported by some editorial commentary explaining how the program works. We also prefer that editorial copy be included in ASCII format on the tape or disk, using any of the word processors currently available for the Color Computer. Also, please include a double-spaced printout of your editorial material and program listing. Do not send text in all capital letters; use upper- and lowercase.

COMPENSATION: We do pay for submissions, based on a number of criteria. Those wishing remuneration should *so state* when making submissions.

For the benefit of those wanting more detailed information on making submissions, please send a self-addressed, stamped envelope (SASE) to: Submission Guidelines, THE RAINBOW, The Falsoft Building, P.O. Box 385, Prospect, KY 40059. We will send you comprehensive guidelines.

Please do not submit material currently submitted to another publication.

flag, since it is not presently used anywhere. The lengths of the first and second words are compared. If the second word is smaller, use that length, but "set" the flag. Each letter in the first word is compared to the corresponding letter in the second word. If a letter in the first word is lower in value than the letter in the second word, the word is correctly placed and you can go on to the next word. If a letter is greater than the same letter in the next word, then the two words must switch places — the five bytes of VARPTR information must be exchanged. If the two words are identical in length, then the flag must be checked. If the flag is zero, the first word is the shorter and in its proper location. If the flag is set, then the second word was the shorter and the two words must switch places. When a swap is made, the program goes back to the beginning of the sort. After debugging, save the source with `W WDPUZZLE.SRC`, and assemble it with `A WDPUZZLE.BIN /NS/WE`.

Another interesting option is to have the letters left after you've circled all the words spell a phrase — doing this is a challenge!

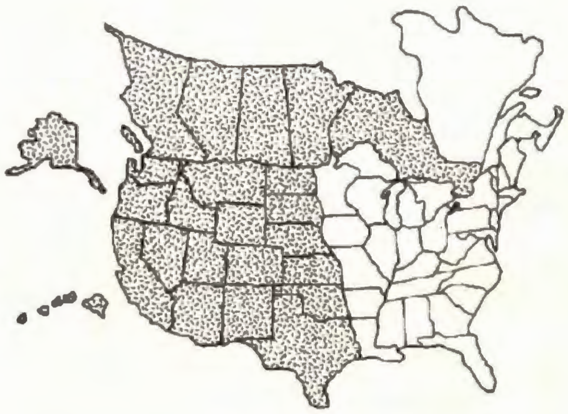
After the sort is completed, control returns to the BASIC program and it prints all the puzzle's words in four columns. The BASIC program also loads the machine-language program (if necessary), limits the word lengths to 15 letters, and determines the size of a puzzle. The first two items in your data lines should be the puzzle title and the number of words. In addition to previously mentioned changes, you may want to add options that allow saving and loading the words from disk, printing more than one copy, or adding a solution printout with an asterisk (*) instead of random letters. Another interesting option is to have the letters left after you've circled all the words spell a phrase — doing this is a challenge!

That's it for this month. Happy word hunting. Next time I'll answer some specific machine-language questions, discuss *Disk EDTASM+*, give you some tips and use a Japanese toy to demonstrate probability. If you have any questions or ideas for future articles, please let me know. ❑

```
02010          JSR    $BC7C
02020          JSR    $BF1F
02030          JSR    $B3ED
02040          ADDB   #64 CHR$(65 - 90)
02050          TFR    B,A
02060 P2L3     JSR    [$A002] PRINT LETTER
02070          JSR    $B9AC    AND A SPACE
02080          LDB    CX
02090          INCB
02100          CMPB   SIZE    ALL THE WAY ACROSS?
02110          BLS    P2L2
02120          JSR    $B958   PRINT A C/R
02130          LDA    CY
02140          INCA
02150          CMPA   SIZE    ALL THE WAY DOWN?
02160          BLS    P2L1
02170          CLR    $6      SET NUMBER FLAG
02180
02190 ALPSRT   LDU    VARPTR  WD$ LOCATION
02200          LDA    -1,U    NUMBER OF WORDS
02210          SUBA   #2      DON'T USE WD$(0) OR LAST ONE
02220          STA    NUMBER
02230          LEAU   5,U     FIRST WORD
02240 ASRT1    LDA    ,U       IT'S LENGTH
02250          CLR    TOTAL   CLEAR SWAP FLAG
02260          CMPA   5,U     COMPARE TO LENGTH OF NEXT WORD
02270          BLS    OK
02280          LDA    5,U     USE THE SMALLER OF THE TWO
02290          INC    TOTAL   SET THE SWAP FLAG
02300 OK       STA    WDLEN
02310          LDX    2,U     LOCATION OF THIS WORD
02320          LDY    7,U     LOCATION OF NEXT WORD
02330 ASRT2    LDA    ,X+     FIRST LETTER OF THIS WORD
02340          CMPA   ,Y+     FIRST LETTER OF NEXT WORD
02350          BLO    ADONE   ALREADY ALPHABETICAL
02360          BEQ    ANEXT   KEEP CHECKING
02370          BRA    ASWAP   HAVE TO SWITCH THEM
02380 ANEXT    DEC    WDLEN   ANY MORE LETTERS?
02390          BNE    ASRT2
02400          TST    TOTAL   DID WE USE NEXT WORD LENGTH?
02410          BEQ    ADONE
02420          BRA    ASWAP   IF SO, SWAP THE TWO WORDS
02430 ADONE    LEAU   5,U     NEXT WORD LOCATION
02440          DEC    NUMBER  ANY MORE WORDS?
02450          BNE    ASRT1
02460          RTS            SORT IS ALL - BACK TO BASIC
02470 ASWAP    LDA    ,U      THIS WORD'S LENGTH
02480          LDB    5,U      NEXT WORD'S LENGTH
02490          STB    ,U       SWAP LENGTHS
02500          STA    5,U
02510          LDD    2,U     THIS WORD'S LOCATION
02520          LDX    7,U      NEXT WORD'S LOCATION
02530          STX    2,U      SWAP THEM
02540          STD    7,U
02550          BRA    ALPSRT  SWAP DONE; BACK TO BEGINNING
02560
02570 NOGOOD   DEC    RTIMES
02580          LBNE   CENTER  TRY A NEW LOCATION
02590          LBRA   RLOOP   USED UP ALL 50 TRIES
02600
02610 PRINT1   DEC    TOTAL
02620          LBNE   START   TRY AGAIN
02630          CLR    $6      SET NUMBER FLAG
02640          INC    SIZE    TRY A LARGER ARRAY
02650          LBRA   BEGIN   START ENTIRE PROGRAM AGAIN!
02660
02670 RTABLE   FDB    $00FF
02680          FDB    $01FF
02690          FDB    $0100
02700          FDB    $0101
02710          FDB    $0001
02720          FDB    $FF01
02730          FDB    $FF00
02740          FDB    $FFFF
02750 MSG      FCC    / CURRENT SIZE IS /
02760          FCB    0
02770 ARRAY    RMB    1
02780          END    LENSRT
```

Advertisers Index

We encourage you to patronize our advertisers — all of whom support the Tandy Color Computer. We will appreciate your mentioning* THE RAINBOW *when you contact these firms.



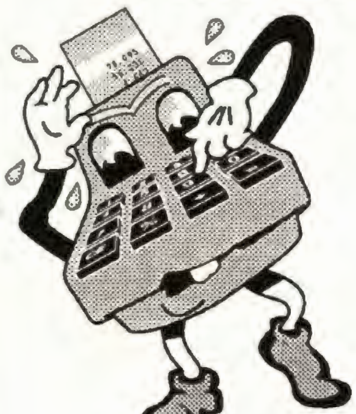
▨ **Call:**
Belinda Kirby
Advertising Representative
(502) 228-4492

☐ **Call:**
Kim Lewis
Advertising Representative
(502) 228-4492

The Falsoft Building
9509 U.S. Highway 42, P.O. Box 385, Prospect, KY 40059

FAX (502) 228-5121

Add UP These Public Domain Values!



Music 1-7

M1 - 8 Utilities & 8 Songs
M2 - 17 Musica Files
M3 - 16 Musica Files } Requires Musica
M4 - 16 Musica Files
M5 - 25 Orchestra Files
M6 - 23 .Bin Files Ready To Run
M7 - 23 .Bin Files Ready To Run

ADVENTURES 1,2

Ready To Run
Order A1 Or A2

UTILITIES 1-8

• 12 Programs Each, 1-4 Require Disk •
U1 - Backup 35, Basicmap, Timer, Tape-to-Disk, +
U2 - Customize, DiskTest, MasterDisk, Offset, +
U3 - View 64, DiskAid, Disk Library, Atari, +
U4 - AutoBoot, MacPicture, StateLog, UniBack, +
U5 - Disassembler, Graph, PaintPot, Squeeze, +
U6 - Clock, Lilst 32, , MiniDos, Pixcomp, +
U7 - Head Print With 30 Mini Pictures
U8 - Fig Forth Language With Tutiorial

GAMES 1-11

• Each Disk/Tape Contains 12 Programs •
GA1 - 3D Tic Tac, Missile, Poker, Tycoon, +
GA2 - Amazing, Balloons, Motor Jump, Slots, +
GA3 - Battleship, Golf, Lander, Zero G, +
GA4 - ABM, LunarWorld, Protect, Subchase, +
GA5 - Blackjack, Lazer, Tipan, Utopia, +
GA6 - Kings, Maze, Shuffle, Tempest, +
GA7 - Chess, City, Life, StarTrek, +
GA8 - Chute, Germ, Hurkle, Lunar, +
GA9 - Civil War, Go-Fish, Stock, UFO Maze, +
GA10 - Cave, HiLow, Scramble, Word Hunt, +
GA11 - Biorythms, Craps 3, Gunner, Martians, +

EDUCATION 1-4

E1 - 12 Programs For Young Kids
E2 - 12 Programs For High School Kids
E3 - 11 Programs Teaching Coco Commands
E4 - 5 Graphics Programs About Australia

GRAPHICS 1-14

GR1 - Atlanta, Cube, Space, +
GR2 - Objects, Wargame, Worldmap, +
GR3 - 9 Coco 3 Graphic Programs
GR4 - 22 Coco Max Pictures
GR5 - 22 Coco Max Pictures
GR6 - 22 Coco Max Pictures
GR7 - 15 Coco Max Pictures
GR8 - 22 .Bin Pictures
GR9 - 22 .Bin Pictures
GR10 - 14 Large .Bin Pictures
GR11 - 8 Mge Pictures
GR12 - Coco Max 3 Pictures
GR13 - Macpaint Graphic Editor
GR14 - 5 Macintosh Pictures

TELECOMMUNICATIONS 1-3

T1 - Haysae, Kermit, Mikeyterm, Teleterm
T2 - Cobbs BBs Terminal Package
T3 - GETERM Communications

HOME MANAGEMENT 1-4

• 12 Programs Each Disk/Tape •
H1 - Checkbook, Loan, Word Processor, +
H2 - Cash Journal, Invest, Maillist, +
H3 - Life, Database, Stocks, +
H4 - Spelling Fix, Spelling Checker, +

And Don't Forget Monthly Back Issues Of Original Subscription Software

RAINBOW CERTIFICATION SEAL

ISSUE #91, JAN 1990	**ISSUE #92, FEB 1990**	**ISSUE #93, MAR 1990**	**ISSUE #94, APR. 1990**	**ISSUE #95, MAY 1990**	**ISSUE #96, JUN 1990**
TRENCH FIGHTER	PENTA-SAME	MR. MOVIE	FLASH CARD	DREAMTIME ADVENTURE	ENTREPENEUR
COCVERT3	ROTISSERIE LEAGUE	INVADERS	REAL ESTATE LISTINGS	XT/AT SYSTEMS	DR. PHREDI
SPEED GAMES	NICK'S QUEST	COCO3 RECIPE MACHINE	BIBLE QUESTIONS 1	BIBLE QUIZ 2	PRINTER FONT CODES
BUSINESS STARTER	SOLITAIR	SILVERCAPE 3 ADV.	SPELLING QUIZ 1	16 FLAGS	BIBLE QUESTIONS PT. 3
CAVERN QUEST3 PT. 2	EDUCATION 4	BABY-SITTER'S BASE	NO LEFT 3	FRACTIONS	CALO ADVENTURE
BUILDERS HELPER	IBM - BASIC	BIBLE SCRIPTURE	PICTURE VIEWER 3	SOLITIARE PUZZLE 4	FORM MAKER
TARZAR IV	BASEBALL CARD ORG.	VOCABULARY QUIZ	TANK SHOWDOWN	512K RAM DRIVES 3	DRAGON ESCAPE
ADDRESS IT	MUNCHY 3	DROP PACK	TARZAR 5 ADVENTURE	FOOD FACTORY 3	PLANE EXPLORER
ANIMAL GRAPHICS	RIVER RAID 3	DOCTOR WHO 3	GENESIS	WHO PART 2	WHO PART 3
* SKYWAY	* CUBER	* GRABBER	* FOURCUBE	* TRAPFALL	* MS. MAZE
ISSUE #97, JULY 1990	**ISSUE #98, AUG. 1990**	**ISSUE #99, SEPT. 1990**	**ISSUE #100, OCT. 1990**	**ISSUE #101, NOV. 1990**	**ISSUE #102, DEC. 1990**
FCC EXAM HELPER	FLIPPY TUTORIAL	IDEA TREE	ZOT MIND GAME	GOSSIP	REAL ESTATE ANALYSIS
EMPLOYEE SCHEDULE	RECIPE/MEAL/GROCERY	MASTER'S LABYRINTH	SUPER TEXT EDITOR	COST ESTIMATOR	COCO PAYROLL
FLYING SAUCER ADV.	DUNGEON MAZE	MORSE CODE	SUPER TEXT FORMATTER	SALON INVENTORY	DR. STRANGE ADVENTURE
SPEECH AID	SNOW SKI	TELESALE CONTACTS	TURTLE SEARCH 3	PERSONAL STATISTICIAN	TANNING SALON
MOWER MADNESS	HOT LOAD	PROMISED LAND QUIZ	COAST TO COAST	BINGO CALLER 3	TREASURE HUNT
JET TRIVIA	SHIP WAR	PROPOSAL WRITER	EASY BASIC 3	ALAIX - 3	RIVERBOAT BLACKJACK
INSTANT CALENDAR	ERROR TRAP	ALPHABET SCRAMBLE 3	DRAGON ADVENTURE	DEATH HUNT	TURTLE RACE 3
WORLD MASTER 3	SPACE WAR 3	MAGIC PUMPKIN	WRITE OUT UTILITY	512K/RAM/CATALOGER 3	SUPER SCREEN DUMP
AIR WAR 3	MAZE MASTER 3	DR. WHO3 PART 4	TRYTHIS	DUCK HUNT	KEYBOARD REMAP
* SKRAMBLE	* WIZARD'S DEN	* FANGMAN	* DONUT DILEMMA	* THE FROG	* CRASH

Last 12 Issues Shown Here

* TOM MIX GAMES

SINGLE ISSUE	$ 8.00
2-5 ISSUES	$ 6.00 EA
6-10 ISSUES	$ 5.00 EA
11 OR MORE ISSUES	$ 4.50 EA
ALL 102 ISSUES	$ 260.00

PURCHASE 20 ISSUES AND RECEIVE A FREE 6 MONTH SUB

MAIL TO:
T&D Subscription Software
2490 Miles Standish Drive
Holland, Michigan 49424
(616) 399-9648

Call or write for a FREE catalog !

VISA MasterCard

• WE SEND 1ST CLASS - NO CHARGE •
• PERSONAL CHECKS WELCOME •

Back Issues

PLEASE CIRCLE
TAPE OR DISK

1	9	17	25	33	41	49	57	65	73	81	89	97
2	10	18	26	34	42	50	58	66	74	82	90	98
3	11	19	27	35	43	51	59	67	75	83	91	99
4	12	20	28	36	44	52	60	68	76	84	92	100
5	13	21	29	37	45	53	61	69	77	85	93	101
6	14	22	30	38	46	54	62	70	78	86	94	102
7	15	23	31	39	47	55	63	71	79	87	95	
8	16	24	32	40	48	56	64	72	80	88	96	

1 YEAR SUBSCRIPTION ONLY $ 60.00 !

Name______________________

Address______________________

City______________ State_____ Zip______

Credit Card #______________________

Expires______________________

TOTAL AMOUNT $______________________

Public Domain

Single Disk/Tape..$8.00 each
2- 5 disks/tapes....$6.00 each
6 or more..............$5.00 each
All 53 disks/tapes $145.00

CIRCLE ISSUES DESIRED

M1	GR1	E1	U1	GA1
M2	GR2	E2	U2	GA2
M3	GR3	E3	U3	GA3
M4	GR4	E4	U4	GA4
M5	GR5		U5	GA5
M6	GR6	H1	U6	GA6
M7	GR7	H2	U7	GA7
	GR8	H3	U8	GA8
A1	GR9	H4		GA9
A2	GR10			GA10
	GR11			GA11
T1	GR12			
T2	GR13			
T3	GR14			

PLEASE CIRCLE
TAPE DISK

NEW FOR '91

SUNDOG SYSTEMS

The Quest for THELDA

Better men than you have failed the quest, but the hand of the good princess Thelda is too much to resist! Stolen by the evil necromancer Divinax, she has been secreted far from prying eyes and it is now up to you to regain the pieces of the mystical Life Force and save the fair lady. The Quest for Thelda is an outstanding arcade program that combines action and adventure to bring you an incredible video game playing experience. Travel throughout over 500 different screens searching for magical objects and spells, fighting horrible creatures, and gathering the various parts of the Life Force guarded by ever-increasing terrors. You'll enjoy the highest quality 320x200 resolution 16 color graphics and digital sound effects that you've come to expect from Sundog Systems. The game play is fast and furious, and only you can rescue the princess and complete the **LEGEND OF THELDA!** Req. 128K CoCo 3, disk drive, and joystick (2-button joystick supported)

$34.95

Hint book now available for Thelda! Contains clues, maps, and even a BASIC program to modify your own characters! **$4.95**

★ THE ★ CONTRAS

In the year 2671, a mysterious object landed on Earth, heralding the alien invasion of the Red Falcon. Only two courageous soldiers have a hope of stemming this alien infestation. Armed with your military training and the latest weapon technology, you and a friend must fight back the horde. Sundog Systems presents our second 512K game, **The Contras**. You'll see why we decided to use 512K on this project! Amazing graphics with 320x225 resolution, 16 color full screen animation and horizontal smooth scrolling! Background digital sound effects and real-time music! One or two player action AT THE SAME TIME! You'll agree that this is one of the best arcade games yet for your 512K CoCo 3. Needless to say, playing hero won't be easy; playing The Contras will be tougher! Req. 512K CoCo 3, disk drive, and 2-button joystick.

$34.95

Crystal City

The superior technology of the evil Overlords has ensnared yet another innocent planet; however, this one contains the Crystal City, bastion of democracy and independence. Earth has had enough! Break through level upon level of their virtually invulnerable defenses, defeat the end guardians, and make your way ever closer to freeing the slaves of the Crystal City! This amazing arcade game is an achievement in its own right. Full 128K/512K utilization! Super fast horizontal hardware scrolling on the 128K computer! Wild digital sound effects and background music score! 30 minutes, or over 30 megabytes, of non-repeating 320x200 resolution, 16 color graphics! This game has it all, order a demo disk to see it first! Can you save the Crystal City from unbeatable foes? Req. 128K CoCo 3, disk drive, and joystick.

$34.95

Zenix

You have been chosen to pilot the spacecraft wielding the plutonium plasma laser against the nefarious insectoids, the Zenians. These bugs have been ripping off the space trade lines for years; now they're about to find a victim who can fight back! Beat back the swarm of pests to their home planet Zenix, and use the laser to end the menace once and for all. Zenix is a lightning fast arcade game for the 128K CoCo 3. The 320x225 16 color graphics are amazing, as are the digital background music score and effects, all on a 128K computer! The fast action and game play will astound you in this GALAGA of extermination. Plug in your joystick and strap in for the ride of your life! Call to order a demo disk to see the action before you buy! Req. 128K CoCo 3, disk drive, and joystick.

$29.95

Visa, Mastercard, Check, Money Order, and COD (USA only, please) accepted. All foreign orders must be sent in US currency Money Orders. Include $2.50 for shipping in USA and Canada. $5.00 Foreign. $3.00 extra for COD orders. PA residents add 6% sales tax. Dealer inquiries welcome. Authors: we're looking for new software!

SinistaaR

You've raved about this 512K arcade game! The graphics and animation are amazing! You've got to hear the digital sounds! 512K (three disks) packed with excitement. 512K CoCo 3 only. $34.95

KYUM-GAI TO BE NINJA

This martial arts arcade game remains a CoCo 3 phenomenon. One of our best sellers to date! Play the incredible combat experience that you've been missing, and call for the availability of the OS-9 version. 128K CoCo 3 action. $29.95.

OS-9 Version Now Available

SOUNDTRAX

This 128K/512K sound sequencing system has become an immensely popular program. Create musical scores using polyphonic digitized sounds and a user-friendly point and click editor. Call to order the SoundTrax demo program, and get 10% off your purchase with receipt. $34.95

New Available: SoundTrax instrument sample disks! Guitars, strings, woodwinds, and just plain weird instruments for your composing pleasure. Completely compatible with the SoundTrax system. 3 flippy (2 sides) disks full of sounds. $12.95 each or $29.95 for all three.

VISA MasterCard

ALSO AVAILABLE:

Hall of the King 1, 2, or 3	
CoCo 1 - 3	$29.95 ea.
Hall of the King Trilogy	$74.95
White Fire of Eternity	
CoCo 1 - 3	$19.95
Dragon Blade	
CoCo 1 - 3	$19.95
Champion	
CoCo 1 - 3	$19.95
Paladin's Legacy	
CoCo 1 - 3	$24.95
Warrior King	
CoCo 3	$29.95
In Quest of the Star Lord	
CoCo 3	$34.95
Hint Sheet	$ 3.95
Kung-Fu Dude	
CoCo 1 - 3	$24.95

Sundog systems

We've Moved!

P.O. Box 766
Manassas, VA 22111
703/330-8989

Being online never felt so fine.

The most interesting online conferences. 10,000 programs you can download for your own use. All the most popular services. Two membership plans: $6 an hour or $1 an hour with the 20/20 Advantage.

Get online. And check out our lineup.

Free membership.

Via modem, 1-800-365-4636.

Press return. At password,

type RB22.

DELPHI

The more interesting online service.

Voice, 1- 800-544-4005 or 617-491-3393.