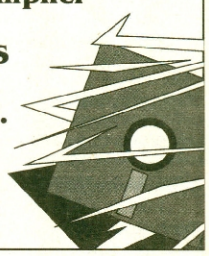
Serving the CoCo Community for
The RAINBOW 12 YEARS
THE COLOR COMPUTER MONTHLY MAGAZINE
April 1993 Vol. XII No. 9 Canada $4.95 U.S. $3.95

Product Review: A C compiler for RS-DOS on Page 6.



Feature Program

A Database for OS-9 Files

by Philip G. Scherer

After you accumulate many OS-9 programs, it's pretty easy to forget what some of them do. It's also inconvenient to search disk after disk, looking for a specific file or program you know is stored *somewhere*. *DBOS9* is a database designed to make such things simpler by allowing you to store the names and descriptions of your OS-9 files and programs in a central location.

For operation, *DBOS9* requires a CoCo 3, OS-9 Level II, at least one disk drive, and the OS-9 modules `del` and `copy`. In addition, if you plan to enter the listings as printed here, you'll need a C compiler and the CGFX library written by Mike Sweet. (This library is available in the OS9 Online sig on Delphi. Mike's username is DODGECOLT.) Alternatively, the compiled, ready-to-run program is on the April 1993 RAINBOW ON DISK.

***DBOS9* is a database designed to make such things simpler by allowing you to store the names and descriptions of your OS-9 files and programs in a central location.**

Up 'n' Running

Before you start *DBOS9*, you need to give it a place to store the information you enter. Create a new subdirectory named `BASE9` in the root directory of your `/dd` device. (Those who are using the OS-9 BBS database from the October issue don't need to do this; this database uses the same data directory as that one.) To do this, simply enter `makdir /dd/BASE9` at the OS-9 prompt.

With the `BASE9` data directory in place, make sure the executable program (`dbos9`) is in your `/dd/CMDS` directory, along with the OS-9 `del` and `copy` commands. Also make sure `dbos9`'s execute attributes are set. If not, you can accomplish this by entering `attr /dd/cmds/dbos9 e pe` at the OS-9 prompt.

To execute the program, just enter `dbos9`. The first thing the program does is look for the database file, named `os9`, in the `/dd/BASE9` directory. It won't find this file the first time it is executed, so it creates the file automatically. *DBOS9* also creates a database-keyword file, named `keyword.dat`, the first time it is executed.

Once *DBOS9* ascertains these files are in place in the `BASE9` directory, the main database menu appears on the screen. This menu provides four options and is "hot-keyed," which means that you don't have to press ENTER after making a selection. The Search menu, which we'll discuss in a moment, works the same way. Some of the stand-alone entries at program prompts, however, do require that you press ENTER.

Entering Data

To enter data for a single record (information about a specific file or program) in the *DBOS9* database, select Option 2 (Enter) from the main menu. In the upper-left corner of the screen you'll see a box with the available keywords. If the program or file for which you are entering data fits one of these keywords, simply type the appropriate keyword number and press ENTER. If none of the keywords on the list seem appropriate, or if you have not yet entered any keywords, press the correct number for the New Keyword option that appears on the list, then enter the new keyword.

DBOS9 keywords may be up to 13 characters in length and may include spaces. It

Database *continued on Page 10.*

Feature Program

CoCo Calculates Net Worth

by Charles Kiedaisch

I wrote *Financial Statement* as a means of tracking personal income and expenses. In addition, the "spreadsheet" printed by the program enables me to quickly determine my net worth. *Financial Statement* is designed to work on any CoCo with at least 16K and Extended BASIC. To use it, first enter the program listing and save it to tape or disk, then run it.

When you run *Financial Statement*, you are asked to make sure your printer is online. This is important, as the program prints data as it is entered, allowing for an unlimited number of entries. When prompted, enter the current date. After this, the program's main menu appears.

To enter data, first select from the main menu the type of entry you are making (weekly, monthly or yearly income or expense). You are then asked for a short description of the item and the amount for the specified period. After you have entered this data, the printer immediately prints it. Make sure you enter all sources of regular income, as well as all regular and anticipated expenses (bills go here).

Option 7 on the main menu allows you to enter standing assets. For instance, if you have an IRA or perhaps a savings account, the current balance should be entered using this option. Any item that contributes to your net worth is considered an asset.

Net Worth *continued on Page 7.*

In this issue:

THE RAINBOW

Editor and Publisher
Lawrence C. Falk

Managing Editor
Cray Augsburg
Associate Editor
Sue Fomby
Submission/Reviews Editor
Julie Hutchinson
Technical Editor
Greg Law
Technical Assistant
Ed Ellers
Contributing Editors
Martin Goodman, M.D.,
Tim Kientzle, Eddie Kuns
Consulting Editor
Laura Falk
Art Director
O'Neil Arnold
Designers
Sharon Adams, Heidi Nelson
Editorial Director
John Crawley

Falsoft, Inc.
President
Lawrence C. Falk
General Manager
Judi Hutchinson
Asst. General Manager for Finance
Donna Shuck
Asst. General Manager for Administration
Tim Whelan
Administrative Asst. to the Publisher
Ellen Patterson

Bookkeeper/Dealer Accounts
Debra Wilson
Customer Service Representative
Lauren Yates
Business Assistant
Shannon Yoffe
Traci Christophers
Corporate Business Technical Dir.
Calvin Shields
Chief of Building Security & Maint.
Lawrence Johnson

Advertising:
Western Sales Director
Ira Barsky (312) 587-1818
Eastern Sales Director
Kim Lewis (502) 228-4492

For RAINBOW Advertising and Marketing office Information, see Page 15.

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; GST No. R125434092. ● ● 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.

Letters to THE RAINBOW

Taking Data to Another OS

Editor:

Is it possible to download a CoCo 3 disk to a PC-compatible, and is there a program to do this?

Wayne A. Johnson
Route 2
Thamesford, ON N0M 2M0
Canada

Sure. A number of utilities have been developed over the years for transferring data from the CoCo to MS-DOS computers. While most of the commercial products are no longer available, several shareware and public domain offerings have been uploaded to online information services such as Delphi and CompuServe. Also, the June and July 1986 issues of THE RAINBOW contain programs by Marty Goodman that are designed to handle the transfer.

Another means of transferring data is to use a null-modem cable between the CoCo serial port and a serial port on the PC-compatible. Then use communications software on each end, and transfer the file by uploading from the CoCo and downloading with the MS-DOS machine.

Regardless of the route you follow, remember that in most cases CoCo programs *cannot be executed on the PC-compatible. The biggest exception to this is BASIC programs, which can be edited to work with the PC's specific flavor of BASIC, whether it is GW-BASIC or QBASIC. Still, such programs need to be saved in ASCII format on the CoCo (before the transfer) in order to be decipherable by the PC.*

PMODE4 Screen Dumps

Editor:

I have a CoCo 3 with a CM-8 monitor, two FD-502 disk drives and a Star Micronics NX-1001 printer. I got involved with the CoCo about three years ago and have taught myself some BASIC programming, mostly with help from THE RAINBOW. I don't have a modem, so I don't have access to Delphi. I copy most of the programs from the magazine and learn from them.

Looking through back issues, I came across your PMODE4 screen dump (May 1992) and typed it in. I had some trouble at first, but after I changed the DIP switches, it worked fine. There is one thing that I don't understand: why does it print an "x" at the beginning of each row?

Calvin Wilcox
6626 Edgemoor Avenue
Solon, OH 44139

As explained in the article, two different programs for producing screen dumps were presented. One is designed to work with Radio Shack printers in the Tandy mode, and the other is meant for use with IBM/Epson-compatible printers. The Star NX-1001 falls into the latter category. However, there may be some minor differences between the IBM/Epson-compatible and the NX-1001 control-code sets. Since most printers produce a garbage character when they encounter a control code they don't understand, we bet this is where the problem lies. Carefully check the assembly-language listing for the control codes it uses, and compare these with the appropriate codes in your printer manual.

A Ham in Need, Indeed

Editor:

I have a CoCo 2 for which I want to get amateur-radio and packet software, and Morse-code programs. I am also looking for a disk drive and a printer for this CoCo.

On a related note, I have a CW/RTTY card that fits into the CoCo ROM slot. The cable has come unsoldered on this unit, and I don't have a wiring diagram to help me put it back together. The cartridge was made by Kantronics. Can anyone help me with this?

Tony Byrum
2002 2nd Avenue S.
Ft. Dodge, IA 50501

We imagine several CoCo users are still into amateur radio. Perhaps another reader can point you in the right directions.

Looking for the Hershey Font

Editor:

I am looking for a copy of the public-domain Hershey font for use with William Barden's utilities published in the March 1988 issue of THE RAINBOW. I have written Mr. Barden, but he has been unable to provide me with a copy of the font. If one of your readers can provide me with this font set or tell me where to get it, please write to me at the address below.

Trevor Boehm
77 Inwood Crescent
Winnipeg, MB R2Y 1A2
Canada

Wants to Save Screens

Editor:

Is there a program (other than one in machine-language) for saving the screen? I'd like one like that mentioned on Page 147 of the March 1987 issue of THE RAINBOW.

Denis Benjamin Marcil
222 Lomas
Sherbrooke, PQ J1J 2R3
Canada

The program you noted in your letter originally was bundled with a complete printer package by Dayton Associates (9644 Quailwood Trail, Spring Valley, OH 45370, 513-885-5999). For other screen-printing software, see the May 1992 issue of THE RAINBOW.

Sailing Off to C

Editor:

I've been reverse engineering the OS-9 C compiler (6809) library into its original C source code. I don't know how close my source looks to Microware's original, but it compiles into the same object code, and that's good enough. I recently discovered that this compiler and the `c.asm` macro assembler are themselves written in C, and I've considered reverse engineering the C source for them also. However, before I start, I was curious if anyone has already done all the work?

To be honest, it isn't so much the desire to have the original C source for the compiler, assembler and linker as it is simply to have a version in source form that I can customize. I've ported a Small-C (a subset of K&R C) compiler, but it isn't very good. The only full C compiler I know of is the GNU C compiler, which is far too big. Does anyone know of any C compiler (any processor — I'll port it) or a macro 6809 assembler in source form?

Carey Bloodworth
1601 North Hills Blvd.
Van Buren, AR 72956

Building a New System

Editor:

I have just gotten back into the CoCo world and have managed to find a CoCo 3 and an FD-502 disk drive. I have also been given a Tandy printer, but it's a parallel printer. Can I use Tandy's serial-to-parallel port converter (Cat. No. 26-2829) on a CoCo 3? I also need a Multi-Pak Interface. Can you tell me where I can find one.

Marcus Springer
101 S. Central
Connersville, IN 47331

You should able to use Tandy's serial/parallel converter, but you'll have to build a special cable to go between the CoCo's 4-pin serial port and the converter. A better solution would be to get a converter designed specifically for the CoCo. Both Owl-Ware (see the ad on the back cover of this issue) and Dayton Associates (see our response to Denis Marcil's letter in this issue) offer such devices.

The Multi-Pak Interface was hard to find even before Tandy officially discontinues it. After that time, the MPI became impossible to find. At this point, hope another reader has one he'd be willing to part with.

THE RAINBOW welcomes letters to the editor. Mail should be addressed to: Letters to Rainbow, The Falsoft Building, 9509 U.S. Hwy 42, P.O. Box 385, Prospect, KY 40059. Letters should include the writer's full name and address. Letters may be edited for 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, enter `RAI` to get to the Rainbow Magazine Services area of the SIG. At the RAINBOW> prompt, enter `LET` to reach the LETTERS> prompt, then select Letters for Publication. Be sure to include your complete name and address.

Be a computer programmer!

Only NRI gives you hands-on training with the latest programming tools:

■ A 486sx computer with 80 meg hard drive ■ Windows
■ Visual Basic ■ Power C ■ QBasic ■ MS-DOS
■ And much more!

Only NRI at-home training gives you real-world programming skills in three in-demand languages: QBasic, C, and Visual Basic, today's hot new language designed for writing popular Windows applications. Best of all, you get hands-on training with a powerful new 486sx-based computer system, complete with 80 meg hard drive, Windows, and professional programming software you keep!

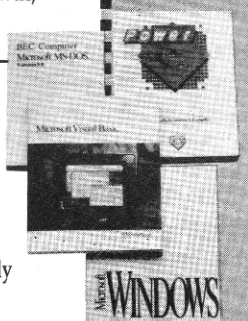
NRI, the leader in at-home computer training, shows you how to take advantage of today's newest programming opportunities

Get in on the ground floor of one of today's fastest-growing career fields: computer programming. The Bureau of Labor Statistics forecasts that job opportunities for programmers will increase much faster than average over the next 10 years, with as many as 400,000 new jobs opening up by 2005.

And the fastest-growing segment of programming jobs will be PC programming, fueled by the phenomenal popularity of Windows, the efficient power of C, and the ascent of exciting new languages like QBasic and Visual Basic.

Now, with NRI at-home training, you can get the new skills you need to build a top-paying career — even a full- or part-time business of your own — in this high-growth, high-opportunity field.

NEW! The only programming course that includes a powerful 486sx-based computer, 80 meg hard drive, Windows, Visual Basic, and more — all yours to keep!



Right from the start, NRI gets you actively involved in the challenge of real-world programming. Step by step, you learn to create the kinds of full-featured, powerful programs today's employers and clients demand...including programs designed for use in a Windows environment!

Only NRI gives you first-hand programming experience with a state-of-the-art 486sx mini-tower computer system, complete with hard disk drive, a full megabyte of RAM, high-density floppy drive, mouse, monitor, and more — all yours to train with and keep!

Plus you explore the extraordinary capabilities of three in-demand programming languages. You learn to design, code, run, debug, and document programs in QBasic, C, and Visual Basic. Best of all, since Visual Basic is specifically designed for creating Windows applications, you learn to generate fully functioning Windows programs, complete with text boxes, command buttons, and other sophisticated graphical interface elements.

No previous experience necessary

Train with NRI, and immediately start getting the money-making job skills you need to be a computer programmer — no matter what your previous background.

NRI's step-by-step lessons and hands-on programming projects help you first master the design concepts used every day by successful PC programmers. Then, with the support of your experienced NRI instructor, you quickly move on to learn programming in three of today's hottest languages.

By the time you complete your course, you have a clear understanding of programming methods, languages, and techniques... and you're ready to handle any programming task with confidence.

Send today for your FREE catalog

See how NRI at-home training gives you the programming know-how, the computer, and the software you need to get started in this top-paying field. Send today for your FREE catalog!

If the coupon is missing, write to us at the NRI School of Computer Programming, McGraw-Hill Continuing Education Center, 4401 Connecticut Avenue, NW, Washington, DC 20008.

SEND COUPON TODAY FOR FREE NRI CATALOG!

NRI Schools McGraw-Hill

McGraw-Hill Continuing Education Center
4401 Connecticut Avenue, NW, Washington, DC 20008

☐ For career courses approved under GI Bill, check for details

☑ **Check one FREE catalog only**

☐ **COMPUTER PROGRAMMING**
☐ PC Applications Specialist
☐ Programming in C++ with Windows

Other Computer Career Courses
☐ Microcomputer Servicing
☐ Desktop Publishing
☐ Bookkeeping and Accounting
☐ Computer-Aided Drafting

Name (please print) Age

Address

City/State/Zip

Accredited Member, National Home Study Council

5410-0493

DELPHI BUREAU

EDDIE KUNS

File-Transfer Protocols

For uploading and downloading, Delphi supports most of the commonly used file-transfer protocols in addition to a few relatively obscure ones. While older file-transfer protocols were designed to transfer only one file at a time, some protocols are "batch" protocols, which allow you to transfer more than one file in a single session. Batch protocols also automatically send the filenames of the files to be transferred as well as some additional file attributes such as file size and protections.

Most modern file-transfer protocols are *error-checking*; they divide the file into several "blocks" that may be of either fixed or variable size. If a block is not sent correctly, the receiving end "complains" that the block arrived in error, and the block is sent again. This error checking ensures that the file is transmitted correctly even if there is noise on the communications line. As long as such a file-transfer protocol finishes sending all the blocks, the file was transferred successfully — even if errors were encountered along the way.

While the block size used by a protocol does not affect the contents of a file, it does have an impact on the speed at which the file is transferred, especially for people who use Sprintnet or Tymnet to connect to Delphi. The Sprintnet and Tymnet networks have a long *latency*, or response time. This latency varies with the time of day and current network use — both Sprintnet and Tymnet are used by many computer systems other than Delphi.

The result of network latency is that there is a definite and noticable pause between transmitted blocks. Since transfer protocols that use smaller blocks have to send a greater number of blocks, they also spend more time waiting for the network to respond between blocks. This results in increased transfer time. As an example, the Xmodem protocol uses 128-byte blocks, while Ymodem (generally) transfers a file 1024 bytes at a time. A file that is 4096 bytes long would require four Ymodem blocks, thus having three pauses between the blocks. This same file would be sent in thirty-two blocks by Xmodem, resulting in thirty-one pauses between blocks. The moral: Xmodem is almost always considerably slower than Ymodem when used on networks like Sprintnet and Tymnet. If you call Delphi directly, avoiding network latency, you will not notice a very large time difference between the two protocols. (Ymodem results in fewer blocks, but each block is larger and takes longer to send than an Xmodem block.)

Most transfer protocols are called "half-duplex" because the sender sends one block then waits for an "I got it" from the receiver before sending more blocks. However, there are two ways to avoid the delay of waiting for a response. One of the easiest methods is called *windowing*. A windowing protocol sends blocks even when previous blocks are not yet acknowledged by the other end. For example, WXmodem provides a four-block window. This means if the other end has returned an "OK" for Block 32, it can send blocks 33, 34, 35 and 36 without waiting for a response. If the communications speed is low enough, a response for Block 32 would arrive before transmitting Block 36. In this case, windowing would allow continuous transmission with no pauses. The protocol stops sending data only when *several* blocks of data have been sent without response from the other end.

The other method of avoiding network latency is the more-complicated *streaming*. Streaming protocols, such as Zmodem, continue sending data, often without requiring any acknowledgement from the other end. The transfer is stopped and data retransmitted only on a request from the receiving end.

A final important feature of file-transfer protocols is whether or not they are *network transparent*. A protocol that is not transparent, such as Xmodem, requires an 8-bit connection to be able to send 8-bit binary files. A protocol that is (or can be set to be) transparent, such as Kermit, encodes any characters the network may be unable to transmit into multiple characters that the network *can* transmit. For example, many networks use the XON and XOFF characters — Control-Q and Control-S, respectively — to start and stop transmission (referred to as *flow control*). If you transmit a binary file that contains an XON or XOFF character across a network that uses XON/XOFF flow control, that character will be "consumed" by the network as a flow-control character with possible unexpected side effects. (Fortunately, Sprintnet and Tymnet do *not* use XON/XOFF flow control.) This is why Kermit and some other protocols that can encode control characters into network-transmittable characters are so important.

We've discussed some of the background for file-transfer protocols this month. Next time we'll look at how each of the common protocols works as well as when and where each should be used.

Eddie Kuns is pursuing a doctorate in physics at Rutgers University. He lives in Aurora, Illinois, and works as a programmer and researcher at Fermilab. Eddie is the database manager of the OS-9 SIG and can be reached online as EDDIEKUNS.

Telnet Echo Hints

Last month I described how you can use Telnet to get from Delphi to other computers. Many computer systems allow you to enter your password without it being echoed to your screen. However, if you are connected to Delphi using Sprintnet or Tymnet, you may see your password echoed to the screen when using Telnet to connect to another computer. This happens because Sprintnet and Tymnet echo the characters to *your* computer — not Delphi, and not the computer to which you are connecting.

If you want Delphi itself (or the computer to which you want to connect) to echo the characters — and you usually will want to when you use Telnet — use the `/ECHO HOST` command to temporarily change your echo setting. If you are unsure what your current echo setting is, enter `/ECHO` to find out.

Setting host echo is useful for more than invisible passwords: If you want to use a full-screen editor on the computer you are connecting to, you need to set host echo before using Telnet from Delphi. Remember that, for the same reasons, you need to select host echo if you want to use *Delphi's* full-screen editors.

Uploads At a Glance

In the OS9 Online Applications database, **Mike Guzzi** (MIKE GUZZI) released `catl`, a program that allows you to create catalogs for OS-9 *Profile*. He also uploaded a utility that works with `catl` to catalog `.GIF` files. **Michael Dalene** (MDALENE) contributed `zeroadd` to solve a problem he encountered with `catl`.

In the System Modules database, **Michael Graffam** (ILLUSIONIST) released new window descriptors for those who want to use more windows than were provided by Tandy. **Erich Schulman** (ESCHULMAN) contributed an `OS9P4` module that adds a new system call to dump the 6809's registers.

In the Programmers Den database, **Ken Scales** released the first version of an OSK terminal-information library; this is useful to programmers writing or porting *Curses*-based programs. `terminfo`, like `termcaps`, is a way of describing how different terminals perform various functions such as cursor positioning. If you are having trouble creating complicated C variable declarations, you'll be interested in `CDECL`, uploaded by **David Graham** (NIMITZ). This program deciphers C declarations and can also create them from English descriptions.

In the OSK Applications database, **Tim Kientzle** (TIMKIENTZLE) released a complete OSK port of *TeX* (including *LaTeX* and *BibTeX*, as well as the many other parts of the system). *TeX* is a typsetting system that can be used to produce high-quality output on dot-matrix and laser printers.

In the OSK System Modules database, **Mark Griffith** (MARKGRIFFITH) released the latest serial drivers for the MM/1. **Mike Sweet** (DODGECOLT) uploaded the latest version of `windio` for the MM/1, as well as documentation he has collected from many sources for the features supported by `windio`.

In the CoCo SIG CoCo 3 Graphics database, **Johnny Williams** (DRILLMASTER) uploaded an Elvis puzzle data file to be used with the *Puzzler* program. To use this puzzle, you need `PUZZLER.BAS` from the `COCO 3 PUZZLER` group in the Games & Graphics database.

DATABASE REPORT

OS9 Online:

General Information:
IDE BUS INFORMATION
9MIKE Mike Filipietz
CUSTOM DATE MODULE PROMO
JSUTEMEIER Jim Sutemeier
NEW VIDEOS AVAILABLE
WTHOMPSON Wayne Thompson
OSK/AMIGA/FALCON
MEKEARNEY Michael Kearney
HI-DENS FLOPPY CONTROLLER HACK
DSRTFOX Francis Swygert

Applications (6809):
NEW POINTERS FOR MVUE
EARTHER Shawn Driscoll
DATEP: DEMO OF CUSTOM DATE MOD
JSUTEMEIER Jim Sutemeier
ZEROADD: ADDS LEADING 0'S
MDALENE Michele Dalene
GIF CATALOG UTIL
MIKE_GUZZI Mike Guzzi
CATL: CATALOG DISKS FOR PROFILE
MIKE_GUZZI Mike Guzzi
LINFCOUNT: COUNTS LINES ON STDIN
ILLUSIONIST Michael Graffam
CLYDE 2.00 SCREEN SAVER
SANDRIDER Charles West
CATALOG 1.1: DISK FILE CATALOG
MOHRT Tim Mohr
LISTER: FILE LISTING UTIL
RICKGRAY Rick Gray

System Modules (6809):
WINDOWS 14-32
ILLUSIONIST Michael Graffam
OS9P4 REGISTER DUMP MODULE
ESCHULMAN Erich Schulman

Games & Graphics:
CLOAKING KLINGON SHIP (FLI)
GRAPHICSPUB Bob Montowski
VEF2GIF VERSION 1.1
MEYE001 Homer Meyer
LINES: A SCREEN SAVER
MEYE001 Homer Meyer
OS-68K FLICKER ANIMATION
GRAPHICSPUB Bob Montowski

Music & Sound:
REN & STIMPY SOUNDS #2
DEANHOLDER Dean Holder
REN & STIMPY SOUNDS #1
DEANHOLDER Dean Holder

Programmers Den:
TERMINFO FILES FOR OSK CURSES
KSCALES Ken Scales
GCC 1.42
NIMITZ David Graham
CPREP - ANSI C PREPROCESSOR V1.0
JMLSOFT Jim McDowell
CDECL: DECIPHER C FUNC CALLS
NIMITZ David Graham
GUIDE TO PROGRAMMING STYLE
JBUCATA Jason Bucata
TERMCAP DEMO
PAGAN Stephen Carville

OSK Applications:
HDBACKUP 1.5:HARD DRIVE BACKUP
MARKGRIFFITH Mark Griffith
FF 1.01: FIND FILE UTIL
PAGAN Stephen Carville
TEX SYSTEM FOR OSK
TIMKIENTZLE Tim Kientzle
NEW GRAPHICAL CLOCK FOR MM/1
JOELHEGBERG Joel Hegberg
SPHERE: MOLECULAR MODELER
EMTWO Paul M. Fitch, Jr.
BYACC: BERKELEY YACC
NIMITZ David Graham
EXTRACTOR FOR OSK
KEITHBAUER Keith Bauer

OSK System Modules:
NEW MM/1 SERIAL DRIVERS/DESCRIPT
MARKGRIFFITH Mark Griffith
WINDIO REF. MANUAL: 1ST EDITION
DODGECOLT Mike Sweet
WINDIO EDITION 48 FOR THE MM1
DODGECOLT Mike Sweet

Tutorials & Education:
COPYING FILES, OS9->MSDOS
BOBKEMPER Robert Kemper

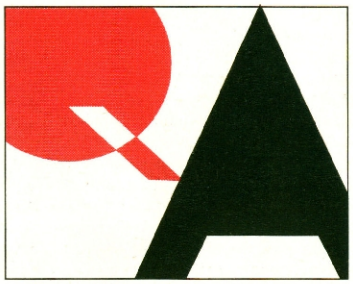
CoCo SIG:

CoCo 3 Graphics:
DOCTOR WHO PIX (DS69-B)
DEANHOLDER Dean Holder
ELVIS.PUZ
DRILLMASTER Johnny Williams

Hardware Hacking:
COCO 2 MAC NULL MODEM
MARTYGOODMAN Marty Goodman
ADD A RESET BUTTON
TERMITE Jim LaLone

Music & Sound:
ORCHESTRA 90 MUSIC
DEANHOLDER Dean Holder

CoCo Consultations



MARTY GOODMAN

64K and the "E" Board

I have a CoCo 1 with an "E" board that does not appear to be modified. The computer has a "32K" button on it. Is it really a 32K computer or can it address all 64K? Does it need to be modified in order to address all 64K? How can I tell if the modification has already been done? If it hasn't, how do I fix it?

Wes Ratcliff (WESRATCLIFF)
Stockton, California

A CoCo 1 "D" and "E" boards must be modified before they can address all 64K of memory the 6809 supports. (The "F", or "NC," board support 64K right out of the box, despite being labeled as 32K units by Tandy.) The modification to the "E" board, allowing access to the upper 32K of memory, is usually pretty easy to spot — look for some bent pins on some IC chips and "odd" wires going between them. On the other hand, a few very dedicated hackers have performed this modification *underneath* the board in such a fashion that it is totally invisible from above. To be sure whether or not your "E" board has been modified for 64K, here's a simple test: use an ohmmeter set to the Rx10 scale to measure the resistance between Pin 5 of the 74LS138 chip (U11) and ground. If the resistance is near zero (a tenth of an ohm or less), the computer has *not* been modified for addressing all 64K of memory. If the resistance is some tens or hundreds of ohms, the modification may already have been performed.

To modify an "E" board CoCo 1 to address a full 64K of memory is a relatively simple procedure. First remove the 74LS02 chip (U29) and bend pins 4, 5 and 6 so they go straight up. Solder a fine wire between Pin 6 and Pin 8, but be careful: Pin 8 remains pointing down and will be reinserted in the socket; make sure you attach the wire at the very top of Pin 8 so as not to cause interference. Now solder an 8-inch length of wire to Pin 4 and another to Pin 5 of the 74LS02. Put a small piece of electrical tape on the metal part of the shield next to where the 74LS02 goes, preventing the bent pins from coming in contact with the wall and causing a short. Now replace the 74LS02 chip in its socket and remove the 74LS138 chip (U11). Bend up Pin 5, then replace the chip in its socket. Trim the wires to length, then solder the wire coming from Pin 5 of the 74LS02 chip to Pin 5 of the 74LS138 chip. Also connect the wire from Pin 4 of the 74LS02 chip to TP1, a staking pin located between the 6809 chip and the 40-pin cartridge connector, near Pin 34 of the 6809. With these modifications, the computer is capable of addressing a full 64K.

It is worth noting that Tandy *sometimes* used "half-good" (known to us as "half-bad") 64K DRAM chips in their 32K "E"-board computers. It's probably best to replace all eight 4164 DRAM chips. These chips are commonly available used for 25 to 50 cents each, and should not cost more than $1 apiece new from a chip vendor.

Adding Drives to the FD-502

I notice the FD-502 disk drive from Tandy has somewhat different power connectors and termination than other 5¼-inch drive systems. Can you tell me how to add an extra drive to the system?

Robert Coates
Sandy Hook, Manitoba
Canada

A The FD-502 uses power connectors that are standard not for 5¼-inch drives, but for modern 3½-inch drives. Its drive termination, too, follows the convention for modern 3½-inch 1.44-megabyte drives — the floppy drive in the FD-502 uses a soldered-in 1000-ohm resistor for termination. If you are adding another brand of floppy drive to the FD-502 case, I suggest the following approach: First, cut off the existing power connector on the spare power cable, and in its place attach a 5¼-inch-style power connector. Make sure you have it wired correctly, for if you reverse the 5- and 12-volt lines, you will destroy the new drive. Then, on the added drive, use a terminator-resistor pack rated not at the usual 150 ohms, but at 470 or 1000 ohms. If you cannot find such a terminator pack, try adding the second drive first *with* its existing resistor pack in place, then without, and see which arrangement works better. Note that since most terminator resistor packs for floppy drives are simply DIP component headers with resistors across them, you may be able to build your own terminator using such a header and seven 1000 ohm resistors.

Using An Unknown Terminal With OS-9

I picked up a "junker" dumb terminal for use with my CoCo 3 under OS-9, but I have no documentation for it and don't know how to set its DIP switches for the proper baud, parity, stop bits and so on. The terminal has two banks of DIP switches and two DB-25 connectors on its rear.

Tony Reed (TONYREED)
Richford, Vermont

A Your best bet is to use a null-modem cable between the terminal and the RS-232 Pak on your CoCo 3, then run a terminal program on the CoCo 3 and experiment with different switch settings on the terminal. (Your terminal may have one DB-25 port for serial communication and another for a printer. These are usually labelled as such. Make sure the cable is connected to the serial port.) When characters you type on the terminal start appearing on the CoCo 3 screen, you're on the right track.

Typically there are three to five DIP switches for setting set the baud, and one or two for setting the parity and word length for each port. By playing around with the switches and matching settings with your terminal program, you can probably puzzle out most of the switch settings. Once the baud is determined, parity and word length can be deciphered in the same fashion. Finally, by playing with control-character sequences, you may be able to determine some of the basic screen and cursor control functions. This will be easier if you have some general knowledge of what control codes are supported on similar terminals.

Smartwatch and the Tandy Controller

How do I use a Smartwatch in the 24-pin ROM socket of a Tandy disk controller?

Rick Ulland (RICKULAND)
Milwaukee, Wisconsin

A Art Flexser offered some help on this one. He suggests you plug the Smartwatch into the socket so that pins 1, 2, 27 and 28 overhang the top of the socket (put Pin 3 of the Smartwatch into the hole for Pin 1 of the 24-pin socket). But be sure to jumper Pin 28 of the Smartwatch to Pin 26 of the Smartwatch, or to some other source of +5 volts. You can then plug your 24-pin Tandy Disk BASIC ROM into the smartwatch socket, making sure that Pin 1 of the ROM chip goes into the hole for Pin 3 of the Smartwatch. (Of course this, in turn, connects to Pin 1 of the original Tandy ROM socket.)

Killing Call Waiting

Calls coming through because of Call Waiting are interrupting my modem communications. How can I fix this without losing Call Waiting's benefits?

Charles A. Marlow (CHARLESAM)
N. Massapequa, New York

A In most areas, dialing *70 disables Call Waiting, giving you a new dial tone. In other areas, you must dial 1170. In either case, Call Waiting is disabled only for the single call in which it is used. If your modem is Hayes-compatible, you can make a call by entering the following sequence:

```
AT DT *70,,nnn-nnnn
```

where *nnn-nnnn* is the phone number you want the modem to call (if required, use 1170 in place of *70). The two commas tell the Hayes-compatible modem to pause for a few seconds while waiting for Call-Waiting disable to take effect.

Disk Controllers and Drives

Will the Tandy FD-502 drive system work using a 26-3029 disk controller?

Robert L. Fansler, Jr. (ROBERT191)
Chattanooga, Tennessee

A Yes. As a matter of fact, any 5¼-inch 180K, 360K or 720K drive will work fine with any Radio Shack controller. Note, however, that the first controller Radio Shack released (Cat. No. 26-3022) does not work with the CoCo 3. That first disk controller can be recognized by the facts that all the chips in it are socketted and that there are three adjustable potentiometers on the circuit board. Indeed, the 26-3029 controller (the second controller Tandy released) is one of the best Tandy ever made for the CoCo. Note, too, that most third-party disk controllers also work fine with most drives, though a few (such as the first controller J&M made) also have compatibility problems with the CoCo 3. Also, any 3½-inch 720K drive can be used with most CoCo disk controllers. In addition, 3½-inch 1.44-megabyte drives can be used, but only in the 720K mode.

Is the Hard Drive Busy?

I have a Tandon 252 10-megabyte hard drive that I am using with my CoCo 3. This drive does not have a "busy" light on it, and I'd like to know how to add one?

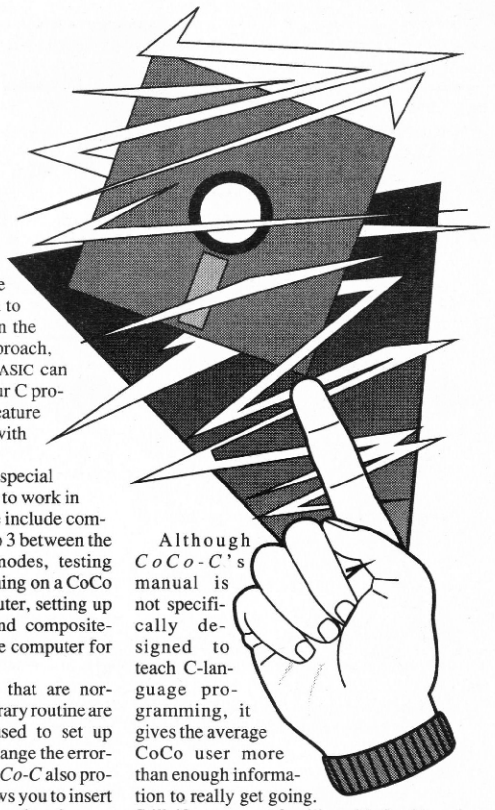
Steven Taulborg (TAULBORG)
Reynoldsburg, Ohio

A MFM drives have one 34-pin edge connector and one 20-pin edge connector on them. Looking at the 34-pin edge connector, pins 25, 27, 29 and 31 are, respectively, the Drive Select 1, 2, 3 and 4 lines going from the MFM controller board to the hard drive. Thus, it is likely that if you had an LED powered via a transistor or one or two CMOS inverter gates (to take the load off the select line), you could use one of those select lines (probably the Drive 1 select line) as your "drive in use" LED. You'll have to play around a bit depending on whether the select line is active low or high (use a logic probe first to check this out).

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 THE RAINBOW's CoCo SIG. His non-computer passions include running, mountaineering and outdoor photography. Marty lives in San Pablo, California.

Product Review

CoCo-C: A C Compiler for RS-DOS



Even though Microware's C compiler has been available to CoCo users for years, those who prefer the Disk-BASIC environment haven't had a reliable implementation of C for some time. Yes, there have been several "small C" compilers available through BBSs, and even a few C interpreters. But a well-documented version of C has been needed for Disk BASIC for quite a while.

CoCo-C gives the Disk BASIC user the ability to write C source code, then compile that code into binary programs that can be loaded from disk or tape and executed. *CoCo-C* isn't a complete implementation of C since it supports only character and integer data types and doesn't allow for complex data types. But *CoCo-C* does provide most of the environment that experienced C programmers are used to seeing.

CoCo-C is delivered on a flippy disk; the system for CoCo 3 users is on one side, and the programs for use with the CoCo 1 and 2 are on the back. (A flippy disk has files on both sides, but you don't need a double-sided drive — you turn the disk over.) The primary differences between the *CoCo-C* systems for the CoCo 3 and the CoCo 1/2 appear to be limited to the text editor and the program-initialization routines. Otherwise the files appear to be the same.

The Software

CoCo-C includes all the tools you need to write your own programs in C. Included for CoCo 3 users is Bob van der Poel's *Ultra Editor*, which is a joy to use. This is a powerful text editor, optimized for program editing, that includes such features as dual text buffers that can be opened at the same time, as well as commands to transfer data between them. Although it isn't a complete word processor, *Ultra Editor* comes close. Full-screen editing in 40 or 80 columns and simple two-key commands allow you to quickly prepare your C source code for the compiler. *Ultra Editor* also supports macros, allowing you to customize the editing environment and perform repetitive functions easily. (This editor can also be used for other languages, such as assembly language and BASIC.)

The 32-column line editor supplied for CoCo 1/2 users is much less complex, yet it provides enough features to edit any program. Still, if you need more-powerful editing capabilities, consider using a word processor that supports straight ASCII files, such as *Telewriter*.

Also included with *CoCo-C* is an assembler that compiles to standard Motorola-syntax assembler code. By eliminating CoCo-specific functions, you can use *CoCo-C* on your Color Computer to develop ROMable code for just about any 6809-based system.

Programs written using *CoCo-C* can be interfaced with programs in other languages (i.e., *CoCo-C* programs can be called from BASIC as subroutines). At the same time, a unique interface between *CoCo-C* and the BASIC ROMs in the computer gives your C programs all of the power of BASIC; the `bascmd` function allows you to make calls to the routines in the BASIC ROMs. Using this approach, anything your version of BASIC can do (*including graphics*), your C program can also do. This feature could also be used to work with floating point numbers.

CoCo-C features some special functions that make it easier to work in a CoCo environment. These include commands for switching a CoCo 3 between the CoCo 3 and CoCo 1/2 modes, testing whether the program is running on a CoCo 3 or an earlier Color Computer, setting up for the CoCo 3's RGB and composite-video modes, and setting the computer for the high-speed mode.

Some special functions that are normally used by the `CSTART` library routine are documented and can be used to set up buffers in memory and to change the error-checking characteristics. *CoCo-C* also provides a mechanism that allows you to insert or even `include` assembler routines in your C program, giving you even more intimate control of your system.

As I implied before, *CoCo-C* is a well-documented implementation of C for Disk BASIC. No, it isn't C++, and it doesn't have a huge library of extras. But *CoCo-C* gives you most of the standard C functions. Besides, with `bascmd`, the BASIC ROMs themselves become a fairly extensive library of routines.

Finally, all of the expected expressions and operators are supported, including the shift operators. Most of the normal program control statements, such as `if/then/else`, `while/do` and `for` work as they should. Some of the usual conversion functions do not work as expected or are not supported because of the limited types of data that can be handled.

The Documentation

I was impressed with the quality of the manual supplied with *CoCo-C*. Since writing users manuals is my job, I have some idea what's involved in producing a good, understandable manual; *CoCo-C*'s manual appears to meet all of the criteria. You are taken through a logical progression from entering the source code for a program to compiling, assembling, linking and finally executing that program. Each function is presented using the typical C syntax statement, then a complete description explains how the function is used, what type of data it requires and what type of data it returns. An example of the function when used in a C language program is given to help clarify its use. In addition, program examples in the back of the book provide step-by-step and line-by-line explanations showing you how to enter a program and what you need to do to compile it.

Also included in the manual are complete technical specifications to help you understand how you can connect programs written in *CoCo-C* to other types of programs as well as how *CoCo-C* puts your program together. This section explains how the compiler works with C-language source code to construct the finished executable program.

Although *CoCo-C*'s manual is not specifically designed to teach C-language programming, it gives the average CoCo user more than enough information to really get going. Still, if you are not familiar with C and need more information, the manual provides a list of reference books that should help.

The Real World

One of the most important things about writing programs in C is the portability of a program from one computer to another. Most C-language text-based programs should be fairly easy to convert for *CoCo-C*. The primary concern is that Disk BASIC's disk structure is somewhat limited in that it does nor provide a hierarchical directory structure and leaves only one side of the disk available for storage. Allowing for this when converting programs written for other systems requires considerable changes. Since C was originally developed for Unix, an operating system that allows complex disk structure and lots of space, this could be an important consideration when trying to convert another C program to run under Disk BASIC. On the other hand, working with files is made much easier through the application of some unique input/output functions that work well with the CoCo.

As long as these and the data-typing limitations are kept in mind, there should not be any trouble writing and converting useful C language programs for your CoCo. And remember, C is a powerful language that encourages the programmer to expand it's capabilities. By writing your own functions to work with other data types, or perhaps using the Disk BASIC interface, there should be no limit to the possibilities *CoCo-C* offers for writing complex and useful software. The structured approach to programming for which C is so famous gives the programmer the tools he needs to develop programs for the CoCo that aren't limited by the BASIC ROM routines and are much easier to work with than programs written in assembly language. *(Infinitum Technology, P.O. Box 356, Saddle River, N.J. 07458, (914) 356-7688; introductory offer: $59.95 plus $4 S/H.)*

— Bill Budenholzer

Net Worth *from Page 1*

To stop entering data, telling *Financial Statement* to finish printing your totals, select Option 8 from the main menu.

Financial Statement is designed for use with the DMP-130 printer. The only control code used, however, is CHR$(27);CHR$(20), which appears in Line 400. This code tells the DMP-130 (and most other Radio shack printers in the Tandy mode) to switch to condensed print. Change this code as necessary for your printer.

Charles Kiedaisch is a retired tool-and-die designer who still does some independent work and uses his CoCo 3 to create master drawings. He enjoys building useful programs for the Color Computer.

16K ECB

The Listing: STATEMNT

```
1 'FINANCIAL STATEMENT
2 'BY CHARLES KIEDAISCH
3 'COPYRIGHT (C) 1993
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 CLS:'FINANCIAL STATEMENT-SPRE
AD SHEET
20 R=0
30 PRINT:PRINT:PRINT"MAKE SURE P
RINTER IS ON LINE"
40 PRINT:PRINT"PRESS <ENTER> WHE
N READY"
50 INPUT E$
60 GOTO 300
70 CLS:PRINT:PRINT"INPUT DATA"
80 PRINT
90 PRINT"SELECT INPUT"
100 PRINT"  1) WEEKLY INCOME"
110 PRINT"  2) MONTHLY INCOME"
120 PRINT"  3) YEARLY INCOME"
130 PRINT"  4) WEEKLY EXPENSE"
140 PRINT"  5) MONTHLY EXPENSE"
150 PRINT"  6) YEARLY EXPENSE"
160 PRINT"  7) ASSETS"
170 PRINT"  8) END INPUT"
180 AO$=INKEY$:IF AO$=""THEN180E
LSE190
190 AO=VAL(AO$):ON AO GOTO 200,2
10,220,240,250,260,280,510
200 GOSUB640:PRINT"WEEKLY INCOME
 ";:INPUT P:M=P*52/12:Y=P*52:GOT
O 230
210 GOSUB640:PRINT"MONTHLY INCOM
E ";:INPUT M:Y=M*12:P=Y/52:GOTO
230
220 GOSUB640:PRINT"YEARLY INCOME
 ";:INPUT Y:M=Y/12:P=Y/52:GOTO 2
30
230 Q$="INCOME":GOTO 480
240 GOSUB 640:PRINT"WEEKLY EXPEN
SE ";:INPUT E:F=E*52/12:G=E*52:G
OTO270
250 GOSUB 640:PRINT"MONTHLY EXPE
NSE ";:INPUT F:G=F*12:E=G/52:GOT
O 270
260 GOSUB 640:PRINT"YEARLY EXPEN
SE ";:INPUT G:F=G/12:E=G/52:GOTO
 270
270 Q$="EXPENSE":GOTO 480
280 GOSUB 640:PRINT"FACE VALUE "
;:INPUT A
290 Q$="ASSET":GOTO 480
300 REM OUTPUT TO PRINTER
310 PRINT:LINEINPUT"DATE? ";DATE
$
320 CLS:PRINT@233,"**PRINTING**"
330 TT=0
340 MM=0
350 YY=0
360 EE=0
370 FF=0
380 GG=0
390 AA=0
400 PRINT#-2,CHR$(27);CHR$(20)
410 PRINT#-2,TAB(56)"**FINANCIAL
 STATEMENT**"
420 PRINT#-2,TAB(3)"DATE:    ";DA
TE$
430 PRINT#-2," ":PRINT#-2,STRING
$(133,"*")
440 PRINT#-2,TAB(55)"WEEKLY";TAB
(66)"MONTHLY";TAB(79)"YEARLY";TA
B(91)"WEEKLY";TAB(102)"MONTHLY";
TAB(115)"YEARLY";TAB(129)"FACE"
450 PRINT#-2,"CATAGORY";TAB(20)"
DESCRIPTION";TAB(55)"INCOME";TAB
(67)"INCOME";TAB(79)"INCOME";TAB
(90)"EXPENSE";TAB(102)"EXPENSE";
TAB(114)"EXPENSE";TAB(128)"VALUE
"
460 PRINT#-2,STRING$(133,"-")
470 GOTO 70
480 CLS:PRINT@233,"**PRINTING**"
:PRINT#-2,USING"## %        % %

%    $$###.##    $$####.##  $$###
##.##    $$###.##    $$####.##  $
$#####.##  $$#####.##";R;Q$;A$;P
;M;Y;E;F;G;A
490 PP=PP+P:MM=MM+M:YY=YY+Y:EE=E
E+E:FF=FF+F:GG=GG+G:AA=AA+A
500 GOTO 70
510 CLS:PRINT@233,"**PRINTING**"
:PRINT#-2,TAB(51)"------  -
------  ------  ------
--"
520 PRINT#-2,TAB(34)"TOTAL";:PRI
NT#-2,USING"          $$######.
##  $$######.## $$######.## $$###
###.## $$######.## $$######.## $
$######.##";PP;MM;YY;EE;FF;GG;AA
530 PRINT#-2
540 PRINT#-2,TAB(27)"TOTAL INCOM
E";:PRINT#-2,USING"          $
$#####.##  $$######.## $$######.#
#";PP;MM;YY
550 PRINT#-2,TAB(27)"TOTAL EXPEN
SE";:PRINT#-2,USING"          $
$#####.##  $$######.## $$######.#
#";EE;FF;GG
560 TT=PP-EE:ZZ=MM-FF:XX=YY-GG:W
W=XX+AA
570 PRINT#-2,TAB(49)"  ------
-  ------  ------"
580 PRINT#-2,TAB(32)"BALANCE";:P
RINT#-2,USING"          $$####
#.##  $$######.## $$######.##";TT
;ZZ;XX
590 PRINT#-2,TAB(32)"ASSETS";:PR
INT#-2,USING"
          $$######.##";AA
600 PRINT#-2,TAB(75)"------"
610 PRINT#-2,TAB(13)"ESTIMATED Y
EAR END NET WORTH";:PRINT#-2,USI
NG"
    $$######.##";WW
620 PRINT#-2,STRING$(133,"*"):PR
INT#-2," "
630 CLS:PRINT@230,"PROGRAM ENDED
":END
640 R=R+1:P=0:M=0:Y=0:E=0:F=0:G=
0:A=0
650 PRINT"DESCRIPTION ";:LINEINP
UT A$:RETURN
```

Feature Program

Learn to be *Quik* on the Keyboard

by George and Ellen Aftamonow

Quik, a cute game for the CoCo 3, is not only a lot of fun, it helps you learn a little about the DRAW command used for Color Computer graphics. To get started, enter the complete listing and save it to tape or disk (before running it). Then enter RUN.

Once the game is started, you'll see eight arrows on the screen — one arrow pointing in each of the eight directions for which DRAW subcommands support direct movement (see Figure 1). One of these arrows should be flashing.

The object of *Quik* is to quickly press the key that corresponds to the direction of the flashing arrow. As soon as you do, another arrow starts flashing. The computer chooses the arrows randomly, so if you press the correct key and the arrow continues flashing, press that key again. This process continues for a specific length of time (which you select after running the program but before starting the game). When the time is up, the number of correct responses you made appears on the screen and you are given the option to play again.

Quik is a lot of fun to play, and the game is educational in a somewhat unique way. We hope you enjoy it, too.

George and Ellen Aftamonow, two self-taught programmers, believe computer users need another number cruncher as badly as a pig needs a wallet. So they like to sit down and enjoy the challenge of writing entertainment software. They can be contacted at 46 Howe Street, Milford, CT 06460, (203) 878-3602. Please include an SASE when requesting a reply.

CoCo 3

The Listing: QUIK

```
1 'QUIK
2 'BY GEORGE & ELLEN AFTAMONOW
3 'COPYRIGHT (C) 1993
4 'FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 PALETTE0,0:PALETTE5,63:HSCREEN2:HCLS0:HCOLOR5,0
15 HPRINT(10,10),"(C)MP OR (R)GB?"
20 I$=INKEY$:IFI$=""THEN20
30 IFI$="C"THEN40ELSEIFI$="R"THEN50ELSE20
40 PALETTECMP:PALETTE0,0:PALETTE1,17:PALETTE2,39:PALETTE3,57:PALETTE4,53:PALETTE5,63:PALETTE6,41:PALETTE7,57:PALETTE8,44:PALETTE12,39:PALETTE14,51:GOTO70
50 PALETTERGB:PALETTE0,0:PALETTE1,17:PALETTE2,38:PALETTE3,29:PALETTE4,55:PALETTE5,63:PALETTE6,45:PALETTE7,57:PALETTE8,44:PALETTE12,38:PALETTE13,42:PALETTE14,15'
 0=BLK 1=GRN 2=RED 3=BLU 4=YLW 5=WHT 6=PUR
70 DATA6,6,1,17,"H",160,6,2,38,"U",315,6,3,29,"E",315,90,4,55,"R",315,174,5,63,"F",160,174,6,45,"D",6,174,7,57,"G",6,90,8,44,"L"
74 '*X & Y PAINT CO-ORDINATES, PALETTE NO. & COLOR, DIRECTION
75 FORZ=1TO8:READX(Z),Y(Z),P(Z),C(Z),C$(Z):NEXT
79 '*MAIN SCREEN
80 HCLS0:HSCREEN2:HCOLOR12
90 HLINE(60,50)-(270,130),PSET,B
100 HDRAW"BM2,4R34G10F4G12H6G10U30":HDRAW"BM160,4F22L14D6L16U6L14E22":HDRAW"BM318,4D32H10G6H12E6H10R32":HDRAW"BM318,90G22U14L6U16R6U14F22":HDRAW"BM2,90E22D14R6D16L6D14H22"
105 HDRAW"BM2,178U32F10E6F12G6F10L32":HDRAW"BM160,178H22R16U6R12D6R16G22":HDRAW"BM318,178L32E10H6E12F6E10D32"
106 FORZ=1TO8:HPAINT(X(Z),Y(Z)),P(Z),12:NEXT
110 HDRAW"BM100,80L6NU16H2U12E2R6F2D12G2NF2BR14 L6NU16H2U14R2BR8D14G2BR8 L2U16R2D16BR14 L6NU16H2U12E2R6F2BD12G2BR8 L2U16R2D8NE8NF8D8"
115 HDRAW"BM190,80L6NU16L2NLU16NLR8F2D12G2BR8 NU16L2U16R8F2D4G2L6F8BR6 NU16L2U14E2R6F2D6NL8D8BR6 H2U14R2D16E6NU4F6U16"
120 HPRINT(20,12),"BY":HPRINT(9,14),"GEORGE & ELLEN AFTAMONOW"
130 FORQ=1TO3:PLAY"T255O5BADBADBAD":NEXTQ
140 HCOLOR0:HLINE(62,52)-(268,128),PSET,BF
150 HCOLOR5:HPRINT(10,10),"NEED INSTRUCTIONS Y/N?"
160 I$=INKEY$:IFI$=""THEN160
200 HCOLOR0:HPRINT(10,10),"NEED INSTRUCTIONS Y/N?":IFI$="N" THEN214
202 HCOLOR3:HPRINT(9,8),"THERE ARE  EIGHT ARROWS.":HPRINT(9,9),"EACH IS  POINTING IN THE":HPRINT(9,10),"DIRECTION THAT COINCIDES ":HPRINT(9,11),"WITH THE COMPUTER'S DRAW":HPRINT(9,12),"COMMAND.":HPRINT(13,14),"PRESS ANY KEY"
203 I$=INKEY$:IFI$="" THEN203
204 HCOLOR0:HLINE(62,52)-(268,128),PSET,BF
205 HDRAW"BM164,90C3NL12NH12NU12NE12NR12NF12ND12G12":HPRINT(15,11),"L":HPRINT(25,11),"R":HPRINT(20,8),"U":HPRINT(20,14),"D"
206 HCOLOR2:HPRINT(17,9),"H":HPRINT(23,9),"E":HPRINT(23,13),"F":HPRINT(17,13),"G"
207 I$=INKEY$:IFI$="" THEN207
208 HCOLOR0:HLINE(110,60)-(210,120),PSET,BF
210 HCOLOR4:HPRINT(9,8),"AN ARROW WILL RANDOMLY":HPRINT(9,9),"FLICKER. THE OBJECT IS":HPRINT(9,10),"TO  PRESS THE KEY THAT":HPRINT(9,11),"REPRESENTS THE ARROW'S ":HPRINT(9,12),"DIRECTION.":HPRINT(13,14),"PRESS ANY KEY"
212 I$=INKEY$:IFI$="" THEN212
213 HCOLOR0:HLINE(62,52)-(266,126),PSET,BF
214 HCOLOR4:HPRINT(15,8),"CHOOSE  1-3":HPRINT(14,10),"1) SHORT GAME":HPRINT(14,11),"2) MED.  GAME":HPRINT(14,12),"3) LONG  GAME"
220 I$=INKEY$:IFI$=""THEN220
221 '*G=LENGTH OF GAME
222 IFI$="1"THENG=600ELSE IFI$="2"THENG=1200ELSEIFI$="3"THENG=1800ELSE222
225 P$="T250O2L200GEA":PLAYP$
230 HCOLOR0:HLINE(106,62)-(220,104),PSET,BF
240 T=0:SC=0
249 '*CHOOSE RND PALETTE
250 C=RND(8)
299 '*FLICKER PALETTE:T=TIME
300 T=T+1:I$=INKEY$:IFT=G THEN340ELSEIFI$="" THENPALETTEC,0:PALETTEC,C(C):GOTO300
310 IFI$<>C$(C) THEN300
320 SC=SC+1:GOTO250
340 PLAY"T200L200EAGAEAGA":HCOLOR2:HPRINT(15,8),"YOU SCORED":HPRINT(11,10),STR$(SC)+" IN"+STR$(G/20)+" SECONDS."
350 HPRINT(14,14),"PLAY AGAIN?"
360 I$=INKEY$:IFI$="" THEN360
370 IFI$="Y" THEN213 ELSEIFI$="N" THENCLS:RGB:END:ELSE360
```

Feature Program

One Address or Many?

by Charles Kiedaisch

Whether or not they write letters to friends, most people spend at least a little time each month addressing envelopes. Not everyone, however, gets high marks for penmanship. And when greeting cards are in order — especially during the holidays — writer's cramp becomes a problem (usually for those who *don't* write often). The two programs presented here can be a real help during these times.

The program shown in Listing 1, *Address*, prints your return address and the recipient's address on just about any envelope. Before you run the program, make sure you enter your name and address in lines 160 through 180. When *Address* is executed, you are first asked to enter the recipient's name and address. The program then prints your return address and asks what size envelope you are using. Press 1, 2, 3 or 4 accordingly, and the program proceeds to print the remainder of the envelope.

Address is designed to accommodate four different envelope sizes: standard, long, odd-sized and small. Actually what happens is that the printhead is moved (using PRINT # -2, TAB(T)) to a different position based on the selected envelope size. The tab values are set up in lines 280, 300, 320 and 330. Feel free to change these as you see fit.

Address Two (Listing 2) works much like *Address* except that it allows you to store data for the addresses to which you frequently send mail. In addition, you can elect to print all stored addresses or only a specific address, on the printer or to the screen only. You can also print a complete directory of all stored addresses.

Before running the program, make sure the names and addresses you want are stored in the data lines at the end of the listing. Use the same format indicated when you enter your data, and make sure the last DATA statement starts with the word END. Also make sure you enter your return address in lines 620 through 640. The tab values for different-sized envelopes are in lines 980, 1000, 1020 and 1030 should you want to change them.

Some users may wonder why I wrote *Address Two* in such a fashion that it uses DATA statements instead of storing address information on disk. In the first place, using DATA statements makes the program easier to use with tape-based CoCo systems. Secondly, it allows you to create several different versions of the program, using specific groups of addresses in each, much like separate mailing lists. This is handy when you send cards or letters to different groups of people on different occasions. And you can put the names and addresses of all your creditors (those who don't provide pre-addressed envelopes, anyway) in one program listing.

Both *Address* and *Address Two* are designed to work with the Tandy DMP-133 in the Tandy mode. The control codes used and the lines in which they appear are shown in Figure 1.

In an effort to help the postal service by printing clearly, we only help ourselves, increasing the chance that our mail will get where it's supposed to go. I believe you'll find *Address* and *Address Two* to be useful additions to your library.

Charles Kiedaisch is a retired tool-and-die designer who still does some independent work and uses his CoCo 3 to create master drawings. He enjoys building useful programs for the Color Computer.

Control Code	Function	*Address* line number(s)	*Address Two* line number(s)
CHR$(27);CHR$(14)	start elongation	60, 370	440, 1070
CHR$(27);CHR$(15)	end elongation	100, 410	480, 1110
CHR$(27);CHR$(20)	condensed	160	620

Figure 1: *Address* and *Address Two* Control Codes

16K ECB

Listing 1: ADDRESS

```
1 'ENVELOPE ADDRESS PRINTER
2 'BY CHARLES KIEDAISCH
3 'COPYRIGHT (C) 1993
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE"
10 CLS:REM*ENVELOPE ADDRESS*
20 LINEINPUT"NAME ";T$
30 LINEINPUT"NUMBER AND STREET "
;G$
40 LINEINPUT"CITY AND ZIP CODE "
;M$
50 GOTO 160
60 CLS:PRINT#-2,CHR$(27);CHR$(14
)
70 PRINT T$:PRINT#-2:PRINT#-2:PR
INT#-2:PRINT#-2,TAB(T);T$
80 PRINT G$:PRINT#-2,TAB(T);G$
90 PRINT M$:PRINT#-2,TAB(T);M$
100 PRINT#-2,CHR$(27);CHR$(15)
110 PRINT:PRINT"ANOTHER ADDRESS
(Y/N)"
120 N$=INKEY$
130 IF N$=""THEN 120
140 IF N$="Y"THEN 10
150 IF N$="N"THEN 430
160 PRINT#-2,CHR$(27);CHR$(20);"
JOE SOMEBODY"
170 PRINT#-2,"11306 ANY ST"
180 PRINT#-2,"SOME PLACE IL 6044
8"
190 CLS
200 PRINT"          1) STANDARD E
NVELOPE"
210 PRINT"          2) LONG ENVEL
OPE"
220 PRINT"          3) ODD SIZE F
NVELOPES"
230 PRINT"          4) SMALL ENVE
LOPE"
240 PRINT
250 PRINT"     SELECT (1,2,3 OR
4)"
260 AN$=INKEY$:IF AN$=""THEN 260
270 ON VAL(AN$)GOTO 280,300,320,
330
280 T=21
290 GOTO 60
300 T=40
310 GOTO 60
320 T=23:GOTO 340
330 T=15:GOTO 340
340 CLS:PRINT"ADJUST ENVELOPE IF
 NECESSARY"
350 PRINT"PRESS (Y) TO PRINT"
360 PR$=INKEY$:IF PR$=""THEN 360
:IF PR$="Y"THEN 370
370 PRINT#-2,CHR$(27);CHR$(14)
380 PRINT T$:PRINT#-2,TAB(T);T$
390 PRINT G$:PRINT#-2,TAB(T);G$
400 PRINT M$:PRINT#-2,TAB(T);M$
410 PRINT#-2,CHR$(27);CHR$(15)
420 GOTO 110
430 CLS
440 PRINT"PROGRAM ENDED"
450 END
```

Listing 2: ADDRESS2

```
1 'FILER/ADDRESS PRINTER
2 'BY CHARLES KIEDAISCH
3 'COPYRIGHT (C) 1993
4 'BY FALSOFT, INC.
5 'RAINBOW MAGAZINE
10 CLS:REM*NAME AND ADDRESS LIST
/ENVELOPE ADDRESSING*
20 S=1
30 M=1000
40 PRINT"          **************
***"
50 PRINT"          1) LINE PRINTER
"
60 PRINT"          2) SCREEN DISPL
AY ONLY"
70 PRINT"          3) PRINT NAME D
IRECTORY"
80 PRINT"          4) END PROGRAM"
90 PRINT"          **************
***"
100 PRINT"SELECT(1,2,3 OR 4)"
110 AN$=INKEY$:IF AN$=""THEN 110
120 ON VAL(AN$)GOTO 130,140,660,
560
130 RESTORE:L$="G":PRINT:GOTO 15
0
140 RESTORE:L$="P":PRINT
150 PRINT"     DO YOU WANT ALL N
AMES (Y/N)"
160 X$=INKEY$
170 IF X$=""THEN 160
180 IF X$="Y"THEN 210
190 RESTORE:PRINT:PRINT"ENTER TH
E NAME TO SEARCH FOR"
200 INPUT S$
210 REM**PROCESSING AREA**
220 PRINT
230 PRINT
240 FOR I=1 TO M
250 READ B$
260 IF B$="END"THEN 10
270 READ T$,G$,M$
280 S=0
290 IF B$<>S$THEN 310
300 S=1
310 IF X$="Y"THEN 330
320 IF S<>1 THEN 550
330 CLS:PRINT T$
340 PRINT G$
350 PRINT M$
360 IF L$="P"THEN 490
370 PRINT:PRINT"SHALL I PRINT (Y
/N)"
380 P$=INKEY$
390 IF P$=""THEN380
400 IF P$="Y"THEN 420
410 IF P$="N"THEN 490
420 GOSUB 620
430 GOSUB 890
440 PRINT#-2,CHR$(27);CHR$(14)
450 PRINT#-2:PRINT#-2:PRINT#-2:P
RINT#-2,TAB(T);T$
460 PRINT#-2,TAB(T);G$
470 PRINT#-2,TAB(T);M$
480 PRINT#-2,CHR$(27);CHR$(15)
490 PRINT:PRINT"ANOTHER ADDRESS
(Y/N)"
500 N$=INKEY$
510 IF N$=""THEN 500
520 IF N$="Y"THEN 540
530 IF N$="N"THEN 10
540 CLS:IF X$="N"THEN 190
550 NEXT I
560 REM**** TERM PT****
570 PRINT
580 PRINT"INPUT <RUN> TO RESTART
"
590 PRINT
600 END
610 L$="P":GOTO 140
620 PRINT#-2,CHR$(27);CHR$(20);"
JOE SOMEBODY"
630 PRINT#-2,"11306 ANY ST"
640 PRINT#-2,"SOME PLACE IL 6000
0"
650 RETURN
660 REM**PRINT NAME DIRECTORY**
670 CLS:PRINT"PRESS <BREAK> TO S
TOP"
680 PRINT"INPUT <CONT> TO CONTIN
UE"
690 PRINT
700 RESTORE:PRINT"DO YOU WANT A
PRINTOUT (Y/N)"
710 H$=INKEY$
720 IF H$=""THEN 710
730 IF H$="Y"THEN 740
740 FOR I=1 TO M
750 READ B$
760 IF B$="END"THEN 10
770 READ T$,G$,M$
780 S=0
790 IF B$<>S$THEN810
800 S=1
810 PRINT
820 PRINT B$;") ";T$
830 IF H$="N"THEN 860
840 PRINT#-2,B$;") ";T$
850 NEXT I
860 FOR D=1 TO 500
870 NEXT D
880 GOTO 850
890 CLS
900 PRINT"          1) STANDARD E
NVELOPE"
910 PRINT"          2) LONG ENVEL
OPE"
920 PRINT"          3) ODD SIZE E
NVELOPES"
930 PRINT"          4) SMALL ENVE
LOPE"
940 PRINT
950 PRINT"     SELECT (1,2,3 OR
4)"
960 AN$=INKEY$:IF AN$=""THEN 960
970 ON VAL(AN$)GOTO 980,1000,102
0,1030
980 T=21
990 GOTO 1010
1000 T=40
1010 RETURN
1020 T=23:GOTO 1040
1030 T=15:GOTO 1040
1040 CLS:PRINT"ADJUST ENVELOPE I
F NECESSARY"
1050 PRINT"PRESS (Y) TO PRINT"
1060 PR$=INKEY$:IF PR$=""THEN 10
60:IF PR$="Y"THEN 1070
1070 PRINT#-2,CHR$(27);CHR$(14)
1080 PRINT T$:PRINT#-2,TAB(T);T$
1090 PRINT G$:PRINT#-2,TAB(T);G$
1100 PRINT M$:PRINT#-2,TAB(T);M$
1110 PRINT#-2,CHR$(27);CHR$(15)
1120 GOTO 490
1130 REM**DATA FOLLOWS*
1140 DATA 1,LAST NAME,00000 NOPL
ACE,HOME 00000
1150 DATA 2,MR BILL SMITH,2345 N
ORTH ST,TRENTON NJ 87654
1160 DATA 3,MRS CATHY DOE,3456 W
EST AVE,MIAMI FL 76543
1170 DATA 4,DENNY DIMWIT,4567 EA
ST ROAD,CHICAGO IL 65432
1180 DATA 5,JOE BLOW,567 AVENUE
B,SALEM OR 54321
1190 DATA 6,MARY SMITH,678 AVENU
E C,PORTLAND OR 54322
1200 DATA 7,JOHN W,890 AVENUE D,
SALEM OR 54321
1210 DATA 8,BOB BOOB,980 AVENUE
F,WALLA WA 76598
1220 DATA 9,HARRY HOOD,934 SOUTH
 ROAD,GARY IN 68907
1230 DATA 10,MR AL JONES,1234 SO
UTH ST,NEW YORK NY 98765
1240 DATA END
```

Database *from Page 1*

doesn't matter whether you enter lower- or uppercase characters — all keywords are automatically stored in uppercase. The program allows up to a total of 15 keywords by which you can categorize your programs and files. If, during the course of operation, you delete all database records associated with a specific keyword, that keyword is automatically removed from the list.

If the filename you enter for a record already exists in the database, *DBOS9* will not store that record. The program searches the database before any records are written to ensure that there are no duplicates.

After you select an existing keyword (or create a new one) for the new record, you are asked to enter the OS-9 filename for the file or program for which you are recording information. *DBOS9* accepts filenames up to 24 characters in length. Again, all filenames are automatically switched to all uppercase characters.

The next prompt allows you to enter up to three text lines of up to 49 characters each. These lines can be used to describe the program or file, or to remind you of preliminary operating instructions for that program.

Finally you are asked for the disk identifier. This is a special place for whatever identifying names you use for the disks in your library. After this, you are given the option of storing the record in the database or returning to the main menu without storing the information.

It is important to note that if the filename you enter for a record already exists in the database, *DBOS9* will not store that record. The program searches the database before any records are written to ensure that there are no duplicates.

Searching for Records

To search *DBOS9*'s `os9` database file, press 1 at the main menu. A submenu appears with the following options: 1) Keyword Search, 2) Name Search, 3) First Letter Search, 4) List Names and 5) Main Menu. The function performed by each of these entries is fairly obvious when you are running the program, so we'll take only a brief look at them here:

Option 1 — To search for records based on the selected keywords, press 1. After the keyword box is displayed in the upper-left corner of the screen, select the numnber for the keyword on which you want to search. *DBOS9* finds the first record categorized under the selected keyword. Subsequent records are displayed as explained below.

Option 2 — To call up the record for a specific file or program, press 2 at the Search menu. When prompted, enter the exact name of the program or file in question. Since all filenames are converted to uppercase, it doesn't matter whether you

Database *continued on Page 12.*

OS-9 Level II

Listing 1: `os9top.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

struct db9 {                            /*declare global structures*/
   char key[15];
   char name[25];
   char descr[3][60];
    char disk[15];
   } os9;

struct tempo{
   char key[15];
   char name[25];
   char descr[3][60];
    char disk[15];
   }temp;

pflinit();
FILE *fp,*kp;                           /*declare global variables*/
char database[]="/DD/BASE9/os9";
int count,recurse;
long t;
char keywords[15][15];

main()                                  /*start of program*/
   {

   char ch=0,ch2;
   int x,init=0;

#asm
info "copywrite P.Scherer 1992"
#endasm

   setbuf(stdin,0);                     /*set buffers to 0*/
   setbuf(stdout,0);
                                        /*create database if non-existent*/
   if((fp=fopen(database,"r+"))==NULL){
         if((fp=fopen(database,"w+"))==NULL)
             exit(1);
         fwrite(&init,sizeof init,1,fp);

         strcpy(os9.name,"??");         /*initialize database file*/
         fwrite(&os9,sizeof (struct db9),1,fp);
           }
                                        /*create keyword.dat*/
   if((kp=fopen("/dd/base9/keyword.dat","r+"))==NULL){
        if((kp=fopen("/dd/base9/keyword.dat","w+"))==NULL)
            exit(1);
        for(x=0;x<15;x++){              /*initialize keyword.dat*/
            strcpy(keywords[x],"??");
            fwrite(keywords[x],15,1,kp);}
         fclose(kp);}
    OWSet(1,1,0,0,80,24,0,2);

    do{                                 /*create main menu*/

      if(ch!=51){
         OWSet(1,1,12,6,60,13,0,4);
         OWSet(1,1,10,7,60,13,0,1);
         OWSet(1,1,11,8,58,11,3,2);
         CurXY(1,25,1);
         puts("MAIN MENU");
         puts("\n  1) Search");
         puts("  2) Enter New Record");
        puts("  3) Backup Database");
         puts("  4) Exit");}

   do{                                  /*force a selection from 1 to 4*/
         CurXY(1,0,9);
         DelLine(1);
         CurXY(1,14,9);
         printf("SELECT A NUMBER: ");
         ch=getchar();
         } while(ch<49||ch>52);

   switch(ch) {                         /*call appropriate function*/
         case '1':OWEnd(1);
                  OWEnd(1);
                  OWEnd(1);
                  os9Search();
                   break;
         case '2':OWEnd(1);
                  OWEnd(1);
                  OWEnd(1);
                  os9Enter();
                   break;
        case '3':CurXY(1,0,9);
                  ErLine(1);
                  fclose(fp);
              printf(" Put Disk In /d0 And Press <B> Or Any Key To Quit: ");
                  ch2=getchar();
                  if(ch2=='B'||ch2=='b'){
                      CurXY(1,0,9);
                      ErLine(1);
                      system("del /d0/os9.bak");
                      system("copy /dd/base9/os9 /d0/os9.bak");
                      system("del /d0/keyword.bak");
                      system("copy /dd/base9/keyword.dat /d0/keyword.bak");}
                  fp=fopen(database,"r+");
                  break;
         default: OWEnd(1);
                   OWEnd(1);
                   OWEnd(1);
                  OWEnd(1);
                  }
         } while(ch!=52);
         fclose(fp);
         }
```

Listing 2: os9search.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

extern struct db9{                  /*reference global structures*/
       char key[15];                /*and variables*/
       char name[25];
       char descr[3][60];
       char disk[15];
       }os9;
extern struct tempo{
       char key[15];
       char name[25];
       char descr[3][60];
       char disk[15];
       }temp;

extern int count,recurse;
extern FILE *fp;

 os9Search(){

       register int x;
       char ch,ch2;
       int y,j,z;
       char str[15];

        OWSet(1,1,10,4,60,13,0,4);  /*create main search menu*/
        OWSet(1,1,14,5,52,11,0,1);
        OWSet(1,1,18,6,44,9,0,2);
        CurXY(1,13,1);
        printf("OS9 SEARCH MENU");
        printf("\n 1) Keyword Search");
        printf("\n 2) Name Search");
        printf("\n 3) First Letter Search");
        printf("\n 4) List Names");
        printf("\n 5) Main Menu\n");

        do{
            do{                     /*force a selection from 1 to 5*/
                CurXY(1,1,7);
                DelLine(1);
                CurXY(1,10,7);
                printf("SELECT A NUMBER: ");
                ch=getchar();
            }while(ch<48||ch>54);
            recurse=0;              /*recurse variable used to keep track*/
                                    /*of mutual recursion*/

        switch(ch) {       /*selection #1 calls keyword function directly*/
            case'1':j=3;       /*variable <j> controls keyword() function*/
                    keyword(j);             /*response in all selections*/
                    break;
            case'2':OWSet(1,1,20,10,40,5,0,1);
                    printf("\n Enter Name To Search or <C/R> To Quit\n");
                    printf("\n : ");
                    y=readln(0,str,14);       /*read name entry*/
                    str[y-1]=0;
                    if(*str==0){              /*check for a <C/R>*/
                        OWEnd(1);
                        break;}
                    j=4;
                    for(x=0;str[x];x++)           /*convert to upper case*/
                        str[x]=toupper(str[x]);
                    rewind(fp);                    /*reset file pointer*/
                    fread(&count,sizeof count,1,fp); /*read number of files*/
                    y=0;
                    for(x=0;x<count;x++){         /*search for name match*/
                        z=x;
                        fread(&os9,sizeof(struct db9),1,fp);
                        if(!strcmp(os9.name,str)){    /*if a match is found*/
                          do{                         /*set up display window*/
                            if(x<z)
                                x++;
                            if(!y){
                                OWEnd(1);
                                OWSet(1,1,2,2,76,16,0,3);
                                OWSet(1,1,3,3,74,14,0,4);
                                OWSet(1,1,4,4,72,12,0,2);
                                y++;}
                                x=display(x,j);   /*call display() and pass*/
                            }while(z>x);}}        /*variable x & j. X is*/
                                                  /*used to end search after*/
                                                  /*file is displayed*/
                    if(y){
                        OWEnd(1);
                        OWEnd(1);
                        OWEnd(1);}
                    if(!y){        /*if y is 0 then there is no match*/
                        Clear(1);
                        CurXY(1,0,1);
                        printf("There Are No %s Records",str);
                        printf("\nPress Any Key: ");
                        ch=getchar();
                        OWEnd(1);}
                        break;
        case'3':OWSet(1,1,20,10,50,3,0,1);
                printf("\n Enter Letter To Search or <C/R> To Quit: ");
                ch=getchar();
                if(ch!='\n'){          /*test for a <C/R>*/
                    j=5;
                    ch=toupper(ch);    /*convert to upper case*/
                    y=0;
                    rewind(fp);     /*reset file pointer and read count*/
                    fread(&count,sizeof count,1,fp);
                    for(x=0;x<count;x++){      /*start of letter search*/
                        z=x;
                        fread(&os9,sizeof(struct db9),1,fp);
                        if(os9.name[0]==ch){
```

Program listing continued

```
/*if <y> is 0 then set up window for display().
After one call to display(), y will be incremented
and the window call will not repeat.*/

                        do{
                            if(x<z)
                                x++;
                            if(!y){
                                OWEnd(1);
                                OWSet(1,1,2,2,76,16,0,3);
                                OWSet(1,1,3,3,74,14,0,4);
                                OWSet(1,1,4,4,72,12,0,2);
                                y++;}
                                x=display(x,j);   /*pass variable x & j*/
                            }while(z>x);}}         /*to display(). X is*/
                                                 /*used to end search if*/
                                                   /*required*/

                        if(y){                  /*check to see if a match*/
                            if(x==count){   /*was found*/
                              Clear(1);
                              CurXY(1,10,6);
                              printf("There Are No More %c Entries.",ch);
                              printf(" Press Any Key: ");
                              ch2=getchar();}
                              OWEnd(1);
                              OWEnd(1);}
                        else{                   /*send message if no match*/
                              Clear(1);
                    printf("\n There Are No %c Entries-Press Any Key: ",ch);
                              ch2=getchar();}}
                              OWEnd(1);
                    break;
            case'4':OWSet(1,1,10,2,32,18,0,1);   /*set up file list window*/
                    OWSet(1,1,11,3,30,16,0,2);
                    rewind(fp);      /*reset file pointer and read count*/
                    fread(&count,sizeof count,1,fp);
                    y=0;
                    do{
                        for(x=0;x<6&&y<count;x++){   /*6 file display loop*/
                            fread(&os9,sizeof(struct db9),1,fp);
                            if(strcmp(os9.name,"??"))
                                printf("\n %s\n",os9.name);
                            else x--;
                                y++;}
                            printf("\n Press Any Key\n");
                            printf(" Or <C/R> To Quit: ");
                            ch2=getchar();
                            if(ch2=='\n')  /*end search if <C/R> is entered*/
                                y=count;
                            Clear(1);
                        }while(y<count);
                    OWEnd(1);
                    OWEnd(1);
                    break;
            default:OWEnd(1);   /*close search menu and return to main*/
                    OWEnd(1);
                    OWEnd(1);
                    return;}
        }while(ch!='5');
        }
```

Database *from Page 10*

use lower- or uppercase characters when entering the filename.

Option 3 — If you are not sure of the exact spelling of a file or program's name, this option allows you to enter only the first character of the name. *DBOS9* then finds the first record in the database for which the filename starts with the entered character. Subsequent records are displayed as explained below.

Option 4 — Use this option to see all filenames stored in *DBOS9* database records.

Option 5 — This option returns you to the main *DBOS9* menu.

Whenever a *DBOS9* record is displayed (by options 1, 2 or 3 above), an action list appears below it on the screen. Options available on this list allow you to change the displayed record, delete the record, continue to the next record (not available for records displayed using Option 2), write the record to disk, or return to the Search menu. If you choose to change the record, you *must* write it to disk before exiting the screen. Otherwise, any changes made are not recorded in the database file.

For Safety's Sake

The third option on *DBOS9*'s main menu is Backup. Selecting this option allows you to create a backup set of the `os9` and `keyword.dat` files. The default backup is performed the from `/dd` device to the `/d0` device. If `/d0` *is* the `/dd` device on your system, the backup files are written to the same disk. (These default device selections appear in the function `os9top.c`, Listing 1. You can change them and recompile the source code.)

The backup files are stored using the names `os9.bak` and `keyword.bak`. If you later need to use these files with *DBOS9*, you must change their names to `os9` and `keyword.dat`, respectively, and place them in the `/dd/BASE9` directory.

DBOS9 is written in C and is made up of five functions: `os9top.c`, `os9search.c`, `os9enter.c`, `os9ch_del.c` and `keyword.c`. `os9top.c` contains the main menu and the database-initialization operation. It also handles the data-file backup. `os9search.c` contains all of the search functions and uses `keyword.c`. `os9enter.c` handles the entry of new data and also uses `keyword.c`. `os9ch_del` contains the change and delete functions, which are called by `display()`. `keyword.c` handles all keyword manipulation and contains the function `display()`, which displays file data on the screen.

The program uses what is called mutual recursion during some of the operational sequences. For instance the logic might procede as follows: `os9search()` calls `keyword()` which calls `display()` which calls `change()` which calls `keyword()`. In such a scenario, `keyword()` is used twice, but it is stored in memory only once.

Phil Scherer is a mechanical-design engineer for automatic packaging and assembly systems. In addition to working with OS-9 on the CoCo, his hobbies include snorkeling and horticulture. He can be contacted at 6191 NW 34 Hwy., Ft. Lauderdale, FL 33309. Please include an SASE when requesting a reply.

Listing 3: `os9ch_del.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

extern struct db9{              /*reference global variables*/
       char key[15];
       char name[25];
       char descr[3][60];
       char disk[15];
       }os9;

extern struct tempo{
       char key[15];
       char name[25];
       char descr[3][60];
       char disk[15];
       }temp;

extern int count;
extern FILE *fp;
extern long t;

change(z)
    int z;{
    char ch;
    int x,y;

 do{                    /*force valid selection with loop*/
    CurXY(1,0,10);
    ErLine(1);
    printf(" Select Number To CHANGE or <C/R> To Quit: ");
    ch=getchar();
    if(ch=='\n'){
        ch='5';
        break;}
   }while(ch<49||ch>52);

    CurXY(1,0,10);
    ErLine(1);

    switch(ch){
       case'1':x=1;
               keyword(x);
                   z--;
               break;
       case'2':printf(" Enter New NAME or <C/R> To Quit: ");
               strcpy(temp.name,os9.name);  /*store current name in os9.temp*/
               y=readln(0,os9.name,25);
               os9.name[y-1]=0;
               if(!*os9.name)
                   strcpy(os9.name,temp.name);  /*if quit, restore os9.name*/
               else{
                   for(y=0;os9.name[y];y++)
                       os9.name[y]=toupper(os9.name[y]);}  /*convert to*/
               z--;                                        /*upper case*/
               break;
       case'3':Clear(1);
               CurXY(1,25,2);          /*new description entries*/
               printf("DESCRIPTION");
               printf("\n %s",os9.descr[0]);
               printf("\n %s",os9.descr[1]);
               printf("\n %s",os9.descr[2]);
/*store os9.descr in temp.descr*/
               strcpy(temp.descr[0],os9.descr[0]);
               printf("\n\n Enter New Description or <C/R> To Quit");
               printf("\n 1: ");
               y=readln(0,os9.descr[0],60);
               os9.descr[0][y-1]=0;
               if(!*os9.descr[0]){
                   strcpy(os9.descr[0],temp.descr[0]); /*if quit, restore*/
                   z--;                                /*os9.descr*/
                   break;}
               printf(" 2: ");
               y=readln(0,os9.descr[1],60);
               os9.descr[1][y-1]=0;
               printf(" 3: ");
               y=readln(0,os9.descr[2],60);
               os9.descr[2][y-1]=0;
               z--;
               break;
       case'4':printf(" Enter New Disk Identifier: ");
               strcpy(temp.disk,os9.disk);   /*store os9.disk in temp.disk*/
               y=readln(0,os9.disk,15);
               os9.disk[y-1]=0;
               if(!*os9.disk)
                   strcpy(os9.disk,temp.disk); /*if quit, restore os9.disk*/
               z--;
               break;

             }
       return z;
    }

delete(z)
     int z;{
    char ch;

    CurXY(1,0,10);
    ErLine(1);
    printf("Delete This File?? Y or <N>: ");
    ch=getchar();
    if(ch=='Y'||ch=='y'){
        z=20000;           /*if file is deleted, return 20000*/
        strcpy(os9.name,"??");

/*locate file pointer for the seek function and write deleted file*/

        t=ftell(fp);
        fseek(fp,t-sizeof(struct db9),0);
        fwrite(&os9,sizeof(struct db9),1,fp);
        Clear(1);
        CurXY(1,10,5);
        printf("File Deleted-Press Any Key: ");
        ch=getchar();}
     else
        z--;
    return z;}
```

Listing 4: os9enter.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

extern struct db9{              /*reference global variables*/
       char key[15];
       char name[25];
       char descr[3][60];
       char disk[15];
       }os9;
extern struct tempo{
       char key[15];
       char name[25];
       char descr[3][60];
       char disk[15];
       }temp;
extern int count,recurse;
extern FILE *fp;

 os9Enter(){

       register int y;
       char ch;
       int x,j=1;


/*j is passed through keyword() to allow an abort
selection to occur in keyword(). If it returns greater
than 1 then the rest of code is skipped.*/

       recurse=0;
       j=keyword(j);

       if(j==1){

           OWSet(1,1,10,5,60,7,0,1);              /*create enter window*/
           OWSet(1,1,14,6,52,5,0,2);
           CurXY(1,10,1);
           printf("ENTER FILE NAME or <C/R> To Quit");
           CurXY(1,0,3);
           printf(":");
           y=readln(0,os9.name,25);
           os9.name[y-1]=0;
           if(!*os9.name){                        /*test for C/R*/
               OWEnd(1);
               OWEnd(1);
               return;}
           for(y=0;os9.name[y];y++)               /*convert to upper case*/
               os9.name[y]=toupper(os9.name[y]);
           x=0;
           rewind(fp);
           fread(&count,sizeof count,1,fp);    /*read file count*/
           for(y=0;y<count;y++){
               fread(&temp,sizeof(struct tempo),1,fp);   /*read each file*/

/*record the location of first deleted file.
allows overwriting unused space in file storage area.*/

               if(!x){
                   if(!strcmp(temp.name,"??"))
                       x=y+1;}
               if(!strcmp(os9.name,temp.name)){  /*watch for existing file*/
                   Clear(1);
                   printf("\n The file Already Exists-Press Any Key: ");
                   ch=getchar();
                   OWEnd(1);
                   OWEnd(1);
                   return;}}
            OWEnd(1);
            OWEnd(1);
           OWSet(1,1,5,5,70,11,0,1);  /*create file description window*/
           OWSet(1,1,9,6,62,9,0,2);
           CurXY(1,10,1);
           printf("ENTER FILE DESCRIPTION - THREE LINES\n");
           printf("\n:");
           y=readln(0,os9.descr[0],60);     /*file description entries*/
           os9.descr[0][y-1]=0;
           printf("\n:");
           y=readln(0,os9.descr[1],60);
           os9.descr[1][y-1]=0;
           printf("\n:");
           y=readln(0,os9.descr[2],60);
           os9.descr[2][y-1]=0;
           OWEnd(1);
           OWEnd(1);
           OWSet(1,1,10,5,60,5,0,1);     /*create disk i.d. window*/
           OWSet(1,1,14,6,52,3,0,2);
           printf("\n Enter Disk Identifier: ");
           y=readln(0,os9.disk,15);
           os9.disk[y-1]=0;
           OWEnd(1);
           OWEnd(1);
           OWSet(1,1,2,2,76,16,0,3);    /*create display window and*/
           OWSet(1,1,3,3,74,14,0,4);    /*display file*/
           OWSet(1,1,4,4,72,12,0,2);
           display(j);
           CurXY(1,8,9);
           printf("Do You Want To Enter This File? Y or <N>: ");
           ch=getchar();
           if(ch=='y'||ch=='Y'){
               rewind(fp);
               fread(&count,sizeof count,1,fp);  /*read file count*/
               if(!x){
                   x=count;
                   count++;}
               else
                   x--;
               rewind(fp);
```

Program listing continued

```
/*write count with new number then seek to the end of the last
file and write new file.*/

            fwrite(&count,sizeof count,1,fp);
```

```
            fseek(fp,((long) sizeof(struct db9))*x,1);
            fwrite(&os9,sizeof(struct db9),1,fp);}
            OWEnd(1);
            OWEnd(1);
            OWEnd(1);}
    return;}
```

Listing 5: keyword.c

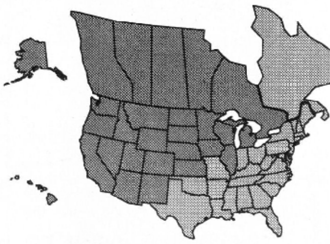
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

extern struct db9{                    /*reference global variables*/
    char key[15];
    char name[25];
    char descr[3][60];
    char disk[15];
    }os9;
extern int count,recurse;
extern long t;
extern FILE *fp,*kp;
extern char keywords[15][15];

keyword(j)
    int j;{

       register int y,x=0;
       char ch,num[4];
       int u,w,z,v=0;

      if(!recurse){
          if((kp=fopen("/dd/base9/keyword.dat","r+"))==NULL)
              exit(1);
          recurse++;}
      rewind(kp);                     /*read in keyword entries*/
      for(y=0;y<15;y++){
          fread(keywords[x],15,1,kp);
          if(keywords[x][0]!='?')
              x++;}
      if(j==3||j==4||j==5)
         if(!strcmp(keywords[0],"??")){   /*check for keyword entries*/
             OWSet(1,1,2,2,30,3,0,4);
             printf("\n No Keywords-Press Any Key: ");
             ch=getchar();
             OWEnd(1);
             return ;}
      if(x<15){                       /*insert ?? in unused file positions*/
          for(y=x;y<15;y++)
              strcpy(keywords[y],"??");
              y=0;}
       x--;
       OWSet(1,1,3,0,22,x+7,0,4);    /*make keyword window sized to*/
       OWSet(1,1,4,1,20,x+5,3,2);      /*the number of keywords*/
       printf("\n      KEYWORDS\n");
       if(keywords[0][0]!='?'){       /*list existing available keywords*/
           for(y=0;y<=x;y++)
           printf("\n %d) %s",(y+1),keywords[y]);}

/*j==1 is used by enter and change*/

       if(j==1){
           if(x<14)
               printf("\n %d) New Keyword",(y+1));
           OWSet(1,1,23,10,48,3,0,1);
           printf("\n Pick a Number or <ENTER> to Quit: ");
           for( ; ; ){                /*force a valid selection with loop*/
               y=readln(0,num,3);
               num[y-1]=0;
               if(*num){
                   y=atoi(num);
                   if(y<1||y>(x+2)||y>15){
                       Clear(1);
                    printf("\n Invalid Number-Re-Enter or <C/R> to Quit: ");}
                    else {
                        if(recurse>1||y>0)
                        break;}}
               else {
                  j=15;    /*C/R message is carried back by j=15*/
                  break;}
              }
              if(j!=15){
                if(y>0&&y<=(x+1))      /*enter selection into new file*/
                    strcpy(os9.key,keywords[y-1]);
                else if(y==(x+2)){     /*enter new keyword if selected*/
                    Clear(1);
                    printf("\n Enter New Keyword: ");
                    x++;
                    y=readln(0,keywords[x],14);
                    keywords[x][y-1]=0;
                    if(keywords[x][0]==0){
                         *num=0;
                         j=15;   /*C/R is returned by j=15*/
                         strcpy(keywords[x],"??");}
                    if(*num){          /*convert to upper case*/
                        for(y=0;keywords[x][y];y++)
                            keywords[x][y]=toupper(keywords[x][y]);
                       strcpy(os9.key,keywords[x]);}}  /*record new keyword*/
            rewind(kp);
            fwrite(keywords,sizeof keywords,1,kp);  /*rewrite keyword.dat*/

        }}

/*j==3 is used by search functions*/

         else if(j==3){
               OWSet(1,1,23,10,48,3,0,1);
               printf("\n Select Number To Search or <C/R> To Quit: ");
               do{                    /*force valid selection with loop*/
                  y=readln(0,num,3);
                  num[y-1]=0;
                  if(*num){
                      y=atoi(num);
                      if(y<1||y>(x+1)){
                          Clear(1);
                  printf("\nInvalid Number-Re-Enter or <C/R> To Quit: ");}
                      else  *num=0; }
                      else y=0;
                  }while(*num);
              if(y>0&&y<(x+2)){   /*after selection read file count*/
                  rewind(fp);
                  fread(&count,sizeof count,1,fp);
                  w=0;
                  for(z=0;z<count;z++){      /*read each file*/
                     fread(&os9,sizeof(struct db9),1,fp);
                     t=ftell(fp);
                     u=z;
      if(!strcmp(os9.key,keywords[y-1]))     /*check for keyword match*/
            if(strcmp(os9.name,"??")){     /*check for deleted file*/
                do{
                    z=u;
                      if(w==0){
                         OWEnd(1);
                         OWEnd(1);
                         OWEnd(1);
                         OWSet(1,1,2,2,76,16,0,3);
                         OWSet(1,1,3,3,74,14,0,4);
                         OWSet(1,1,4,4,72,12,0,2);}
                       w++;   /*increment w to prevent window repeat*/
                       z=display(z,j);
                       if(z==20000){   /*20000 returned means delete*/
                           v=1;
                           fseek(fp,t,0);
                           z=u;}
                      Clear(1);
                   }while(u>z); }}
                   if(z==count){   /*send message after all files*/
                       Clear(1);              /*have been read*/
                       CurXY(1,12,5);
                       if(w==0){
                           v=1;
                       printf("There Are No %s Files\n",keywords[y-1]);}
                       else
                  printf(" There Are No More %s Entries.\n",keywords[y-1]);
                              CurXY(1,12,6);
                              printf(" Press any Key: ");
                              ch=getchar();}}}
                     if(w<2&&v==1){   /*delete unused keyword*/
                         strcpy(keywords[y-1],"??");
                         rewind(kp);
                         for(y=0;y<15;y++)
                             fwrite(keywords[y],15,1,kp);}
       recurse--;
     if(!recurse)
        fclose(kp);
        OWEnd(1);
        OWEnd(1);
        OWEnd(1);
        return j;

    }

display(z,j)
    int z;
    int j;{
        char ch;
        long t;
        Clear(1);
        printf("\n 1)%s  ",os9.key);
        CurXY(1,20,1);
        printf(" 2)%s\n",os9.name);
        printf("\n 3)");
        FColor(1,3);
        printf("%s\n",os9.descr[0]);
        printf("   %s\n",os9.descr[1]);
        printf("   %s\n",os9.descr[2]);
        FColor(1,0);
        printf("\n 4)");
        FColor(1,3);
        printf("%s",os9.disk);
        FColor(1,0);
        if(j!=1&&j!=2){   /*test j for the correct response*/
        do{    /*force a valid selection with a loop*/
            CurXY(1,0,10);
            printf(" <C>hange-<D>elete-");
            if(j!=4)
                printf("Co<N>tinue-");
            printf("<W>rite or <C/R> to Quit: ");
            ch=getchar();
            ErLine(1);
            if(ch=='C'||ch=='c'){
                z=change(z);
                return z; }
            if(ch=='D'||ch=='d'){
                z=delete(z);
                return z;}
            if(ch=='W'||ch=='w'){
                t=ftell(fp);
                rewind(fp);
                fread(&count,sizeof count,1,fp);
                fseek(fp,((long) sizeof(struct db9))*z,1);
                fwrite(&os9,sizeof(struct db9),1,fp);
                fseek(fp,t,0);
                z--;
                return z;}
            if((ch=='n'||ch=='N')&&j==4)  /*disable n selection if j==4*/
                ch='x';
        }while(ch!='\n'&&ch!='n'&&ch!='N');
        if(ch=='\n')    /*if C/R is entered return z>count*/
            z=count+1;
            return z;
            }
        else {if(j!=1){
            RevOn(1);
            CurXY(1,0,9);
            printf(" Press Any Key To Continue: ");
            RevOff(1);
            ch=getchar();}}
            z=count-1;
            return z;}
```

Advertisers Index

We appreciate your mentioning THE RAINBOW *when you contact these advertisers.*

Call Ira Barsky
Western Sales Director
(312)587-1818

Call Kim Lewis
Eastern Sales Director
(502)228-4492



The Falsoft Building
9509 U.S. Highway 42, P.O. Box 385, Prospect, KY 40059
(502) 228-4492 • FAX (502) 228-5121

OWL-WARE

Proven Technology

On the Razor's Edge of the Color Computer Frontier

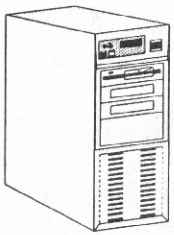
NOW, 120MB Hard Drive Standard

In our 10th Year !

A DECADE OF SERVICE TO THE COMPUTER USER!

486SX-20 SYSTEMS - $1495.00!

Now You can enter the world of 486 computing at a reasonable cost!



the OWL SUPER ATOM - 486

High Powered Computing from a local, well established company.

- **33MHz / 50MHz i486 based Systems with Socket for Weitek CoProcessor**
- **System and Video BIOS in Cache**
- **Large Tower Case : (33MHz, FCC Class B) - (50MHz, FCC Class A)**
- **230 Watt Power Supply & 8 Option Slots**
- System Price includes 120MB HD, 4MB RAM, Std. Resolution Color VGA Monitor, High Resolution VGA Card, 2 High Density FD's, MS-DOS 5.0

$1495 (486SX-20) / $1795 (486-33) / $2095 (486DX-50) / $2185 (486-DX2-66)

- Super VGA Upgrade

3- YEAR WARRANTY on all systems! Including One Full Year on Parts and Labor. Manufactures 3-Year Warranty on All Hard Drives

OWL SUPER ATOM - 386

- **25/40MHz 386DX Based**
- **Small Footprint Case**
- **FCC Class B Approved**
- **200 Watt Power Supply**
- **7 Expansion Slots**
- **4MB of RAM**
- 120MB Hard Drive
- **Std. Resolution VGA Color Monitor**
- **2 High Density FD's**
- **101 Keyboard**
- **MS DOS 5.0**

$1495 (40MHz)

OWL SUPER ATOM - SX

- **16/25MHz 386SX Based**
- **Small Footprint Case**
- **FCC Class B Approved**
- **200 Watt Power Supply**
- **7 Expansion Slots**
- **2MB of RAM**
- 120MB Hard Drive
- **Std. Resolution VGA Color Monitor**
- **2 High Density FD's**
- **101 Keyboard**
- **MS DOS 5.0**

$1195 (25MHz) / $1295 (33MHZ)

386-SX Notebook Computers

20MHz, 60MB HD, 1.4MB FD, 2MB RAM(exp. to 5MB), VGA 640X480 LCD w/32 shades of gray. Ports: 2 Ser, 1 Par, 1 VGA, DOS & Windows, 7.7LBS!

$1395

386-DX Notebook Computers

33MHz, 120MB HD, 1.44MB FD, 32KB CACHE, 4MB RAM(exp. to 16MB), Std. VGA LCD w/32 gray, Ext. Keypad inc., DOS & Windows, 7.7LBS!

$1895

OWL COMPUTER SERVICES

5950 Keystone Drive
Bath, PA (215)-837-1917 (800) 248-6228

Kids & Us - RadioShack® Dealer	*St. Onge Systems*	*Computers & Games*
Pottstown Ave., RT. 663	*Wescosville*	*Muhlenberg Shopping Plaza*
Pennsburg (215)-679-3389	*Call for Appt. (215)-481-9775*	*Reading (215)-929-0540*

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) ~~$180.~~

SOLD OUT!
WE NEED CONTROLLERS!

IF YOU HAVE 502 CONTROLLERS, CALL US!

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

New 3.5", 720K Drives for OS-9 with case & Power Supply **$129. SALE!**

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) $199.

(WITHOUT CONTROLLER)

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.

SALE $79.

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