Published Bi-Monthly

Vol. 2 #5 May/Jun. 1989
$2.75 U.S.A.
$3.75 Canada

CoCo Clipboard Magazine™

THE NEWEST, MOST INDEPTH MAGAZINE FOR TANDY'S COLOR COMPUTER 2 & 3

"Sound" Programming In Assembly

CCBMS Part III

Jim DeStefano Makes Revisions Based on Reader Input!

Intro to MIDI and CoCo

Dr. "Midi" Lester Hands shows you how!

Product Reviews

Including:

RASCAN
VIDEO DIGITIZER

Simply Better Word Processor

KJV Bible on Disk

Window Master

KenTon's Hard Drive Interface

Game Point Software's Games

Leonardo's Paintbox

Plus Our Regular Columns and Features!

<<< GIMMESOFT >>>

A new generation of Color Computer products

MAXSOUND

RAINBOW CERTIFICATION SEAL

A High Quality Digital Audio Sampler and Sequencer

Turn your CoCo III into a REAL digital audio sampler with HIGH quality audio reproduction. Easily add exotic effects, ECHO, stuttering, speed shifting, sequencing, and reverse audio to BASIC or ML programs or GRAPHICS! Now includes Data Compression. Imagine recording any Voice, Music, or Sound effect and being able to use these DIGITAL recordings in your own programs! 3 disk sides includes: INTERFACT/BIN - ML driver for sound effects. G&M/BAS - Adds sound effects to Graphics. SHOWTIME and DEMO disks. SCOPE/BAS - Turns CRT into a Digital Oscilloscope to look at MAXSOUND waveforms. Version 3.0 upgrade (Includes improved ECHO and the ability to print NAMETAGS and locations to the screen and/or printer) ..... $6.95 + Shipping & Handling

Call to hear 'OVER THE PHONE' Demo - 9am to 9pm VOICE only.
DOWNLOAD Demo Files 300/1200/2400 24 hrs - 301-675-7626 MODEM only.
Requirements ..... (128k or 512k CoCo III only) DISK $59.95

Games

See previous Clipboard ads for complete descriptions.

Warrior King	$29.95
In Quest of the Starlord	$34.95
Kung-Fu Dude	$24.95
Pyramix	$19.95
Hall of the King I,II or III	each $29.95
Dragon Blade	$19.95

Utilities

Fkeys III	$19.95
Sixdrive	$16.95
Multi-Label III	$16.95
AD&D Companion	$24.95
MPI-CoCo Locking Plates	
26-3024 or 26-3124	$ 7.95

V-Term Terminal Emulator

VER. 3

RAINBOW CERTIFICATION SEAL

Communicate with VAX, UNIX, Mainframe, and BBS Systems!

-VT-100, VT-52, Vidtex (includes RLE graphics display), and standard CRT emulations.
-Menus can be operated concurrently with other terminal functions. (Disk Basic!)
-Full 28 line by 80 column screen, with 3 bottom lines protected for menus.
-Serial port up to 2400 baud, RS-232 Pak up to 19,200 baud, DCModem Pak at 300 baud.
-XModem, XModem-CRC, Y-Modem, and ASCII file transfers directly to disk or memory.
-RAMDISK like buffer, Capture buffer, Snapshot, Conference mode, and 35/40/80 Tracks.

NEW FEATURES: 15-Entry autodialer, 10 Programmable macro keys for each system, compatible with Hyper I/O and RGBDOS harddrive systems, and baud rates up to 19,200!

Version 03.00.00 upgrade ..... $6.95 + S&H Disk (128k or 512k CoCo III only) $39.95

Telepak II

(CoCo I/II/III) **A TRULEY COMPATIBLE RS-232 INTERFACE!**

Now, from Orion Technologies, comes the answer to the continuing demand for an RS-232 interface. Telepak II now includes a 3 foot DB25 cable, gold card edge contacts, and low power drain (5v) components. Works on ALL Color Computers with or without a Multi-Pak interface. (MPI required on disk systems) Baud rates up to 19,200! (19.2 tested using V-TERM ver 3). Only $49.95

Toll Free 1-800-441-GIME Order Line

Technical assistance: 7pm to 9pm
Orders: 9am to 9pm Eastern time
On-line orders and up to date information: Delphi's CoCo Sig

GIMMESOFT
P.O. Box 421
Perry Hall, MD 21128
301-256-7558 or 301-256-2953

Add $3.00 for shipping and handling
Add $3.00 for COD (USA only)
MD residents add 5% sales tax
VISA/MC/Check/Money Order/COD

CoCo Clipboard Magazine
Vol. 2 Issue 5 May/June 1989

ISSN 0899-6415

Editor and Publisher
Theodore W. Paul, Jr.

Associate Editor
Darlene M. Paul

Contributing Writers: Rush T. Caley, Jim DeStefano, Kraig Brockschmidt, Mike Dooley KE4PC, Bob van der Poel, Bill Brady, Randy Krippner, Paul Bornemann, Shelby LaMont

Printed by: Falconer Printing & Roach Photography

CoCo Clipboard Magazine™ is published bi-monthly (6 times a year) by:

CoCo Clipboard Magazine
3742 U.S. 20, Box 3
Fredonia, N.Y. 14063
(716) 679-0126

Subscriptions to CoCo Clipboard Magazine are:
$15.00 U.S. Funds per year for 6 issues.
$28.00 U.S. Funds for 12 issues. We accept checks or money orders only. Canadian subscriptions are $18.00 in U.S. Funds. All overseas subscriptions are $27.00 U.S. Funds surface mail.

Orders may be sent to:
CoCo Clipboard Magazine
3742 U.S. 20, Box 3
Fredonia, N.Y. 14063

Full refund after one issue. No refunds after two issues.

Your comments and suggestions are welcome. We reserve the right to edit and publish all letters received unless requested not to by the writer.

PLEASE NOTE: Subscription orders received AFTER the first week of a publication month will begin with the NEXT ISSUE. We cannot start subscriptions with back issues.

CoCo Clipboard Magazine

Vol. 2 Issue 5 May/June 1989

TABLE OF CONTENTS

Please Join Us

For our CoCo Clipboard Conferences of CompuServe!

Our on-line - LIVE - sessions are scheduled for the second Saturday of each month at 9pm Eastern.

Join Dan Robins and Ted Paul for up to date information every month!

FROM THE DESK OF...

Ted & Darlene Paul

Just a quick note to those of you who no longer take the "other" computer magazine. *Clipboard* got a mention (more by inference than by name) in their editors's column! Thanks Lonnie! The phone has been extra busy ever since!

Now that "that" is out of the way on to more important business.

First of all some quick product updates that you should know about and probably won't hear about for months EXCEPT in these pages. Bren Stockdale informs us that his Telepak products now come with gold edged connectors AND a 3 foot DB-25 cable. We've used his fine RS232 Telepak for months and have never experienced a drop of trouble.

Second, Bill Vergona has informed us of a new application program to run under Window Master (please read Randy Krippner's review in this issue) called Window Writer. As soon as we get a copy we will get it reviewed.

(As a side note to our advertisers and potential advertisers: If you're sick to death of waiting months and months for a review stop holding your breath and send us a copy. In most cases we can have a review done by our next issue. One ADVANTAGE of publishing a smaller magazine every other month is that we can respond FASTER to changes in the CoCo market. For example see our exclusive story on Game Point Softwares' new Video Digitizer "RASCAN" and be sure to see their ad! Secondly our reviews don't get lost in page after page of clown pictures, re-writes of old articles and 50 ways to lose your disk directory.)

Speaking of Game Point Software, Pete Ellison has informed us that they are announcing a new picture digitizer for the CoCo 3 that has a "40-96" mode. This "40-96" mode will let you store 3 images from a camcorder, one each of blue, red and green and then combine them for approximately a digitized 4096 color image. I will have a report in the review section.

We are pleased to welcome to our advertiser group SEVEN new advertisers, Dorsett Educational Systems, After Five Software, Game Point Software, Arizona Small Computer System, Tomela Co., B.S.I., and Kenneth Leigh Enterprises. We are happy to have them on board and we appreciate their support of *CoCo Clipboard Magazine.* We are also announcing updated billing policy for our advertisers which will make their bugeting much, much easier. Details of this procedure will be sent directly to our advertisers.

Clipboard has been on the road in April and will be traveling again in May. We were pleased to participate in the 1989 NUAGE Computer Fair in Erie, PA. This is a "user group" sponsored show and had many types of computers, vendors and users groups on hand. My thanks to Dennis Skala for the invitation to join the OS9 discussion after the show - sorry I didn't get there but I zigged when I should have zagged trying to find the K-Mart near Peninsula Drive! I trust our visit to the Allentown PA group and Mr. Peter Unks will fare better. We'll be heading down to Eastern Pennsylvania in May. If your group would like a visit from *Clipboard* and you're within 500 miles of Buffalo, New York please give me a call. There's never a charge to your group. Since the visit to the Allentown club is Friday night our Saturday is open - Trenton, Philadelphia, Princeton, New York City and the Long Island are all nearby - let me hear from you!

In this issue we've got more "no nonsense" articles as we continue with our C.C.B.M.S. series. This edition takes a brief break to do some updates and explain where we have been and where we will be going. If you've been looking for a REAL business application package for your CoCo 3 and where you can actually have some input on how it will be created join in! (Oops, you don't have the back issues? No problem give us a call at (716) 679-0126 and have your credit card ready. Back issues are ALWAYS available.) You probably don't know it but every one of our issues are real multi-page magazines. Some places charge you top dollar for a two page re-print. Our first issue wasn't two pages photocopied at the drugstore or 6 or eight or 10. Our first issue was 28 pages and has grown ever since. As a matter of fact when we build an issue of *Clipboard* we plan to ADD pages. With the number of high quality programs and articles

From The Desk continued on 5

From The Desk continued from 4

out there that need to be printed I just wish we could print more! And our subscription count is up as well. Like other magazines we don't quote "readership" - just isn't honest. Our subscription count goes up a little every day.) Sorry about the commercial...

Kraig Brockschmidt is back with a machine language sound program which is very detailed and extensive and Dr. Lester Hands the author of LYRA gives us some hints and tips on the CoCo and Midi Music making. Robert Gault also joins us with a technically oriented piece on disk drives and many other things. Robert is talking with Jim DeStefano on a special addition to the C.C.B.M.S. program which looks like a winner and I'll be asking Robert to handle a wider variety of tasks in upcoming issues.

Did you read Jim DeStafeno's piece last issue? He mentioned how he was running two CoCo's from a single hard drive via RGB DOS and the Kenton Electronics SCII board. I talked with Joe Scinta, Kenton's owner, and he has a major new product to announce - a 30 meg. hard drive that fits right into the empty slot on a Radio Shack FD-502 drive. The unit sells complete for under $500! Is anybody in Ft. Worth reading this magazine? I hope so, because Joe has promised us an article or two on just how this magic works! Joe is an accomplished electronics engineer and really knows his stuff. Remember where you'll read this information FIRST!

We are also looking for more articles on Amateur Radio. You're input is needed, but that doesn't mean we're sitting on our better intentions here in Fredonia. I've got a couple of letters out right now and if everything clicks I am hoping we can announce a major international programming effort for the CoCo and Ham Radio. You know with the "Shack" now selling a good quality 10 meter unit and an easily modifiable magnet mount C.B. antenna it's the perfect time to get into this exciting and "compatible" hobby.

Also under study for these pages are articles on using your CoCo with astronomy and I'd like to hear from anybody that is using a CoCo with model railroads. I'd be interested in running articles on anyone operating a multi-line BBS on the CoCo and the FIDO-NET network. Frankly friends I don't know of any other CoCo magazine that is making the same effort we are in expanding the use of the CoCo and bringing that information to you - issue after issue!

I know I promised some small business information in this column but I'm going to put it off till next time. We've go too much interesting stuff to print this issue. Until July thanks for your support of *CoCo Clipboard Magazine.*

CoCo
Clipboard
Magazine

RASCAN

VIDEO DIGITIZER

The Rascan Video Digitizer is a state-of-the-art image processing system designed to take advantage of your Color Computer 3's graphic capabilities.

The Rascan Video Digitizer connects easily to any color or black & white video camera, video recorder or video disc player and captures images with precision accuracy.

Why settle for a 256 x 256 image area when the Color Computer can display so much more? We asked that question ourselves. Our only answer was to provide an *image area of 640 x 200 and 320 x 200!* Say good-bye to those useless lo-resolution images created by other digitizers on the market.

Life is not simply black & white, that's why we added living color to our Digitizer. Now, through the use of advanced programming techniques, 512K Color Computer 3 owners can capture images from their video camera and display them in *4096 Super Hi-Resolution graphics!*

Capture images effortlessly. Simply select **the image capture option** and turn your **Rascan unit on. Within seconds your image will be captured and displayed** on your screen. **Images can be fine tuned** by use of the contrast and **brightness** knobs found on the Rascan unit.

Rascan also features a *professional pop-up menu system* which will allow for easy palette manipulation and color painting of captured images.

The Rascan Video Digitizer *comes complete with Rascan driver software, an easy to read manual, sample graphic images disk and print driver disk* (supporting most printers). Although no further graphic editors are necessary to produce quality images, Rascan images can be easily loaded into ColorMax and CoCo Max graphic editors.

Whether your interests are in desk-top publishing, report generation or simply for fun, the Rascan Video Digitizer will provide you with images of near photographic quality!

FEATURES	RASCAN YES	RASCAN NO	DS-69b YES	DS-69b NO
Support of 640 x 200 16 Level Grey Images	X			X
Support of 640 x 200 4 Level Grey Images	X			X
Support of 320 x 200 16 Color Images	X			X
Support of 4096 Hi-Res Color Graphics in 512K mode	X			X
Support of Multiple Image Buffers in 512K mode	X			X
Control of Contrast & Brightness via Control Knobs found on Digitizer	X			X
Professional, Easy to Use Pop-Up Menu System	X			X
Designed Exclusively to Take Advantage of the power of the Color Computer III	X			X
Built in Histograph Utility to Aid in Image Quality	X			X
Easy to use Paint and Palette editing, no need for additional Graphic editors	X			X
15 Day Full Money Back Guarantee	X			X
Interface through Joystick Ports	X			X
Requires additional cost of Y-Cable or Multi-Pak interface		X	X	

THE RASCAN VIDEO DIGITIZER

$159.95

NO RISK GUARANTEE

If you are not completely satisfied with the performance of your Rascan Video Digitizer, you may return it, undamaged within fifteen days for the full refund of the purchase price plus shipping costs.

GAME POINT SOFTWARE

P.O. Box 6907, Burbank, CA 91510-6907
(818) 566-3571 • BBS: (818) 772-8890
Toll Free: 800 877-2232 ext. 139

MasterCard VISA DISCOVER

Personal checks, money orders, and American C.O.D. orders **accepted.** Include $3.00 for S/H. $2.50 extra for C.O.D. orders. (Cal. res. **add 6.5% tax.**)
ATTENTION PROGRAMMERS: Game Point Software is looking **for talented** writers. Top royalties guaranteed.

Reader Mail

Dear Clipboard:

I am a new subscriber to your magazine and I just got my first *ClipDisk* from your staff. I took advantage of the 6-2-1 offer you advertised and I strongly advise everyone else to do the same. I have spent many hours going through each issue I received and anxiously await my first regular issue.

MY COCO system is made up of a 512K CoCo 3 with two Tandy disk drives (white). In addition I have a Magnavox 8CM515 monitor, a Tandy DMP 132, a Tandy DWP 210, a Okimate-20 and just about all the standard Tandy soft= ware available for the CoCo 2 and 3.

I would like to learn the C language and Basic-09. The articles in your magazine have already helped me understand some Basic-09 so please keep up the work. I have dealt with OS-9 for almost three years now and have found that the more I know the more interesting it becomes. I have completed converting almost all of my Level I software to Level II. I am having trouble getting TSWORD to work on Level II but everything else seems to work. I also managed to get my drives to work as 40 track single sided 6ms drives. I am currently trying to decide what drives to get as replacements. Any advice you could offer would be very helpful. I had thought about 80 track drives but I understand that even ADOS-3 can't support these and still maintain compatibility with RS DOS.

I was very pleased to find a HANDWRITTEN note with my *ClipDisk* and would like to thank Mr. DeStafeno for taking the time to write that welcome note. It is the little extras like that, that makes a difference.

I typed in Mr. Krippner's Multi-Vue Icon Editor from your Nov. / Dec. '88 issue and even though I've checked my typing I can't get the Save Icon (svicon) procedure to work correctly. The file that is saved is only one byte long and when the Multi-Vue system displays the Icon it is actually a disk Icon with a bunch of dots placed randomly under it. That is not the Icon I designed. Other than that, the program works very well. I suspect that the problem has to do with the buffers but as yet I have not found it.

I think your magazine is FANTASTIC and I hope it's around for years to come!!

Terry Parks

Garden Grove, CA

Dear Terry:

First of all I will be sending a copy of your letter to Randy Krippner and ask him to look into the Icon save problem you have described. Many of our readers know that our writers usually have their HOME addresses printed at the end of their articles. Our writers WANT to hear from their readers and are more than willing to help. For those writers not posting an address you can drop us a note hear and we will forward the letter for you!

In regards to the choice of drives I can't make a recommendation as you can imagine. However I would strongly urge you to look into one floppy and then a hard drive! With the lower prices of hard drives and the availability software to work with these drives don't go with two floppies. Ken-Ton electronics now has a 30 meg. hard drive that will fit into the top slot of your FD 502 Tandy housing. You may also want to look into an "outboard" drive like those sold by Arizona Small Computer Peripherals.

And thanks for the nice wishes about our efforts. This latest issue has a 13% page increase - we're growing and it's all due to great support to readers and advertisers just like you !

Gentlemen:

Enclosed please find my subscription order.

I've had it with some of these other computer magazines dealing with kiddie stuff for far too many issues (I've tried them all). One has to grow up sometime and I'm hoping your magazine features something more than cutsie pie, animated circles and squares bobbing about on the screen (after too many minutes typing in a program of that nature) and/or ugly, poorly animated little gobblins chasing (and eating) other ugly little characters all over the screen.

I am going for the "big" deail; 6-2-1. If I have a choice of back issues please fill

Reader Mail continued on 9

Reader Mail continued from 7

the order with anything dealing with "getting started with OS-9" (I don't understand it, yet) and, please, mix that with anything having to do with amateur radio. I suffer from computer-itis as well as "ham"-onia. Both seem to be very debilitating diseases. By the way, not that you should care, but I have a CoCo 3 with 512K and just recently made up my mind to give OS-9 a shot. I am 780 years old and need all the help I can get in the tutorial sense. Old dogs and new tricks, you see! I have been fooling with CoCo computers for over five years, now, and cannot get it through my head that it is possible for these machines to do what they do. On the other hand, I'm still not convinced one can speak into a microphone in Arizona and be heard around the world; black magic, that's what it is.

Thank you for your help. I'll let you be the judge of the back issue deal (assuming I have a choice) "Getting Started With OS-9 articles" and anything pertaining to ham radio will suit me just fine. Until later I remain,

Jim Oberto
Phoenix, AZ

Dear Jim:

We got pretty tired of lousy graphics on our CoCo just like you did. Since some people *won't learn new tricks we stared CoCo Clipboard Magazine.*

We made a decision a while back regarding our 6-2-1 back issue policy that I think you'll like. The ad says you can get the "6 most recent back issues" and that policy stays the same. It's not only easier for us, but you're getting bigger, more meatier magazines! However we will send ANY back issue or issues you may want for your 6-2-1 order! All you have to do is ask and we will fill the order.

You better believe we are interested in what types of equipment you have Jim. The statistics we've gathered from the surveys we had in the last two issues help us tailor the magazine to our readers needs. Our advertisers need to know what's out there so as to better serve the customer base. Information like yours is always helpful.

Thanks for writing!

Dear Clipboard:

I certainly hope you're as good as you say you are. To me (an assembly programmer) "Rainbow" is worthless.

Name withheld
Green Bay, WI

While I've held back on allowing comments about our competition in the Reader Mail column it's about time to let our readers have their say. I withheld the writers name for his sake, but the letter is here, in my files, just as are all our letters - in case anyone wants to know.

That wraps up this column for another month. A word of thanks to Lee Veal in Rowlette Texas for all of his reviews and the massive four page letter he send back in November. If we can we'll try to print it soon even it's the only one in the column!

C.C.B.M.S. III

Jim DeStefano

(Note: This article refers to a series of Business Management programs published in *CoCo Clipboard Magazine* beginning with the Jan.'89 issue. It is not the policy of *CoCo Clipboard Magazine* to "bait" new subscribers into purchasing back issues, however it would be impossible to print this series of programs in one issue. Back issues of the magazine are available by calling (71) 679-0126 or see the back issue order page in this edition.)

Although the last article closed with the promise of an Accounts Receivable module in this issue, letters and phone calls tell there are some things that must be cared for first. This offering will address a bug, two glitches and a twist. If the roof stays on we'll catch the A/C module next time.

One of the glitches has to do with what was not discussed in previous articles. In addition to the program listings, we did talk about the objective of the series, how the programs work and recommended equipment. However, there was no mention as to why each module was written as it was. Part of this article will address that omission. This is needed because people want to make modifications, but are not sure which module should be changed to get the desired result.

In addition, we have a lot of new subscribers that need to be brought up to speed. The series objective is to write module 'spines' that can be tailored for use in the management of specific small businesses. The given program user can do the modification themselves or send in a help request. Either I or another member of the CCBMS Users Group will address the 'help' request. If you make mods yourself, we ask that you send them in so they can be kept on file for reference.

Those interested in using the spines for their own business and those interested in making mods are encouraged to write making their wishes known. To date, Bob Gault is working on a "chain" program. Roxann Brown of Brown's Secretarial Service has been the one with the most detailed "help" requests.

The mandatory equipment list includes a CoCo-3, a monitor (monochrome is fine), one floppy drive and at least an 80 column printer that does compressed printing. The highly recommended list includes a hard disk drive, 20Mb minimum, and a 132 column (wide) printer. A second floppy will work for a while, but if you don't have one, don't get one. A second floppy would be a wasted investment. (Write or call for specific hard disk recommendations.)

We have two program modules functioning, the Customer Maintenance File (CMF) "CSFLMT.BAS" and the Sales Order Entry File (SOE) "SLORNT.BAS"; plus a system menu module, "SYSMNU.BAS". Both modules "sport" in memory search; which results in faster operation. SOE reduces the total stored data quantity by sharing (not duplicating) data that is already in the CMF module. This sharing of data, and using pointers in the data files makes the programs a bit difficult for programmers to follow, but that is the best method I know to get what we need; minimum storage space and maximum speed.

There will be several more modules, A/R, Inv., A/P, G/L, etc., as determined by requests. The package is written in modules for specific reasons. The two main determinants are external memory conservation and operation speed. With that objective firmly in mind, we can better determine what modules should be modified for what purposes. As stated in the beginning of the series, "I ain't no programmer", so we all know the code could be written more efficiently. However, the aim is to get a good user's package working. The code can be made shorter, etc. later.

The clue to determine what mods go where revolves around the design philosophy of each module. In the previous issues there was no mention made of the WHY. Lets take a look at that. A logical question might be, "Why not combine CMF and SOE into one module? After all, one can't have a sales order without a customer." That is true enough, but look at the nature of the data in each module and how the data effects speed and required external storage space.

The data in the CMF file is of the permanent type. Normally it will not be deleted or even changed, while a sales order leans very much the way. That being the case, we can make a rule, CMF should have only non-changing data and no fields that will be left without data. That leaves CMF with very

CCBMS continued on 11

CCBMS continued from 10

basic information about the customer; name, location, phone number, etc.

CMF has three fields that tend to violate the rule to make the results a little more flexible. There is a 32 character comment field and two 3 character fields; "Type" and "Class". All the other modules are (will be) setup so the three fields always screen print, but the comment field never prints on any report except the customer file print out. The Type and Class fields will print on every customer report except those that will be seen by the customer. (For our purposes, lets say anything printed out is to be considered a report.)

But why is the CMF restricted to just this unchanging information? Well, the information in the CMF will always be on the disk, consuming space, no matter if there is data in all the fields or not. The more space each customer is allocated, the more space that will be used. There is a way to keep that from happening, but then the program is slow when looking for the data. Therefore we put only data in the CMF module that is surly needed to "know" the customer and record sales properly.

Roxann noticed CMF doesn't provide for the deletion of a customer. She requested that ability be put in, saying, "Some customers do move away." Of course she is right. However, the module makes the customer's number and the customer's record number, the same number. If the computer knows or the operator tells it either one, the record can be gotten without going through an intermediate look-up table. Under such conditions, the request does not result in a search, but rather a "go and get" request.

That means even if a customer becomes "inactive" their record can not be cleared and used by another customer. It can't be done because of historical records. The programs differentiate between customers by customer number, not name.

The historical data generated by a computer is one of the three main reasons to computerize a business; the other two are error reduction and reduced "working on the books" time. The deduced business historical information may be the most important of the three. I guess every business owner knows which is his top customer. However, few if any know the top ten, in order, or the top ten inventory items; not usage, but rather in gross profit producing order. Computerization yields that important information as a by-product.

Just the opposite from CMF, the sales data retained by the SOE module is more temporary. There are two types of business sales. There is the sale that is paid for at the time of the purchase. It is normally called a POS (Point of Sale) business. The other general category is called a DP (Deferred Payment) business. The Sales Order Entry (SOE) module does not directly address the POS method. (I'm hoping someone involved with a Point of Sale business will adapt SOE to that need.)

Since the Deferred Payment business' SOE is more complex, SOE was written to cover that situation. SOE has the same needs, speed and minimum storage space requirements as CMF, but the nature of the data is different. That is, if there is no trouble, once a sale has been paid we no longer need the sales order; not like we need the customer data, which is used over and over again.

However, we do want to be able to refer to the sales order if there is any question about the information on it. So we do want to retain the sales order on-line for a time, but not indefinitely, as we do with the customer information in the CMF module.

The design logic of SOE is much like CMF, but uses two data files to save external storage memory. The sales order number and its record number are the same number. Again, once either one is known the data can be retrieved quickly. This also means no sales order can be deleted, only deactivated or closed. However, that does not mean the intent is to keep all sales orders on line forever. There is no need to do that, and if we did, expensive on-line storage space would be wasted, plus a search for a given sales order would be slower.

In general, few businesses will need to keep a sales order on-line for longer than two years. As such, we can do a slight of hand that will allow us to have our cake and eat it too. The plan is to change the extension of both SOE data files to identify them by the year their data was (is) accumulated. When a file becomes two years old it will become inactive due to its date. That way we will be able to reuse the sales order number (and record number) from year to year. We would surely recognize sales order #1 committed in 1989 to be a different order from sales order# 1 committed in 1988; don't you agree? Changes will have to made to the module to comply with the above, but that will be easy to do.

In the last issue we discussed why SOE has two data files. Let it enough here to say that since a sales order can have any number of items, we would waste a lot of external memory if mandated a sales order to allow any fixed number of line items. The two files keep us from wasting any memory due to different length sales orders.

The important point to keep in mind, don't add data fields to CMF because it keeps its data on-line forever. Other modules will keep their data on-line for a fixed length of time and then deactivated. Therefore it would be better to put any additional data in them. Different modules are written because different types of data; Inv., G/L, etc. are handled best in a manner suited to their own needs.

CCBMS continued on 12

CCBMS continued from 11

"Didn't explain" glitch number one down, one to go. More than one person has reported saving errors; "I get the program(s) working fine, and then save it. When I come back later and reload it I get a lot of SN Errors when I run it. Looking at the lines I find them different then they were when I saved them". The problem is a bug in the DOS. Some disk controllers are affected by it more than others. If you are not having trouble and don't have RGB-DOS, you still had better abide by the following. (RGB-DOS has the I/O bug fixed.)

The problem shows itself when a save or load is made in the hi-speed mode. The programs switch the computer into the high speed mode automatically. This done to get the best speed result. (I use RGB-DOS and therefore had no trouble.) The bandaid is to put the slow speed poke in front of every disk I/O and the a hi-speed poke after the I/O. "CSFLMT" needs the speed pokes put in the following lines; 240, 1060, 3130, 6190 and 8030.0 By way of example, line 1060 before the fix, 1060 PUT #1,LOF(1)+ 1LOCATE0,0:PRINTLOF(1)+1 and 1060 after the fix, 1060 POKE65496,0:PUT #1,LOF(1)+1: POKE 65497,0:LOCATE0,0:PRINTLOF(1)+1. If your word processor has macros, just put the two pokes in them and use FIND on the line numbers.

"SLORNT" needs the same treatment in lines 240, 270, 320, 1130, 1230, 3130, 4030 (around two), 4040, 5062 thru 5065, 5068 and 20010.

AND the following is most important. Since the programs run in the hi-speed mode, the computer is still in the hi-speed mode after you hit the Break Key. If you then save the program you will experience the problem. The cure is to poke the machine into the slow speed mode before doing any save or load; use POKE65496,0. Keep in mind, this is true for any program that runs in the hi-speed mode.

Two glitches down; let's look at the bug. Mrs. Brown found entering more than twelve line items in the SOE ("SLORNT") sales order caused a SN Error. The Sales Order is designed to allow no more than twelve line items. However, there were no provisions made to care for the resulting error upon the entry of a thirteenth item. The change to line 3370 as detailed in LISTING 3 cares for the problem.

A side point, the difference between a sales order and an invoice is, a sales order records the occurrence of one sale, (could be one or more items) while an invoice is a request for payment for one or more sales orders. The invoice is given to the customer. The POS business normally has no need for an invoice since payment is made at the time of the sale.

CCBMS continued on 13

CCBMS continued from 12

Since invoices generally represent more than one sale they are logically compiled in the same module that maintains the record of unpaid purchases; the Accounts Receivable module.

Only the "twist" remains. The "help" requests are filled with the desire to add fields to both CMF and SOE. I was surprised to find so many people don't feel confident in adding a field to a file. It is straight forward, but for sure, it must be done right. The method presented here will apply to any program. A popular request has been to add a fourth customer address line to "CSFLMT.BAS" (CFM). As detailed above, it is not a good idea to add anything to CMF, but due to the number of requests, a fourth address line will be made part of the official "CSFLMT" program. The following does just that.

A real-life situation would allow for adding a field to a program's file that already contains data. We will need to not only add the field to each record of the data file, but we will have to add the ability to the program to input the data in the new field, and the ability to display it, and print it on reports. All this must be done without destroying the data already in the file.

We know a file is made of one or more records, and each record is made of one or more fields, and each field is made of one or more character. In addition, we know our BASIC allows two kinds of files, sequential and random access. We also know the records of a random access file, as opposed to a sequential file, are all the same size; have the same number of fields and characters per field. (And to maybe over clarified, the same field in any record has the same number of character spaces, but the fields in any one record can have a different number of character spaces.)

With such a fixed arrangement, after the program establishes the length of each record and the length of each field within each record, the operating system along with the program can "know" where each record is in the file, and what data is in which field.

When we add a field to an existing file this "fixed knowledge" is destroyed. So we need to write a little data conversion program that takes the data out of the existing file, record by record, and puts it in a new file that makes room for the added field. The conversion should leave us with a new file with all the old data in it, in the proper position, and a new field with no data in it. From there we just have to input the new data via program's change mode.

The customer address is in "CSFLMT". That is where the fourth address line will be added. (This how-to description is written as though I know nothing about the program.) To discover all that must be done to add a field to the data file associated with "CSFLMT", we must take a look at what is going on. (This program was first published in the January '89 issue of CoCo Clipboard.)

Doing a DIR on the disk after running "CSFLMT" we fine only one file was created, "CUS FILE.DAT", therefore it must contain all the customer data. The first chore is to fine the program lines that define the file; "CUS FILE.DAT". An easy way to do that is to load in the program and resave it as an ASCII file; after the quotation mark at the end of the program name, just add ,A to when you save it; I used "CSFLMT*A",A. Then, load in your word processor, turn word wrap OFF, load in the program to be modified, and do a search for the OPEN, FIELD and SET statement lines.

While you're in the word processor, you can make the needed mods to the appropriate lines and save the result under a different name. Then jump out of the word processor and run the "new" program. If all was done correctly you're home free. If you get an error when you try to load the new program, it could be because the program was not being saved by the word processor as a ASCII file; check the manual to find out how to do that. Also, no blank lines are allowed in a program. I've found some word processor create a blank line at the beginning of a loaded program. Just delete all blank lines before saving the new program. For sure, each line must begin with a line number and carriage returns in the middle of a program line will cause a loading error. That is why word wrap must be turned OFF.

Looking at line 1030 in "CSFLMT", we see the OPEN statement for a file named "CUS FILE". The line tells us with the "D" that it is a random access file, (one just has to call the record number to access the data in the record). The #1, means it uses the #1 disk I/O (in/out) buffer. (There are 15 possible, with 2 automatically setup on start-up). The file name is "CUS FILE", having no extension tells us the extension will be .DAT, the default extension; and 152 tells us there are 152 characters allowed in each record. This will have to be changed if we add a field.

Line 1040 contains the FIELD statement. It tells us buffer "#1" will be used, and identifies the length and name of each field. Since the first entry is 6AS CN$, we know the first field will be "6" characters long. (The total of the number of characters in all the fields in the line must be equal to or less then the total number of characters in each record specified in line 1030; 152. I think of the "AS" as "as" or "equals". The name of the first field is identified as CN$. The identification of each field in the record follows the same logic.

Finally, in line 1050 we find the SET statement. It is needed only for the save action, not read. Its job is not immediately

CCBMS continued on 15

ATTENTION SCULPTOR USERS

DO YOU NEED MORE?

Here's A Special Offer. Get A FREE CSG IMS Demo!

Flexibility

The **CSG IMS** Information Management System incorporates a full-featured procedural programming language. This incredibly robust compiled language gives you application flexibility that is unprecedented in the OS9 software arena.

Easy Forms

Need a complex screen form layout? It's easy! Just 'paint' it on the screen and **CSG IMS** will do the rest. Printed reports are just as quick and easy. Of course, you can further customize screen forms and reports with **CSG IMS**'s programming language.

Powerful Databases

CSG IMS's remarkable database structure provides excellent performance, multi-user file access, and a ruggedness that makes it almost impossible for you to lose your data. And it's huge record and file size capabilities can handle even the most complex applications with ease.

And More

But there's much more to **CSG IMS** than we can describe in this space.

Free Demo

So to let you find out for yourself, we are offering a free demo copy of **CSG IMS.** Just send us a proof of purchase for Sculptor, or any other OS9 database product, and we'll send you a demo disk and full manual AT NO CHARGE! Please by sure to specify your system type, disk size, and OS9 version with your order.

Clearbrook Software Group Inc.
USA: P.O. Box 8000-499, Sumas, WA 98295-8000
CANADA: P.O. Box 8000-499, Abbotsford, B.C. V2S 6H1
TELEPHONE: (604) 853-9118

FREE OFFER

CCBMS continued from xx

apparent. However, the disk manual says SET can be preceded by a "L" for left, or a "R" for right; "CSFLMT" uses the L. L means the data is put in the field beginning at the first character space in the field (If the input data is longer than the field, the right hand end of the data will be chopped off), an "R" causes just the opposite action and results.

After LSET we see strings not in the FIELD statement equaling strings that are the FIELD statement. I think this is the hardest requirement to understand. The strange point is, the string names setup in the FIELD statement can NOT be used to put the string value in the field itself. Sounds weird to me, but that is the way it is.

That gives us the choice of using the same string name as the name used in the FIELD statement throughout the program, and then changing it to some other name just before "setting" and saving the data; using both names in the SET statement. However the better strategy may be to use a name other than the FIELD statement name throughout the program. Then the program field name could be used directly in the FIELD and SET statements; equally the string name used in the program, "CSFLMT" does both. (Hey, van der Poel, don't laugh; remember in the beginning I said I'm not a programmer, and hindsight is 20/20.)

Now that we know what is needed to get the desired results, lets do it. First we have to determine if there is room in the record for another field. We know from looking at the OPEN statement there are already 152 characters in the record, and a record can hold up to 255. There is a lot of room left.

Next we want to determine where the new field will go in the record. If I was going to this without knowledge of the program, while I had the program in the word processor I'd make a print out of the program.

It is not a requirement, but it is a sanity keeping idea to keep like fields in a file together. To determine where in the record to put the new field we should determine which strings contain what data. To find this, I'd run the program and hit the BREAK Key as the address was being displayed. Then I'd look for strings, $ signs, in the print out around the line number that was displayed when the BREAK Key stopped the program. I'd just screen print every string I found by keying in, PRINT string name, and see what they contain. As soon as I had identified which string held the street address, I'd know where to put the fourth address string in the FIELD and SET statements. TR$ is the street string. The fourth address line will go between the customer name and street address to put in titles, etc.

CCBMS continued on 16

CCBMS continued from 15

The next trick will be to come up with a string name that has not been already used in the program. Again, if I didn't "know" the program, I'd put it back in the word processor, go into the FIND mode, enter a "$" and make a hand written list of all the "hits". In our case lets use FL$ for "F"ourth "L"ine. We want to put the new string in lines 1040 and 1050; easy done using the word processor. But before we do that, let's change the program revision number, line number 4, from 1.0.0 to 1.0.1; just a minor revision. Then zip down to the end of the program in the word processor and enter: 12000 '*** 12010 '*** Revision Record *** 12020 '*** 12030 '*** 1.0.1, Add fourth address line, FL$; and hi and low speed pokes in lines 240, 1060, 3130, 6190 and 8030.

I like to make lots of saves as I go along; I saved the program right from the word processor, naming it "CSFLMT*A.BAS". In my mind, the "A" notes that it is a ASCII save. (As a program, they take a lot longer to load then normal, so when I see the "A" I just let the disk drive do its thing while I get another cup of coffee.)

Next jump to line 1040. The place we are going to display the new address field, FL$, is just in front of TR$. Therefore we might just as well define it just in front TR$ too. We see the street string is 25 characters long, so we might as well make the fourth address line 25 characters long as well. Just insert 25AS FL$, between CS$, and 25AS.

Now that we know how many characters the new field will contain, we had better change the record definition length in line 1030 before we forget to do it. The length was 152, we added 25, so our new length is 177; well within the 255 limit. Line 1030 should now read, 1030 OPEN"D",#1,"CUS FILE", 177:RETURN

The mod to 1050 is a little more complex. We know what string we want to insert, and we know where we want to insert it, but we don't know what name to use for the second string. However, if we just look at the line we can see the second string of all the other fields is just an array string of the original string. So let's do the same thing with our new field. Just insert LSET FL$= FL$(1): between CS$(1):and LSET. (By the way, I'm using the split screen capability of Simply Better to write this. The text is in one screen and the program is in the other. I'm able to write the line in the screen that has the program and by pressing a key, copy it to the screen that has the text. What a delight.)

Okay, the "CUS FILE.DAT" file's record has been redefined. Now we have to add the new field to all the places in the program where it will get its data, and where it is displayed and printed. Also, if the original file has data in it, it won't work with the newly defined record because it was defined using the old record size. We will either have to hand input the old data in the new file or write a conversion program. I like to let the computer do the "monkey work", so we'll write a conversion program.

But first, as long as we have the program in the word processor, lets discover all the places FL$ should be inserted in the program by doing a search for TR$. I found it to be in lines 1040 and 1050, which we knew, and 3130, 3160, 4040, 4050, 6210 and 8040. We will have to check them out, but first I'm anxious to make a conversion program work.

The objective of the conversion program will be to use the old "CUS FILE.DAT" routines to call the data from the file on the disk into memory, PUT the data in the new "CUS FILE.DAT" routine and save it back to the disk. Keep in mind we won't be able to use the same name for both files, so lets call the new file "TEMP FIL.DAT". However, be advised, if something goes wrong all the data in "CUS FILE" could be lost. We had better begin the conversion by copying "CUS FILE" to something like "CUS ORIG.DAT". That way, if something goes wrong we can copy it back and try again. When we are finished and have made sure all is correct "CUS ORIG.DAT" can be KILLed.

To save a little keying effort, save the program that is in the word processor with a new name, say "CCBMS*A.BAS". Then all the lines that don't apply to "CUS FILE.DAT" disk I/O and its strings can be ripped out, deleted. Using COPY and whatever else is appropriate, convert the remaining lines from "CSFLMT" to write the conversion program right in the the word processor. Save it as "CONV.BAS". Check out what I came up with in Listing 1.

Note there will be two data files on the disk when "CONV.BAS" is finished running. "CUS ORIG.DAT" will be the old "CUS FILE .DAT" in the original format. If you need to restore the file, all you need to do is RENAME "CUS ORIG.DAT" to "CUS FILE.DAT". THis is just a safety factor. Once you are sure all is running correctly, "CUS ORIG.DAT can be deleted.

The object of our work, the new "CUS FILE .DAT" file will be on the disk too, ready to be run with "CSFLMT", but before it can be used, "CSFLMT" must be modified to allow input and display of the fourth address line. We will have to modify "SLORNT" too, to allow the display of the fourth address line. Again, to ease the strain, resave "SLORNT" in the ASCII format, load the result in your word processor and find the lines that contain TR$. I found lines 280, 1040, 1050, 3130, 6210 and 8040.

Okay, I've run the conversion program, LISTING 1, and made all the needed changes to "CSFLMT" to get the fourth address line to work. The following are the changes that were made to upgrade "CSFLMT" to revision 1.0.1: (Note, Subscribers to Clip Disk: the

CCBMS continued on 19

CCMDisk Service

PRESENTS

ClipDisk

Hate to type in those long listings Lynn? Save time and money - subscribe to ClipDisk today!

FULL YEAR (6 issues) Only $49.95 !

Now all the BASIC, C, BASIC09, Assembly Language and OS9 programs you see in *CoCo Clipboard Magazine* are available on disk. With *ClipDisk* you'll save time and be more productive computing with your Color Computer.

A subscription to *ClipDisk* is $49.95 for 6 issues. Single issues are available as well for just $9.95. Of course in order to get the full benefit of *ClipDisk* you'll need the corresponding issue of *CoCo Clipboard Magazine*. All *ClipDisks* are mailed first class.

So don't delay, get your subscription in today!

I've enclosed (check one) ( ) 1 year of ClipDisk $49.95
( ) single copy 9.95

Specify month desired _____

Canadian customers can also obtain ClipDisk for $49.95 when paid in U.S. Funds.

Name: _____

Address: _____

_____

City: _____

State/Prov: _____

Zip/Postal Code _____

6 - 2 - 1

$83.25

SAVE 15% FROM LIST PRICE

Just in time for holiday giving it's 6-2-1! "6-2-1" is our new program to save your money! 6 back issues of *Clipboard*, a 2 year subscription to *Clipboard* and 1 year of *ClipDisk*. Normally $97.95 now you can save 15% only $83.25! Just write in "6-2-1" on the order form on page 18.

MC-Visa-Discover accepted.

Please charge my purchase to my (circle) VISA / MasterCard / Discover Card.

MasterCard Interbank Number | | | | |

Card # _____

Exp. Date _____

Signature _____

PLEASE NOTE: All charge card orders are handled courtesy of On-Line Computer Services. Your charge card billing will show On-Line Computers and NOT CoCo Clipboard Magazine.

CoCo Clipboard Magazine

3742 U.S. 20, Box 3, Fredonia, NY 14063

Alpha Software Technologies
Discount Coupon

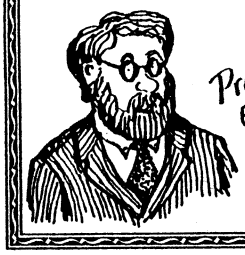
This coupon entitles the bearer to a 10 percent discount on ANY product from Alpha Software Technologies.

WITH COUPON & YOUR OLD ADOS3

$15.00 OFF

MJK & MJK3 DOS

COCO CONECTION
5003 B ST.
PHILA, PA 19120-3929
215-457-1809



E-Z FRIENDLY

Save 10% on C.S.G.
with this coupon

Clearbrook Software Group

CSG IMS

Information Management System

CSG IMS is a full featured relational database manager for the Color Computer and OS9.

- Interactive access to databases for quick queries.
- Includes a structured compiled language supporting program modules with full parameter passing and recursion.
- User defined screen and report formats.
- Record, index, and file sizes are virtually unlimited.
- Text, BCD floating point (14 digits), short and long integer, and date data types.

CSG IMS for CoCo2/3 OS9 L1/2 (single user)	$169.95
CSG IMS for OS9 L2 or 68000(multi user)	$495.00
CSG IMS demo with manual	$30.00
OTHER CSG PRODUCTS:	
ERINA Symbolic User-mode Debugger for OS9	$69.00
SERINA System-mode Debugger for OS9 Level 2	$139.00
MSF MS-DOS File Manager for CoCo3/OS9 L2	$45.00
MSF with SDISK3	$65.00

Shipping — N. America: $5, Overseas: $10

Clearbrook Software Group, Inc.
U.S.: P.O. Box 8000-499, Sumas, WA 98295
CANADA: P.O. Box 8000-499, Abbotsford, B.C. V2S 6H1
Phone: (604)853-9118

OS9 is a trademark of Microware Systems Corp.
MS-DOS is a trademark of Microsoft Corp.

MasterCard VISA

Don't Miss Out! save 10% on

GAME POINT

Game Software Only

P.O. Box 6907, Burbank, CA 91510-6907
(818) 566-3571 • BBS: (818) 772-8890
Toll Free: 800 877-2232 ext. 139

SAVE NOW 10% on

DORSETT
Educational Systems, Inc.
Box 1226, Norman, OK 73070

Dealer Inquiries Welcome

SAVE $15
WITH THIS COUPON

ON A SCSI INTERFACE
WITH R.T.C.

OR $15 ON A 2 CHANNEL
DUAL COMM BOARD

FROM:

KEN-TON ELECTRONICS
187 GREEN ACRES RD.
TONAWANDA, NY 14150

<<< GIMMESOFT >>>

This certificate entitles

to ——— 10% (TEN) ——— off

their next purchase.

Non-transferable and not valid with any other sale prices or certificates.
This offer expires: 6-30-89

CoCo Clipboard

RGB COMPUTER SYSTEMS

294 STILLWELL AVE.
KENMORE, NY 14217

S A V E

$3.00

with this coupon on your next order from RGB Computer Systems

B U L L E T I N

Call for details on our new product which allows you to boot OS9 directly from the Hard Disk without floppy or special EPROM

CCBMS continued from 16

copy of "CSFLMT" on your disk has all changes made to it. However, don't forget, if you have data in "CUS FILE.DAT", you will have to either KILL the file and re-enter the data by hand, or run "CONV.BAS".) Note, for those that do not subscribe to Clip Disk: (a) There are periods at the ends of sentences. Don't type them into the program. (b) Those lines not detailed in the text are in LISTING 2.

Change LOCATE value in lines; 2070, 0,5 to 0,7; 2080, 0,7 to 0,9; 2090, 0,5 to 0,7; 2100, 50,7 to 50,9; 2110, 62,7 to 62,9 and in 2120, 0,9 to 0,11.

Retype (copy) line 2070 to line 2075. Change line 2070, (See LISTING 2). Add line 3131. Change 3130.

Change LOCATE value in lines; 3160, 0,5 to 0,7; 3170, 0,7 to 0,9; 3180, 0,9 to 0,11 and delete the semicolon at the end; 3190, 57,5 to 57,7; 3200, 57,7 to 57,9 and in 3210, 68,7 to 68,9.

Add 3155 and 4045, and change 4050. Change Y1= and LOCATE in lines 4060, 4070 and 4080, 7 to 9 and 0,7 to 0,9. Change Y1= and LOCATE in line 4090, 9 to 11 and 0,9 to 0,11. Change Y1= in line 4110, 5 to 7. Change Y1= in lines 4120 and 4130, 7 to 9. Change C< in line 6110, 15 to 13. Change lines 6140, 6150, 6200 and 6210. Change C= in line 6230, 14 to 13.

The needed mods to "SLORNT" are a lot easier. Refer to LISTING 3 for detail of mods not shown here in the text Change 1.0.0 in line 3 to 1.0.1; in line 1030, 152 to 177. Insert between CS$, and 25AS in line 1040, 25AS FL$. Insert between CS$(1): and LSET TR$, in line 1050, LSET FL$=FL$(1):. Change lines 280 and 3370. Add 12000 thru 12030. That's all there is to to the twist.

Next time we'll have an integrated Accounts Receivable module "spine" that will do invoicing, statements, aging, etc. Don't forget, write or phone with your suggestions, questions and/or thoughts; and, "Thank you." to all that have expressed an interest in this project.

CCBMS Listing 1

```
0 '*** Run in 80 col display. Pr
ess any key each time scroll sto
ps
10 PCLEAR1:GOTO9030
1000 COPY"CUS FILE.DAT"TO"TEMP O
LD.DAT"'*** preserve "CUS FILE"
1005 '*** Will use buffer #1 and
 #2
1030 OPEN"D",#1,"TEMP OLD",152
1031 OPEN"D",#2,"TEMP FIL",177
1035 X=X+1
1040 FIELD #1,6AS CN$,30AS CS$,2
5AS TR$,15AS CT$,2AS TT$,10AS ZP
$,45AS CM$,13AS PH$,3ASTY$,3AS C
L$
1042 GET #1,X
1050 CN$(1)=CN$:CS$(1)=CS$:TR$(1
)=TR$:CT$(1)=CT$:TT$(1)=TT$:ZP$(
1)=ZP$:CM$(1)=CM$:PH$(1)=PH$:TY$
(1)=TY$:CL$(1)=CL$'*** line 1050
 changed to this config
1051 FIELD #2,6AS CN$,30AS CS$,2
5AS FL$,25AS TR$,15AS CT$,2AS TT
$,10AS ZP$,45AS CM$,13AS PH$,3AS
TY$,3AS CL$
1052 LSET CN$=CN$(1):LSET CS$=CS
$(1):LSET FL$=FL$(1):LSET TR$=TR
$(1):LSET CT$=CT$(1):LSET TT$=TT
$(1):LSET ZP$=ZP$(1):LSET CM$=CM
$(1):LSET PH$=PH$(1):LSET TY$=TY
$(1):LSET CL$=CL$(1)
1053 '*** all has been defined,
and now we just need to GET the
data from the old file and PUT i
t in the new file
1065 PRINT"CN$";CN$"#":PRINT"CS$
";CS$"#":PRINT"FL$";FL$"#":PRINT
"TR$"TR$"#":PRINT"CT$";CT$"#":PR
INT"TT$";TT$"#":PRINT"ZP$";ZP$"#
":PRINT"CM$";CM$"#":PRINT"PH$";P
H$"#":PRINT"TY$";TY$"#":PRINT"CL
$";CL$"#":PRINT"     - - - - - -
- - -"
1070 PUT #2,X
2000 '***
2010 '*** display new file to be
 sure it is correct
2020 '***
2030 '*** FIELD #2,6AS CN$,30AS
CS$,25AS FL$,25AS TR$,15AS CT$,2
AS TT$,10AS ZP$,45AS CM$,13AS PH
$,3ASTY$,3AS CL$
2040 GET #2,X
2050 PRINT"CN$";CN$"#":PRINT"CS$
";CS$"#":PRINT"FL$";FL$"#":PRINT
"TR$"TR$"#":PRINT"CT$";CT$"#":PR
INT"TT$";TT$"#":PRINT"ZP$";ZP$"#
":PRINT"CM$";CM$"#":PRINT"PH$";P
H$"#":PRINT"TY$";TY$"#":PRINT"CL
$";CL$"#"
2960 EXEC44539:X=X+1::IF X<LOF(1
)+1 THEN1040
3000 CLOSE:KILL"TEMP OLD.DAT":RE
NAME"CUS FILE.DAT"TO"CUS ORIG.DA
T":RENAME"TEMP FIL.DAT"TO"CUS FI
LE.DAT":END
9030 CLEAR9999:FILES2,500:GOTO10
00
11111 CLOSE:KILL"TEMP OLD.DAT":K
ILL"TEMP FIL.DAT":STOP
```

CCBMS Listing 2

```
2070 LOCATE0,5:PRINT"Title . . .
 . "STRING$(25,46)
2075 LOCATE0,7:PRINT"Street. . .
 . "STRING$(25,46)
3130 GOSUB1040:POKE65496,0:GET #
1,X:POKE65497,0:LOCATE0,6:PRINT:
GOSUB2070:IF FG(1)=1 THEN CS$(1)
=CS$:CN$(1)=CN$:FL$(1)=FL$:TR$(1
)=TR$:CT$(1)=CT$:TT$(1)=TT$:ZP$(
1)=ZP$:CM$(1)=CM$:PH$(1)=PH$:TY$
(1)=TY$:CL$(1)=CL$:GOTO3140 ELSE
 LOCATE0,13:ATTR0,0,U
3131 PRINT:ATTR0,0:PRINT:PRINT:G
OTO 3140
3155 LOCATE0,5:PRINTSTRING$(14,3
2)FL$;'*** title inquire display
4045 LOCATE34,22:PRINT"Person's
Name, Building, Etc.";:F$=FL$(1)
:X1=14:X2=14:Y1=5:LN=0:LN(1)=25:
X1=X1+LEN(FL$(1)):GOSUB50:LOCATE
0,5:PRINTSTRING$(13,32);:IF F$>"
" THEN FL$(1)=F$'*** title input
4050 LOCATE34,22:PRINT"Customer
Street. "TS$;:F$=TR$(1):X1=14:X2
=14:Y1=7:LN=0:LN(1)=25:X1=X1+LEN
(TR$(1)):GOSUB50:LOCATE0,7:PRINT
STRING$(13,32);:IF F$>"" THEN TR
$(1)=F$'*** Street input
6140 PRINT#P,"CUSTOMER NAME";STR
ING$(20,32);"TITLE";STRING$(35,3
2);"LINE #"
6150 PRINT#P,"STREET";STRING$(27
,32);"PHONE #              TYPE
   CLASS      IDENT#"
6200 PRINT#P,CS$"   "FL$;STRING$
(15,32);:PRINT#P,USING"######";X
6210 PRINT#P,TR$"        ("PH$"
      ";:PRINT#P,USING"% %";TY$;
:PRINT#P,"      ";:PRINT#P,USING
"% %";CL$;:PRINT#P,"       ";:PR
INT#P,CN$
12000 '***
12010 '*** Revision Record ***
12020 '***
12030 '*** 1.0.1, Add fourth add
ress line, FL$ and slow & hi-spe
ed pokes in PUT/GET lines 240, 1
060, 3130, 6190 & 8030.
```

CCBMS Listing 3

```
280 LOCATE1,2:PRINTCS$;:LOCATE0,
3:PRINT" "FL$:LOCATE1,4:PRINTTR$
:LOCATE1,5:PRINTCT$", "TT$"  "ZP
$:LOCATE1,6:PRINTCN$"   >>> "CM$
" <<<":RETURN
3370 SOUND225,1:LOCATE26,22:PRIN
T"Correct ?, (N/Y)       ";:X1=4
3:Y1=22:GOSUB200:SOUND224,1:ATTR
0,0:IF I$<>"Y" AND I$<>"y" THEN
FG(2)=1:GOTO3250 ELSE IF FG(1)=1
 THEN5040 ELSE C=C+1:IF C>12 THE
N FG(2)=1:GOTO3380 ELSE Y2=Y2+1:
Y1=Y2
3371 SOUND225,1:LOCATE4,22:PRINT
TS$TS$"Enter Another Line ?,(N/Y
) ";:X1=51:Y1=22:GOSUB200:SOUND2
24,1:ATTR0,0:IF I$<>"Y" AND I$<>
"y" THEN FG(2)=1:GOTO3380 ELSE33
00
12000 '***
12010 '*** Revision Record ***
12020 '***
12030 '*** 1.0.1, Add fourth add
ress line, FL$ and slow & hi-spe
ed pokes in lines 240, 270, 320,
 1130, 1230, 3130, 4030, 4040, 5
062 thru 5065, 5068 and 20010.
```

Product Reviews

Review Crew

Hard Drive Interface
Ken-Ton Electronics
187 Green Acres Road
Tonawanda, NY 14150
(716) 837-9168

Hdwr. Type : SCSI Hard Drive Interface with Optional System Clock
Requires : Any CoCo and SCSI/SASI Hard Drive
Price : $89.00; with optional Clock $119.00; $4.00 S&H

By Jim DeStafeno

So what is there to a hardware review? The part either works or it doesn't. The quality is either good or bad. The price is high or in the ball park. Just report what you see and get on with your thing.

That may be true most of the time, but its not in this case. Whether you know it or not, you are right in the middle of pitched technical battle. The fur is flying and its all about SCSI vs Non-SCSI device buses, of which SASI seems to be the leader. But before discussing that, let's look at this Ken-Ton offering.

I wanted to check out the SCSI bus, so number one, I was looking for a SCSI interface and second, the Ken-Ton ad says it meets MIL-Specifications. MIL means military, which is about the best quality that can be had by us normal people. I'm not an electronics quality control expert, but a few years ago I worked for a "MIL-Spec" electronic manufacturer. From what can be seen with the black plastic cover off, the Ken-Ton interface is true to its ad; gold contacts, wave soldered, cleaned PC board, wide well spaced traces, some socketed chips and what might be mistaken for an error, an empty chip socket.

I was a bit surprised to find no documentation in the package. I guess Joe Scinta, President of Ken-Ton Electronics, figures a device with only two plugs and each a different size can't cause the user much trouble. It didn't. One end plugs into the bus on the right side of the computer, or on a "Y" cable; which means its chips are of the low power draw type; or into a Multi-Pac. However, getting a Multi-Pac to use with the Ken-Ton interface would really be a wasted expense. My -3 has 512K with a "Y" cable that has a multi-slot disk controller and the Ken-Ton hard drive interface on it. To date I've had no trouble. The point is, the low power draw components cost more, but not as much as a Mul-Pac. When you compare prices, don't forget to add in the cost of a Multi-PAC for those interfaces that must have one.

The other end of the interface plugs into a ribbon cable that goes to a hard drive. For those that are not quite sure what a hard drive interface does, as I understand it, the computer's operating system; in the case of the CoCo, one of the modified RS-DOS (BASIC) systems like RGB-DOS or Hyper-I/O, or OS-9 sends the HD act request to the interface. The interface prepares the signal and gets it on the right wires (cable or bus) going from the interface to the controller board of the HD. In the case of a SCSI compatible interface, it gives the signal a boost in power and handles the "handshaking" as well, which reduces the delay before transmission begins. That act results in a faster overall transfer rate; as opposed to making the operating system do it, as non-SCSI standards do.

There are several standard hard drive "buses". The most popular bus for the size (read price) hard drive used with the CoCo are SASI and SCSI. SASI is the oldest, SCSI is the newest. Not all of the CoCo gurus have imbraced SCSI, (see below).

The hard drive controller generally comes with the hard drive itself, but that is not necessary; each can be gotten separately. It is really the controller that supports the bus standard; SCSI, SASI or whatever. The controller changes the bus code received to what the particular hard drive needs to do to comply with the request.

As to the operation of the Ken-Ton hard drive interface, it has done its thing in my machine without flaw since day one. In addition, its components must be high quality because in my system its not only inside its own case, but it is inside the hot CoCo case as well. So as far as the interface itself, I can only conclude that it is a very good one. As an added incentive to go with Ken-Ton, the president is an avid CoCo user, so he should be on top of any new developments.

Reviews continued on 22

Reviews continued from 21

The optional Ken-Ton system clock seems as good as the interface. The OS-9 implementation, software driver, does the job very well; while the BASIC implementation and the lack of documentation leaves a bit to be desired for the beginner.

That is, along with the hardware, your dollar gets you a disk with a OS-9 clock configuration program. Once you have installed the software in your OS-9 system the Ken-Ton clock tells the OS-9 system what time it is upon start-up and keeps the system clock accurate through out the run.

The best way to get time into a BASIC program from an on board clock is via an interrupt routine in the operating system. Unfortunately RS-DOS (BASIC) has no provisions for a system clock. The program shipped by Ken-Ton to be used with BASIC is really a subroutine. When called, via PEEKs and POKEs, the routine gets the time from the clock and puts it in a STRING variable. The knowledgeable programmer can use the value any way he wishes.

The clock can be set to 24 hour time or 12 hour time with AM and PM. It also has month and day names and automatically compensates for the different number of days in each month and leap year.

Now, what about the empty socket? Well, that has to do with the SCSI bus. So lets get into it. In comparison, SASI does little more for the user than connect two devices. In our case, the computer and hard drive. Much like the serial port, all devices connected to the a SASI bus "see" the data. Of course this would cause mixed result if more than two devices, computer and hard drive, were on the bus at the same time.

Not bad with two devices, but what does one do if another device is wanted, use an electronic switcher box? The standard SCSI bus allows a total of eight devices on the bus without cross talk. It thereby cares for the multiple device situation. An equally important point concerning the advantage of the SCSI bus, at its base level the bus is a read/write conduit. The computer needs only to send the code that tells the device what data it wants to read or write. The SCSI standard limits the action to just that. From that point the SCSI standard leaves it up to the device and its controller to comply.

Using such a scheme the operating system need only know what it wants done. On the other hand, in general the SASI bus relies on the operating system to tell the device how to do its job.

The SCSI bus makes life much easier for the operating system designer as well as the system user. Just buy the CSI compatible device of choice and plug it into the bus. That way the device manufactures can do whatever they want to control their own device as long as it responds to the SCSI codes properly. Very neat; if you want to change from a 10Mb HD to a 170Mb HD, just unplug to old one and plug in the new, even if the manufacturers, etc., are different.

The point to remember in our case, if you have a SCSI bus system and your device goes South you only need to be sure the new device is SCSI compatible. If you have a SASI system you have to be sure it is compatible with your operating system code and interface; very restricting. Don't let me mislead you into thinking SCSI is only for hard drives, its for anything that normally would be connected to a computer. All of the peripheral companies are switching to the SCSI controllers; tape drives, some floppies, plotter boards, etc. The up and coming laser disks will be SCSI compatible. The trade journals are reporting that by the end of the year or the beginning of next, even the lower priced IBM PCs will use the SCSI bus.

Also, don't let anyone mislead you into thinking a SCSI interface won't work on a SASI bus. It will, but of course the bus will be good for only a computer and one device. Another neat thing about the SCSI bus is that everything on the bus can communicate with everything else on the bus without any data getting mixed. Of course this requires proper software, but try this on for size. Let's suppose you have two or three CoCos on a SCSI bus. Only one of them needs to have a clock for all of them to know what time it is.

To the best of my knowledge, RGB-DOS is the only SCSI compatible BASIC DOS available for the CoCo. Of course since SCSI is backward compatible, it will work with SASI devices, but all SCSI advantages are lost. I talked with Burke and Burke, and Owl-Ware about converting their DOSes to SCSI. Owl-Ware said they are working on it now, Burke and Burke said they doesn't really believe SCSI has any value due to the lack of record locking (see below).

To check out this SCSI bus I have setup a CoCo-3, and to add spice, a -2. Each computer has a Ken-Ton interface plugged into the bus on the side of the computer. As I said before, the one on the -3 is on a "Y" cable with a floppy drive controller, but the -2 only has the interface. Each interface has a cable going the HD.

My RGB-DOS is in an EPROM. The EPROM for the -3 is plugged into the floppy controller, but the RGB-DOS EPROM for the -2 is plugged into what was the empty socket of its interface. (The interface on the -3 is still sporting an empty socket because the EPROM is in the controller.) I was told both the -2 and -3 could get there programs from the HD once the HD had been loaded from a floppy.

I'm here to tell you it not only can, but does. In fact, both computers can load the same, or different, program(s) at the same time with no problem. This leads to some interesting possibilities. Though the following has not been tested, the thoery says it could work.

How about a business system with seven

Reviews continued on 23

Reviews continued from 22

CoCos and a HD on a SCSI bus? One positive result would be no computer slow down because each computer would be working from its own CPU and memory. All that would be needed for such a BASIC system is the CoCos, a SCSI interface plugged into each, a HD to retain the programs/data and lots of ribbon cable, plus a floppy or (tape recorder ??) and of course RGB-DOS.

Each of the CoCos in the system must have its own DOS, but not a floppy controller. If there is no need for a floppy controller on every CoCo, there is little sense to have one just to hold a DOS chip. Since the interface is needed anyway, Ken-Ton has made provision to plug the DOS chip into it.

Want to carry the seven CoCo system to more users? Put OS-9, 512K memory, a Ken-Ton Dual Comm Board (their other major CoCo product) and two terminals on each of the seven CoCos. That would make a 14 user system. Do you have any idea what a commercial Unix/14 user 3.5 Mb RAM system costs?

So why all the controversy about SCSI in the CoCo community? Well there seems to be three major points. As faulty as the thinking may be, one fault sited is the SCSI bus data transfer rate is slower then the SASI rate. Not true, they are the same. But that is not really the point. Even if true, the slowest CoCo SCSI or SASI system transfer rate is 10 times faster than the CoCo can deal with. So in practical terms, when using a CoCo, the point is pointless.

The other two points are more meaningful. One is the SCSI standard results in a bus (cable) length of no more than 6 meters (20 feet). Can't you see the gurus bent over in laughter picturing seven CoCos in a twenty foot around circle. But don't let them kid you. There are standard power boosters that just plug in. With the OS-9/Comm Board system, the terminals can be a 100 feet or more apart. There are also boosters available for that type cable. So the short bus complaint can be cared for easily.

The third point has to do with record locking; SCSI doesn't have it, nor does SASI or RS-BASIC. OS-9 does, but its at the CPU level, which is really the only practical way to do it. This could be a very real problem in a complex business system. However, I have a very simple software fix in my head. Preliminary tests show it works, but I want to do a real wring out test before committing myself. Hopefully by the next issue we can have an in depth look at why record locking is needed in complex systems and the results of the tests.

In conclusion, the Ken-Ton Hard Drive Interface appears to be a 1st class piece of hardware that not only does its job today without flaw, but is already setup to allow you to take advantage of future industry developments. The optional clock documentation and BASIC implementation is a bit on the slim side, but Ken-Ton has done about as much as a clock supplier can do in this reguard. It is the responsibility of the operating system designer to allow the use of a clock, much as OS-9 does. RS-BASIC does not. The supplied OS-9 and BASIC software works just fine. It is easy to recommend Mil-Spec type hardware that not only works flawlessly, but is designed to make use of tomorrow's technology.

Rupert Rythym, Space Intruders and Donut Dilemma
Game Point Software
P.O. Box 6907
Burbank, CA 91510-6907
(818) 566-3571

Pgm Type: Entertainment/Game Software
Requires: CoCo 3, Disk, RGB/TV/Composite Monitor (RGB preferred)
Price : $24.95 ea.

Reviewed by Bill Laurence

One thing I have always hated about a lot of game software with graphics, written for the CoCo has been the really poor use of color and speed. Most of what I've seen has flat screens, jerk a long animation and no depth perception of any kind. Sound has been lacking as well save for the few grunts and explosions which are almost an after thought. For those of you who have experienced many of the arcade style games that run on an Atari 520 or 1040 ST will know what I mean.

Reviews continued on 24

Reviews continued from 23

The Atari games use subtle shadings, contrasts and animation speeds to create smooth running games that look just like dedicated arcade games. Why aren't CoCo games just like these? Are there hardware problems or restrictions? The answer is no.

Game Point Software has now produced three excellent arcade type games and with the slight exception of Donut Dilemma, all demonstrate an incredibly sophisticated amount of graphics, sound and entertainment value. Donut Dilemma is perhaps the only "throwback" in this group. It has all the speed of animation, but it's a CoCo 1/2 graphics type of game, it was Nick Marrentes first effort. My kids loved it and it's easy to play, don't let the lower graphics detail steer you away from this one.

However, after you've tucked the kido's in for the night, pull out Rupert Rythym and Space Intruders. And if you haven't been able to find a good reason to pick up that color RGB monitor now you have two. The opening graphics screen in the Rupert program is absolutely incredible. Everything that I've ever seen in graphics on any other machine (ok except for my neighbors Cray super computer) is matched by these programs. Speed, color, depth of field perception and in Rupert's case sound. And to make things more interesting these games are great playing games as well - no whimpy stuff here. You need your thinking caps and all your motor skills to win either Rupert or Space Intruders. A synopsis of what takes place follows.

Rupert Rythym This game relies on your skill in solving a sound puzzle. The premise is that your musical score has been stolen by Hardrock Harry and hidden piecemeal in his record factory. Armed only with your ability to run, jump and throw firecrackers you must successfully negotiate 16 different rooms of the music factory to recover your work. You also have a limited amount of "pep" pills to take to make yourself invisible. Naturally there are the guards to overcome and to defeat as you pick up the music in each level. They range from a robot to a floating eyeball rover that really is pretty slick. You must use pedestal type elevators to jump from one section to another and the percussion sounds used are simply terrific.

Once you have successfully recovered all 16 portions of music you must rearrange them into their proper order. The game will let you know if the order is correct and if it is you'll be treated to Ruperts' on stage concert. Pete Ellison tells me it's a real wow-ser in animation and sound.

Space Intruders How many times can we defeat the aliens? This games goes back many years to the original Space Invaders arcade game but has been finally rendered in a form for the CoCo which is outstanding. You get three rocket launchers with their recoiling cannon, five different waves of nasties to blast away and of course the usual "mother ships" which will count for bonus points, especially if you've read the directions and saved your 25th shot. As you complete each wave of attackers the next is positioned a little lower and seem to be just a little faster. I learned how to duck and weave under behind the barriers and how to get off two shots rather quickly. After some embarrassing tries I got within two aliens from completing the game! Now many of you are probably wondering why *Clipboard* is even doing a game review, especially after all the speeches concerning "no nonsense" programs, tutorials and such. These two programs are finally worth your hard earned dollars. As we all know, all work and no play can make for some pretty dull hours at the computer. Playing a game like Space Intruders or Rupert helps to unwind the mind and the hands. They're also a nice way to introduce your kids to the CoCo. Kids like computers anyway (must be the result of altered genes passed on by parents who watched too much Howdy, Soupy and Roy Rogers) and so why not teach them how to run your CoCo. I found out my oldest had pulled out Rupert, read the directions to her younger brother and between the two of them were happily playing away while I was at work. The kids now know how to load and execute a disk based program and what to do to shut the machine down.

All three games from Game Point Software come highly recommended.

Leonardo's Paintbox
E.Z. Friendly Software
118 Corlies Avenue
Poughkeepsie, NY 12601

Pgm. Type: Graphics Utility
Requires : CoCo 3, Disk Drive, Joystick
Price : $29.95

Reviewed by Boisy G. Pitre

Leonardo's Paintbox is a user-friendly graphics utility written in BASIC, and requires a CoCo 3 with one disk drive and a joystick (preferably a self-centering, two button joystick). It's main attraction is that you "command the power of your CoCo 3" to do the final drawing while you sit back and watch. All you have to do is draw the graphic on the screen once and Leonardo's Paintbox saves it in a neat BASIC program. After you are finished, you can RUN the program from disk (under the name you saved it) and your drawings will magically reappear!

Although some may not understand why this program is so helpful, it is because Leonardo's Paintbox caters to the "non-programmer" type person. It's rather tired and heartless to plan out every HCIRCLE, HSET, and HPAINT, running the program 500 times as your picture progresses. Leonardo's Paintbox takes the frustration out of de-

Reviews continued on 25

Reviews continued from 24

veloping hi-res graphic programs in BASIC by doing all the work for you.

Leonardo's Paintbox is divided into five parts:

1- Starting the Graphic
2- Drawing Mode
3- Painting Mode
4- Adding Dots of Color
5- Saving the Graphic

Part 1 allows you to center your joystick and choose a color to draw with. You have 16 colors to choose from on a 320x192 hi-res screen. Each palette can be modified to hold a different color if desired. The joystick (or keyboard) is used to move the cursor about the screen. Once you have selected a color and position to draw, you are ready for part 2.

Part 2 actually lets you draw the graphic. Such features as RE-DRAW, UNDO, and CLEAR SCREEN are available. Pressing the red button on the joystick sets lines on the screen. Once you have drawn the basic outline of your picture, you are ready for part 3.

Part 3 lets you paint what you drew in part 2. The 16 color palette is again available for you to paint with. You must use the joystick to position the cursor and apply the paint to the area of the screen.

Part 4 lets you add dots to your graphic. This mode is helpful to give your picture a shading effect.

Part 5 saves your graphic to disk. Leonardo's Paintbox takes the picture you drew and writes it to disk as a BASIC program. Saving a graphic can take quite a while, depending on the intricacy of your picture.

To see your new graphic file, RUN it from disk and see it re-drawn to the screen! It's fun to watch how fast and lively your drawings can be replicated! There is even a demo program that shows you how Leonardo's Paintbox works. The demo is very impressive and shows how much can be done with this great program.

Overall, this program is superb for those who don't wish to write their own BASIC code. The 6 page documentation booklet is well written and explains every aspect of the program. It would be nice if the high-resolution joystick interface was supported. Also, the cursor is rather faint on the drawing and dot screens. You may have to strain your eyes to see it.

The performance of the program itself is good; however, I did find a bug. One particular graphic that I saved to disk was re-draw with a few pixels missing, causing the paint to spill over into the rest of the picture.

Leonardo's Paintbox should be in the library of any artistic programmer. If your looking for an good way to draw graphics in BASIC and don't want to spend time mapping pixels on the 320x192 coordinate sheet, Leonardo's Paintbox offers the easiest, friendliest, and best way to "command the power!"

RASCAN Video Digitizer
Game Point Software
P.O. Box 6907
Burbank, CA 91510-6907
(815) 566-3571
(800) 877-2232 ext. 139

Pgm. Type : Video Digitizer Hardware & Software
Requires : 512K CoCo 3 for all features
Price : $159.95
Options : Color Filter Kit $12.00

Reviewed by Bill Laurence

(Editor's Note - This review is based on phone interviews with Pete Ellison conducted in early April prior to the product release. Additional information has been provided by Game Point. The first RASCAN units were sold in mid April.)

Three years ago I was writing a column on small computer for our local paper here in the Dunkirk-Fredonia area. One of my visits was to a local high school which used a variety of computers . Digitials, Commodore 64's and a Commodore Amiga we all in use. What made the visit interesting was a video digitizer they were running to the Amiga that had incredible resolution, color and editing. The CoCo had a video digitizer and I had seen it run, but nothing like the system I saw on the Amiga. But that was then, and this is now, and now is called RASCAN!

Game Point Software has developed a complete video digitizer system - hardware AND software that takes complete advantage of the power of a 512k CoCo 3. RASCAN provides full black and white and COLOR imaging to the CoCo including what Pete Ellison tells me is the "4096" mode. Above all else the "4096" mode puts the CoCo 3 into the main stream of video digitizing. The "4096" mode provides up to 4096 colors on an RGB monitor when using a video camera as the input source. The procedure is similar to that of the Amiga system I saw in the school.

All that is needed is RASCAN, a color video camera, three colored filters and your subject matter. Now for most of us a color video camera is not something we just go out and buy just to use with our CoCo's. However many video stores and many renters such as Taylor rental will rent a video camera for as little as $25.00 a day. With a good supply of blank floppies you can do a lot of work in just one day. However we will assume that you have access to a color video camera. Select your subject matter place a colored filter in front of the lens and then follow the instructions with RASCAN. You'll

Reviews continued on 27

Color Computer Software from Cer-Comp Ltd.

Window Master V2.2

The hottest new program available for the Color Computer III! Now you can have Windows, Icons, Buttons, Pull-Down Menus, Edit Fields and Mouse Functions built into your Basic or Machine Language Programs easily and quickly, without the need for OS9.

It supports up to 31 Windows on the display, multiple fonts in 54 possible sizes and styles, Enhanced Basic Editing and much more. It adds over 50 Commands and Functions to Basic to fully support the Point & Click Window System. In fact it has so many features it would take several pages to to describe them all.

It is completely compatible with existing Basic programs and takes absolutely no memory away from Basic. It contains a built in Ram Disk which is completely transparent to Basic (512k version) for enhanced operation.

It requires 1 Disk Drive, R.S. Hi-Res Interface & Joystick or Mouse. Includes both the 128k & 512k versions for only $69.95

Window-Ware

Window Writer - A Point & Click Word Processor, features both Mouse & Keyboard type editing, proportional printer support, powerful formatting capability, works with any printer. Requires Window Master & 512k- $59.95

Window Basic Compiler - A Basic Compiler similar to CBASIC only it compiles all the Window Basic statements to create super fast M.L. programs & Desk Accessory programs for Window Master $99.00

Window EDT/ASM - A full featured Editor/Assembler and Debugger for the Window Master System $49.95

Font/Icon Editors - A utility disk with the Font & Icon Editors so you can edit or create your own, includes Basic & M.L. versions $19.95

Advanced Programmers Guide - A manual for the Advanced Basic & M.L. Programmer with information on interfacing to Window Masters complete system including System Calls, Memory Map, Interrupt handling & Extended Memory access. $24.95

512K RAM UPGRADE

Give your COCO 3 all the power it deserves with this easy to install (no soldering/plug in) 100% Tandy compatible 512K memory upgrade. Completely assembled and tested. Includes Ramdisk & Memory Test software described below. $159.95, Upgrade + Window Master $199

512K RAMDISK & TESTER

RAMDISK is an ALL Machine Language program that will give you 2 ULTRA High Speed Ram Disks in you CoCo-3. It does not need or require the OS-9 operating system. It works with R.S. DOS V1.0 or V1.1 and it is completely compatible with Enhanced Color Disk Basic! Plus it allows your CoCo-3 to run at double speed all the time even for floppy disk access!!! It will not disappear when you press reset like some other ramdisk programs. The MEMORY tester is a fast ML program to test the 512K ram. It performs several bit tests as well as an address test so you know that your 512K of memory is working perfectly.

Requires 512K & Disk $19.95

CBASIC Editor/Compiler

The ULTIMATE Color Computer BASIC COMPILER!!!

If you want to write fast efficient machine language programs and you don't want to spend the next few years trying to learn how to write them in Assembly language or with a cheap compiler, then CBASIC is the answer!!!

CBASIC is the only fully integrated Basic Compiler and Program Editing System available for the Color Computer. It will allow you to take full advantage of all the capabilities available in your CoCo without having to spend years trying to learn assembly language programming. CBASIC allows you to create, edit and convert programs from a language you are already familiar with Enhanced Disk Color Basic, into fast efficient machine language programs easily and quickly.

CBASIC supports all the enhanced hardware available in the CoCo 2 & 3, including Hi-Res Graphics, & Screen displays, Extended Memory and Interrupts. We even added advanced commands not available in Basic to give you a level of control only available to very advanced Machine Language Programmers. Plus we made it exceptionally easy to use, not like some other compilers. CBASIC is the friendliest and easiest compiler available for the Color Computer.

CBASIC is a powerful tool for the Beginner as well as the Advanced Basic or Machine Language programmer. CBASIC features well over 150 Compiled Basic Commands and Functions that fully support Disk Sequential and Direct access files, Tape, Printer and Screen I/O. It supports ALL the High and Low Resolution Graphics, Sound, Play and String Operations available in Enhanced Color Basic, including Graphics H/GET, H/Put, H/Play and H/DRAW, all with 99.9% syntax compatibility.

CBASIC makes full use of the powerful and flexible GIMI chip in the Color Computer 3. It will fully utilize the 128K of RAM available and install 2 Ultra Fast Ramdisks if 512K is available, for program Creation, Editing and Compilation. You can easily access all 512K of memory in a Compiled program thru several extended memory commands that can access it in 32K or 8K blocks and single or double bytes.

CBASIC has its own completely integrated Basic Program Editor which allows you to load, edit or create programs for the compiler. It is a full featured editor designed specifically for writing Basic programs. It has block move and copy, program renumbering, automatic line number generation, screen editing, printer control and much more.

Coco 1,2 or 3 Disk $149.00

To order products by mail, send check or money order for the amount of purchase, plus $3.00 for shipping & handling to the address below.

To order by VISA, MASTERCARD or COD call us at (702) 452-0632

(Monday thru Saturday, 8am to 5pm PST).

CER-COMP Ltd.
5566 Ricochet Avenue
Las Vegas, Nevada 89110
702-452-0632

DataPack III Plus V1.1

SUPER SMART TERMINAL PROGRAM
AUTOPILOTand AUTO-LOG Command Processors
X-MODEM DIRECT DISK FILE TRANSFER
VT-100 & VT-52 TERMINAL EMULATION

- No lost data even at 2400 Baud on the Serial port.
- 8 Selectable Display Formats, 32/40/64/80 columns
- ASCII & BINARY disk file transfer via XMODEM.
- Directly record receive data (Data Logging).
- VT-100 emulation for VAX, UNIX and other systems.
- VT-100/52 cursor keys ,position,, PF & Alt. Kbd. keys.
- Programmable Word Length, Parity, Stop Bits .
- Complete Full and Half Duplex operation,
- Send full 128 character set from Keyboard
- Complete Editor, Insert, Delete, Change or Add.
- 9 Variable length, Programmable Macro Key buffers.
- Programmable Printer rates from 110 to 9600 Baud.
- Send Files from the Buffer, Macro Keys or Disk.
- Display or Print the contents of the 50k Buffer.
- Freeze Display & Review information On line.
- Built in Command Menu (Help) Display.
- Built in 2 Drive RAMDISK for 512K RAM.

Supports: R. S. Modem-Pak & Deluxe RS-232 Pak.

Coco 1, 2 or 3 Disk - $59.95

"The SOURCE"

DISASSEMBLER & SOURCE CODE GENERATOR

The SOURCE will allow you to easily & quickly Disassemble Color Computer machine language programs Directly from Disk and generate beautiful, Assembler Source code.

- Automatic label generation.
- Allows specifying FCB, FDB and FCC areas.
- Disassemble programs Directly from disk.
- Automatically locates address.
- Output listings to the Printer, Screen or both.
- Generates Assembler source directly to disk.
- Built in Hex/Ascii dump/display.
- 8 Selectable Display formats 32/40/64/80 .
- Selectable Foreground & Background colors.
- Built in Disk Directory an Kill file commands.
- Menu display with single key commands.
- Written in Ultra Fast Machine Language.

Coco 1, 2 or 3 Disk $49.95

EDT/ASM III

DISK EDITOR ASSEMBLER

EDT/ASM III is a Disk based co-resident Text Editor & Assembler. It is designed to take advantage of the new features available in the CoCo-3 with either 128K or 512K of memory. It has 8 display formats from 32/40/64/80 columns. There is also a free standing ML Debug Monitor.

EDT/ASM III has the most powerful, easy to use Text Editor available in any Editor/Assembler package for the Color Computer.

- Local and Global string search and/or replace.
- Full Screen line editing .
- Easy to use Single key editing commands.
- Load & Save standard ASCII formatted files.
- Block Move & Copy, Insert, Delete, Overtype.
- Create and Edit files larger than memory.

The Assembler features include:

- Supports Conditional IF/THEN/ELSE assembly.
- Supports Disk Library file up to 9 levels deep.
- Supports standard Motorola directives.
- Allows multiple values in FCB & FDB directives
- Allows assembly from the Buffer, Disk or both.

Coco 1, 2 or 3 Disk $59.95

Reviews continued from 25

take three seperate pictures, each through a different filter. The RASCAN software then combines each image in memory to present the 4096 color image. Each colored image may be edited before or after the blending. The potential is unbelievable as you can work with up to 128,000 pixels in 4096 colors. The only drawback is that the subject matter must remain still. Getting your dog or cat to sit still is pretty tough so don't figure on using them. However a family portrait is possible as well as any still life subjects. The 4096 mode is only part of the features of this new system there is still more!

There are three othe video modes all of which operate in high resolution. The first is a 640 x 200 sixteen gray level imaging. In some digitizers this requires several complete scans. RASCAN does it in one! All 16 levels are deleveloped at once and the contrast and brightness can be adjusted directly from the RASCAN control unit as the image is being displayed. Although scan time is about 12 seconds RASCAN comes awfully close to real time computer imagaing as possible on a CoCo. Faster in develpement is the 640 x 200 by four gray level mode. This mode produces a less well defined picture than 16, but takes less memory and can be easier to edit. The final mode is the 320 x 200 sixteen color mode. This mode can use the output from your VCR to produce terrific pictures on your CoCo. VCR owners get ready!

For example you might have a tape of one of the Star Trek movies, or an old John Wayne flick or even video from a family reunion. Just pop the tape in your VCR and use the VCR still mode. Four head VCRs will do much better at this than two head units. The video signal from the VCR goes right into the RASCAN control module. RASCAN software provides paint and palette editing for shaping up your video picture. RASCAN paint and palette software can be superimposed over the image and even in the 64 color mode can do dynamic alterations to the image so you can see what happens as youmake menu selections. The RASCAN images can be easily editied on CoCo Max or ColorMax.

To further assist image manipulation is a something called a HISTOGRAPH. HISTOGRAPH lets you analyze your picture for light levels. Since any picture deals with light or lack of light the inclusion of the HISTOGRAPH software in the driver is very important.

RASCAN also saves a lot of wear and tear on your CoCo as well. The product does not need a y cable. It does not need a Multi-Pak. It was designed to use the joystick ports on the CoCo and even with the ports "tied up" the program provides for picture manipulation without the use of the joysticks.

Of course all of these features have to measured against price and the reliability of the seller. RASCAN sells for $159.95 and that includes everything except for the filter kit. Game Point offers a 15 day no risk guarantee as long as the product is returned undamaged and a one year parts and labor warranty. Considering the number of features, especially the "4096" mode this product is outstanding. If you have been longing for a first class video digitizer in color we recommend RASCAN.

Window Master
CER-COMP Ltd.
5566 Ricochet Ave.
Las Vegas, Nevada, 89110
(702) 452-0632

Pgm. Type: Programming utility/ User interface
Requires : 512K/128K* Coco 3, one drive, mouse, hi-res interface
Price : $69.95

Reviewed by Randy Krippner

The people at Microware had better watch out. Bill Vergona of Cer-Comp could be in the process of making Microware's OS9 and Multi-Vue obsolete.

Window Master is a programming utility and user interface that runs under good old Disk Extended Color BASIC (RS-DOS). Rather than forcing you to learn a whole new operating system, Window Master runs entirely under RS-DOS, adding a host of new commands to the BASIC language as well as a superb hi-res screen display, a RAM disk, programmable function keys and a mouse driver to give you control of the mouse from within BASIC without having to resort to ML drivers. Window Master adds what is known as "event trapping", something no Coco owner should be without. More about that in a moment.

Window Master does windows, of course. Setting up and displaying a window takes only one line of BASIC code. Just type "WINDOW OPEN" followed by the number of the window, and a few parameters that tell Window Master where to put the window on thescreen, what type of window it should be, and how large it should be. That's all there is to it. Window Master takes care of everything else, including restoring the original display after your program closes the window.

Windows can be layered on top of one another. If the window you want to use is currently covered by a different window, just put the mouse pointer on the top of the window you want to use and click, and that window will jump to the top of the screen and become active. You, the programmer, don't have to do a thing. Window Master handles all of the work.

Window Master features what is called "event trapping", which is what makes many of the goodies in this tool work. Event trapping can trap a particular event, such as the clicking of the mouse button, the

Reviews continued on 28

Reviews continued from 27

can do this even if your program is off doing something else. When it notices such an event taking place, it will automatically jump to a subroutine you defined early with an ON MENU GOSUB, ON MOUSE GOSUB or one of the other event trap statements.

What this means to the programmer is that it is no longer necessary to have the program constantly scan for a particular event, and thus waste valuable execution time. Window Master handles it for you. All you have to do is write the code to perform the desired task if the user uses one of the menus, clicks on a button or icon or whatever.

Did I hear someone ask what a "button" is? A button is - well, a button! But this one you push with the mouse instead of with your finger. A programmer can set up "push buttons" on the screen to enable the user to select various options, such as Yes or No, On or Off, or anything he or she wishes. Then, using Window Master's superb event trapping capabilities, you can have the program jump to the desired subroutine or perform the desired activity depending on which "button" the user pushes.

Window Master also supports redefinable icons. Sixteen icons can be used. They are pre-defined by Window Master when first booted up, but can be re-defined by the programmer. Icons can be used to get the user's attention, or can be used like buttons. Window Master's event trapping capabilities will tell you if the user clicked on a particular icon.

The program features the best screen driver I've ever seen. Since Window Master operates in the hi-res graphics mode, you aren't limited to a single character set. You can even define your own unique character sets with an optional editing program. It also supports underlining, italics, bold, sub and super scripts, double width, double height and quad height characters. Both the standard Coco 3 "LOCATE" statement and the Coco II "PRINT @" statements are supported as well. Other goodies include a full 68 gran RAM disk for virtually instant access to data and programs stored in it. There is a built in hi-res screen dump utility that will do a graphics screen dump of either the entire screen or just the currently active window. Almost every key on the keyboard can become a programmable function key by just holding down the ALT key and then pressing the desired key. It has an enhanced BASIC program editor, called up by typing WEDIT instead of EDIT, that is much more flexible than the standard EDIT is. (The standard EDIT is still there, if you want to use it.)

Window Master adds an extremely useful function called DFILE$ which will pop up a window on screen and display the directory of a specified disk drive. The user can then

Reviews continued on 29

Reviews continued from 28

select a file with the mouse and the DFILE$ command returns the name of the user selected. Again, Window Master handles all of the work. All the programmer has to do is type a single BASIC line.

Some of the things Window Master does are fixes to bugs in the standard RS-DOS. Under Window Master, DSKINI no longer wipes out program memory. The PCLEAR 0 bug is gone. You can run the printer at 19,200 baud without having mess around with the slowdown poke before disk access.

I've been using Window Master for almost a month, and have been very impressed. So impressed, in fact, that three days after the review copy arrived, I ordered a copy of my own from Cer-Comp.

There are drawbacks, of course. Since Window Master must entirely re-map the computer's memory and install its own drivers for the mouse, event trapping, etc., Window Master will not be compatible with machine language programs. It is compatible with BASIC programs, especially if you flip back into one of the standard screen display modes.

There is, at the moment, no commercial software available that runs under Window Master, but considering the enormous capabilities of this unique operating environment and programming tool, you can be sure that it won't be long before applications software begins to appear in both the commercial market and in public domain.

Window Master is copy protected. I can't really blame Cer-Comp for protecting Window Master. The theft of software is still an enormous problem, and a program as powerful as Window Master is a prime candidate for illegal copying. I understand, but I don't have to like it.

The documentation has some problems. Although it gives a good description of the basic Window Master commands, a lot is left out. The icons, for example, can be redefined by the user and stored on disk to be re-loaded later, but the documentation doesn't tell you how. An icon editor program is available. It came with the review copy of Window Master, but not with the copy I purchased. By listing this program I figured out how to load my own icon sets for use by my own software

The biggest problem is overscan. Window Master operates in a non-standard 640 X 225 or 340 X 225 display mode instead of the usual 640 X 192 or 340 X 192 modes. This means that on some tv sets or composite monitors, the top left and bottom left corners of the screen may not be visible. On televisions or monitors that have poorly aligned displays, the top and bottom lines of the display may not be visible at all.

Note, however, that this is NOT a problem with Window Master, but with the monitor or tv being used. A simple adjustment of the height and width of the display is usually all that is necessary to eliminate the overscan problems, and if you're usinga RGB monitor, there should be no trouble at all with overscan. (Don't attempt to open up a TV or monitor and adjust the image size yourself unless you know what you're doing! There are capacitors in many monitors and TV's that can retain enough of a charge to give you a painful and perhaps even fatal jolt if you stick your fingers in the wrong place, even if the TV or monitor has been unplugged for days.)

To sum things up, Window Master is a truly superb piece of software. It gives you a "point and click" user interface that is in many ways far superior to that of OS9 L2 and Multi-Vue, and all from within good old RS-DOS. Highly recommended.

* Note: Although a 128K version of Window Master comes with the package, it is limited, and Cer-Comp has stated that future upgrades of Window Master will require 512K of RAM.

Editors Note: Just before press time Cer-Comp send us a copy of a program tentatively called "Window Writer." Similar in concept to their Textpro 4 word processor, "Window Writer" is designed to run "under" the Window Master environment.

Simply Better Word Processor
Simply Better Software
P.O. Box 20726
Portland, OR 97220
(800) 248-8420 order line
(503) 254-7225 tech support

Program Type: CoCo 3 Word Processor
Requires : 128K CoCo 3 * 512K for additional buffers
Price : $29.95

Reviewed by Bill Laurence

The Simply Better ad in the last issue of *CoCo Clipboard* made two interesting statements: "** Setting New Standards... **Setting New Prices..." This word processor does both, it's been widely received and once again ups the ante in the CoCo word processor market place.

First the basics.....:

1. The program will run on either a CoCo 3 with 128 or 512k of memory. It runs in the 80 column "native" mode. It has extremely fast screen updates and it is difficult if not impossible to type faster than the program can handle. It comes with an excellent manual with a continuous plastic spiral binding which props up nicely on top of your computer.

2. It can handle imbedded printer control codes as well as pre-defined printer control codes. I have always placed a high value on

Reviews continued on 30

Reviews continued from 29

imbedded commands. There have been a number of programs that will let you pre-define, but every now and then you need to use something you didn't think about when you started. Going back to the boot program or set up menus often breaks a train of thought while writing, imbedded commands let you sail "write" along. It will work with any printer that supports regular spaced characters. It does not support proportional spacing - which very few if any WP's for the CoCo do. (TextPro IV from Cer-Comp being an exception.)

3. An exceptional price of $29.95! Maybe program author Dale Rickerts has a rich uncle someplace. I'm not sure just how he can produce this software at such a low price and maintain such quality. If nothing else it should totally discourage anybody from pirating the product. Why waste your time with a bogus copy when the original with the manual is so in-expensive!

Now for the details.....:

This program comes with a configuration program that is just one of the best I've seen. Simply Better will let you pre-define up to five printer attributes and will display those attributes *in color* within your text. You can have a backround color and a foreground color to to this. For example a red backround with white letters might be your choice to define italics. A white backround with blue for double wide. It's your choice for colors and functions. All you need is the printer control codes from your manual for each attribute and your ready. The program will also display true underlining on the screen when that feature is called, a very nice touch.

The program has search, find and replace. It does block copy and move. It will provide up to 480k of storage in a 512k system and 90k in the 128k machines. The machine will automatically do disk saves if you wish and although 480k is more than you can fit onto a single disk the boot program provides for partitioning the buffer space into print spool and actual text buffer. The "boot" program is very flexible. You can have several boot programs on your floppy, each with a different name. You can have one setup file for letter writing, magazine writing, business purposes etc. Just type in the right boot file name when prompted for the type of work you need done. If you're worried about over loading a disk simply set the buffer space at about 140k and leave the rest as a giant spooler. I haven't found anything that won't fit into a 480k buffer in normal writing situations.

You can set page length, line length, margins, tabs, headers, footers and do dynamic formatting changes within a document. You can have it automatically place page numbers anywhere you want and can be configured for hearing impaired to flash a border when an error is made instead of beeping. It is fully RS-DOS and RGB-DOS compatible and will work with RGB monitors, TV sets (40 column is the only readable size) and either amber or green composite monitors. Many of these features are standard on a number of quality CoCo 3 word processors, but there are several different features which let this program really stand out.

You can automatically create a table of contents and/or an index table while you are writing. Simply hitting the F1 + I key will mark an item for the index. A press of the F1 + T will set up a contents marker. You can't do both at once, but so what! Just having this capacity is a real plus. This would be very helpful as you are doing an article outline - just hit the F1 + T and the program will create a contents page and you can then write from that point. For technical writers this feature is a welcome plus as well as for story or feature writers.

The program will let you pre-define 10 tasks or macros. Control <Cntrl> Z let's you create a task and <Cntrl> 0-9 will execute it. What can you set up with a task? How about instantly setting up your ten most often used multi-syllabic tech terms like "Dual inverted 28 pin multi-sync flip flops" or "demisemiquavers" (a British term for a 32nd note) and with two keystrokes put them into your text. You could also do block moves, page moves, disk saves almost anything even imbed one task inside another.

Simply Better also has a forms fill feature which is just as slick as all get out. Maybe you haven't caught the latest fad over in the MS-DOS world yet but form filling programs are all the rage. Just pop up the form on the screen, answer all the questions and print it out. Simply Better does everything but put a copy of the form on the screen. After defining what goes where and how, you can simply insert a pre-printed form into your printer and away it goes! Did some one mention a 1040 form? It could be done, as well as invoices, statements etc. It's not as easily done as on a dedicated forms program but it is a nice feature.

Simply Better does mail merge and automatic print spooling by using a split window feature. It will sort sections of text. It will handle the standard bit banger output port or optional parallel port. You can print right, left or centered as well as full justification and has a view mode to "see" just what your text looks like. The only thing the software doesn't include is a spell checker, but it's disk files are regular ASCII and can easily be handled by any number of spell checker programs.

On top of all this Simply Better provides an "800" line for your orders. A package arrived a few weeks after my original package and had disk with a program up-

Reviews continued on 31

Reviews continued from 30

grade. Included with this package were several discount coupons from other CoCo third party vendors as well as a coupon for a newsletter which will support Simply Better users. These certainly aren't part of the program itself but it's a nice touch that Simply Better Software has made available to their customers. You'll even find a subscription blank to our favorite computer magazine - *CoCo Clipboard!*

The ante has been raised in the CoCo word processing market by this low cost, full featured product. At twice the price it's worth every penny.

THE KJV BIBLE ON DISK
BDS Software
P.O. Box 485
Glenview, IL 60025-0485

Pgm. Type : Books of the Holy Bible in ASCII files
Requires : 32K CoCo 2 or CoCo 3, Word Processor
Price : $3.00 per disk

Reviewed by Boisy G. Pitre

Well, I never thought I would see it, but here it is! Finally, someone took the challenge and is attempting to put the King James Version of the Holy Bible on a CoCo disk!!

M. David Johnson of BDS Software has diligently worked to put several books of the Bible on disk. His main goal is to place all 66 books of the Bible on 40 disks. So far, he has 14 New Testament books on disks, with more to come.

The disk that I received for review contained the book of Matthew. The disk includes a special file called README.TXT, which helps you on how to operate the package. Also there are several sheets of documentation to get you started.

Although a word processor is recommended for serious viewing of the chapters, you can use a program called TYPE.BAS included on the disk. TYPE.BAS will display each scripture, one at a time on the screen. However, I would suggest using a good word processor with at least 24K memory to work with and one that has a fast string search routine.

Johnson has separated the book of Matthew into 7 ASCII files. The files, named MAT01.TXT - MAT07.TXT contains several chapters. Johnson's approach to splitting the books is excellent. If the book of Matthew were one big text file, it would be much to big and bulky for smaller word processors to handle. It also helps make finding chapters easier.

As far as I could tell, there were no typographical errors in the ASCII files. It's like reading your Bible right on the screen! Just think what it would be like for this stuff to be running in OS-9 windows concurrently!

I believe that Mr. Johnson is on the right track with his project. The documentation is complete, the files are typographically correct, and the price can't be beat. I recommend these and upcoming books of the Holy Bible on disk without reservation for anyone seriously studying God's Word.

CoCo
Clipboard
Magazine

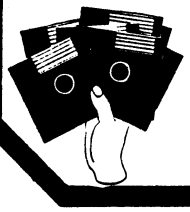
Please Join Us

For our CoCo Clipboard Conferences of CompuServe!

Our on-line - LIVE - sessions are scheduled for the second Saturday of each month at 9pm Eastern.

Join Dan Robins and Ted Paul for up to date information every month!

The Assembly Line



Kraig Brockschmidt

Welcome to The Assembly Line. This month's article is a bit longer than usual but covers a more complicated and much more interesting topic. Also, my Compuserve ID 76701,76 is not valid anymore (this Microsoft account changed hands), as some of you might have noticed. In any case, from now on I'll keep Ted Paul informed of changes, and he should be able to provide you with my current ID number.

Color Computer Sound:

One of the more elusive features of the CoCo are its sound capabilites. You hear seemingly complex sounds coming from your speaker, but how are those sounds actually produced? In all reality, complex sounds can be simple to program but require an understanding of sound itself.

1) The Physics of Sound:

To understand sound we must start with a little physics. Sounds aresimply compression waves travelling through the air. When these waves reachyour ear, they put a changing pressure on your eardrum, causing it to vibrate. If you carefully watch a speaker as music is played through it, you will see it move in and out. This motion is a vibration, and to your ears this vibration is sensed as sound. Two important quantities, or parameters, of the vibration affect the sound that is heard: volume (amplitude) and frequency.

Volume is how far, in and out, the eardrum is forced to move. The more it moves, the louder the sound (which is why your ears hurt from loud sounds, because your eardrums are being moved to the point where it's painful). Faint sounds, which have a very low volume, barely move the eardrum.

Frequency is a measure of how fast the eardrum is moving back and forth, usually specified in Hertz (Hz), or vibrations per second. The higher the frequency the higher the pitch of the sound. Normal human ears are capable of sensing frequencies from 20Hz to about 15000Hz.

The patterns these parameters follow are also important, as they define the 'quality' or 'timbre' of the sound. For example, if the volume changes according to a sine wave at a frequency of 60Hz, you would hear common speaker hum from your stereo. A plot of volume vs. time for a 60Hz sine wave is shown in Figure 1. The period is the time required to complete one cycle of the wave, in this case it is 1/60th of a second (the

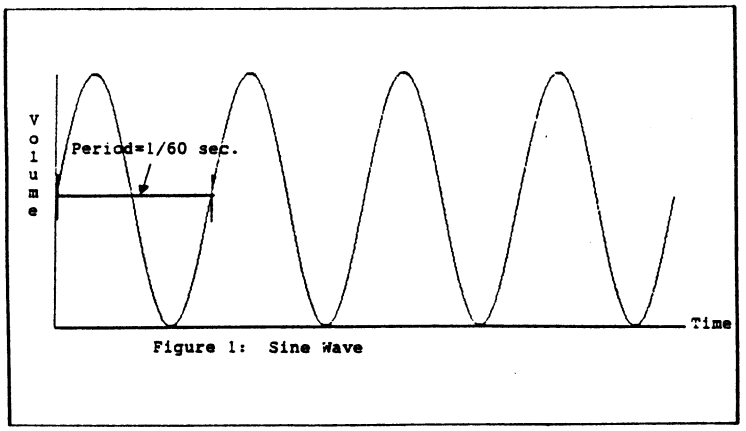


Figure 1: Sine Wave

period is also the inverse of the frequency). Striking a bell produces a wave similar to that in Figure 2, where the volume follows a waveform that decreases in amplitude with time. Examples of these and

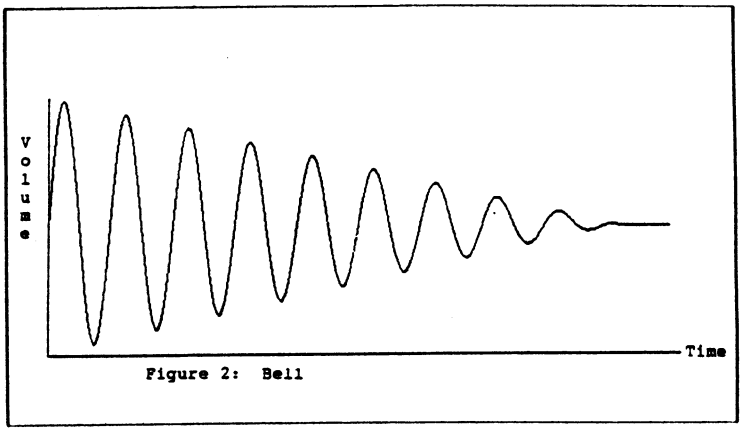


Figure 2: Bell

other situations are explained later. Musical instruments emit more complex waveforms such as that in Figure 3. For more information on waves and sound, consult a physics textbook.

2) Sound Generation on the CoCo:

To generate sound from the Color Com-

Assembly Line continued on 33

Assembly Line continued from 32

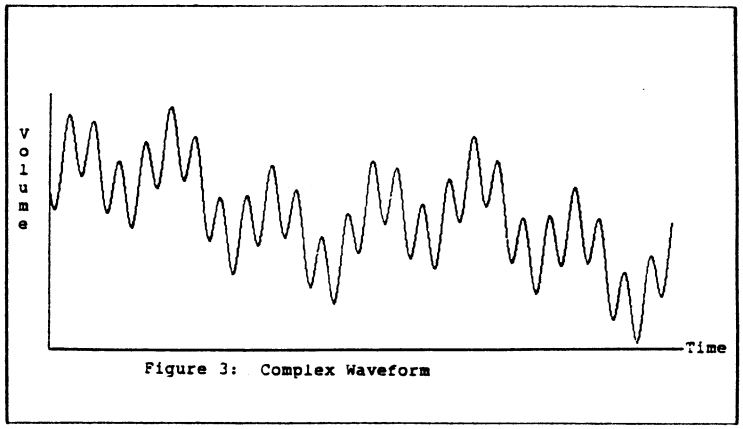


Figure 3: Complex Waveform

puter, we simply need to control the volume, volume pattern, and frequency of motion of the system's speaker. The CoCo has two methods of generating sound, using either a single bit or 6-bits. Since the single bit sound has quite limited uses, my concentration will be the 6-Bit sound capability.

The first step is to enable 6-Bit sound on the CoCo. To enable this sound capability, bit 3 of the I/O port $FF23 must be set. If bit 3 is clear, then the sound is disabled. Lines 810-840 of the assembly listing show how to enable the sound. Two BASIC ROM calls, JSR $A976 and JSR $A974, can be used to enable and disable the sound respectively.

The next step is sending values (volume parameters) to the 6-Bit sound port at specific intervals (which determine the frequency). The 6-Bit sound port is the byte at $FF20. Only the upper 6 bits of $FF20 are used for sound, hence the name "6-Bit sound". Overall, the 8 bits of $FF20 are used as follows:

```
Bit  0:   Cassette data intput
Bit  1:   RS-232 data output
Bit  2:   6-Bit sound least significant bit
Bits 3-6: Four middle 6-Bit sound bits
Bit  7:   6-Bit sound most significant bit
```

Since bit 1 sends data to the serial port, it is best to leave your printer off during experimentation. The samples accompanying this article will cause any on-line printer to print garbage characters. In situations where the printer might be on, AND any sound data with $FD, which always clears the RS-232 data output.

Bits 2-7 define a 6-Bit volume parameter. The larger the number the louder the sound. This digital 6-Bit value is converted into an analog voltage, which is then applied to the speaker. The voltage ranges from 0 volts to 5v, in steps of about 0.08v. The voltage applied to the speaker moves it to a certain position away from its rest position--the higher the voltage the father it moves. If this voltage is changed, i.e. a different number is sent to the sound port, a small 'pop' is heard. Again, the larger the change the louder the pop, since a large change generates a large motion. So if a 0 is sent to the sound port, followed by a $FF, another 0, and so on, a square wave is generated, as shown in Figure 4.

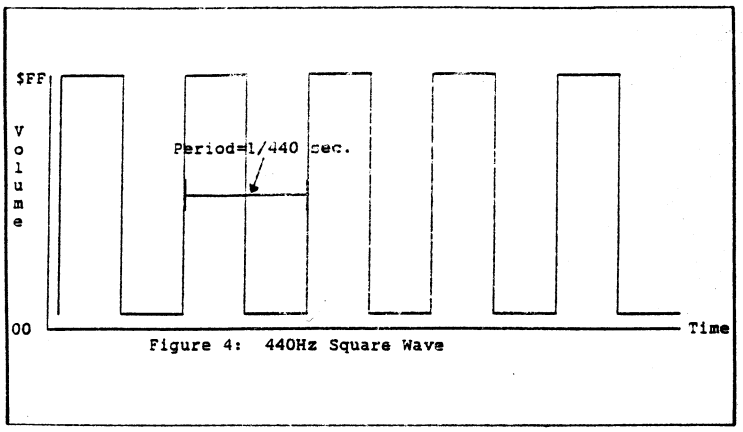


Figure 4: 440Hz Square Wave

The frequency of this square wave is determined by the time delay between sending two zeros (or FF's) to the port, as shown in the Figure. As stated before, the actual time delay is the period and the frequency is the inverse of the period. Since we can control the delay between changes in the volume, we can control the frequency. Normally this is done through a loop of some sort, since executing each loop takes a certain amount of time.

Simple enough? Let's look at the examples now. As always, the program for this article can be directly assembled or POKEd into memory by the BASIC program. EXECing the program will bring up a menu giving a selection of 8 sound demos. To stop any of the sounds you must hit the reset button, then EXEC to re-enter the program. The following is a short desciption of each sound and how it is produced.

3) Sound Examples

1 and 2: 440Hz Square Waves--Figure 4, lines 1070-1280 in the listing.

The first square wave is generated as described above, by sending an $FF to the sound port, delaying for some time, sending a zero, delaying, etc. The second square wave sounds identical to the first, but is produced slightly different. The difference is that the first wave sends a volume, then waits, whereas the second sends a volume each delay cycle. This is simply to prove that keeping the volume constant does not produce a change in the sound.

You will notice and proably question the values after the LDB instructions in lines 1100 and 1190. These were calculated as follows:

Each CPU cycle for my computer is 1.13177 microsecond. This may be different for other computers.

Between storing an $FF and a 0, there's one STA extended, one LDB immediate, $C4 (or $62) DECB and BNE, one COMA, and a BRA instruction.

Assembly Line continued on 35

It's Word Processor Trade-in Time Again!

Send us ANY word processor and get VIP Writer III for $49.95!

Include $3 for shipping. Send $52.95 and your old word processor to the address below. Offer expires 9/15/89 so Hurry!

VIP Writer III Ver. 2 *Cat. #90-908

VIP Writer III offers screen widths of 32, 40, 64 & 80 - all with 24 lines and actual lower case letters using the CoCo 3's hardware display. It runs at double clock speed and has 4-color menus making VIP Writer III FAST and EASY to use! You can choose foreground, background, hilite and cursor colors from up to 64 hues. Color can be turned ON or OFF for the best possible display using a monochrome monitor or TV set. VIP Writer III has a context sensitive help facility to display command usage in easy to read colored windows.

CUSTOMIZER & PRINTER INSTALLER

VIP Writer III comes with a configuration / printer installation program which lets you customize VIP Writer III to suit your own liking. You can set screen width and colors as well as margins and more. You can also install your own printer and set interface type (serial, parallel or J&M), baud rate, line feeds, etc. Once done, you never have to enter these parameters again! VIP Writer III will load n' go with your custom configuration every time!

MORE TOTAL TEXT STORAGE

VIP Writer III has 106K total text storage in a 128K CoCo 3 (495K in 512K). VIP Writer III creates ASCII text files which are compatible with all other VIP Programs as well as other programs which use ASCII files. You can use VIP Writer III to even type BASIC programs! There is a 48K text buffer (438K in a 512K CoCo 3) and disk file linking allowing virtually unlimited text space. VIP Writer III works with up to four disk drives and lets you display directories and free space as well as rename or kill disk files. In addition VIP Writer III is 100% compatible with the RGB Computer Systems Hard Disk.

POWERFUL EDITING FEATURES

VIP Writer III has a full featured screen editor which can be used to edit text with lines up to 240 characters long with or without automatic word wrap around. You can select type-over mode or insert mode. There is even an OOPS command to recall a cleared text buffer. Other editing features include: Type-ahead • typamatic key repeat and key beep for flawless text entry • end of line bell • full four way cursor control with scrolling • top of textfile • bottom of textfile • page up • page down • top of screen • bottom of screen • beginning of line • end of line • left one word • right one word • DELETE character, to beginning or end of line, word to the left or right, or entire line • INSERT character or line • LOCATE and/or CHANGE or DELETE single or multiple occurrence using wildcards • BLOCK copy, move or delete with up to TEN simultaneous block manipulations • TAB key and programmable tab stops • word count • line restore • three PROGRAMMABLE FUNCTIONS to perform tasks such as auto column creation and multiple copy printing.

Writer III or Library /W owners: Upgrade to the VIP Writer III 2.0 for $10 + $3 S/H. Send ORIGINAL disk and $13 total.

Rated "BEST" in RAINBOW Sept. 1988

AUTOMATIC TEXT FORMATTING

VIP Writer III automatically formats your text for you or allows you to format your text in any way you wish. You can change the top, bottom, left or right margin and page length. You can set your text flush left, center or flush right. You can turn right hand justification on or off. You can have headers, footers, page numbers and TWO auxiliary lines which can appear on odd, even or all pages. You can also select the line on which they appear! You can even change the line spacing! Parameters can be altered ANYWHERE !

PREVIEW PRINT FORMAT WINDOW

VIP Writer III features an exclusive format window which allows you to preview your document BEFORE PRINTING IT! You are able to move up, down, left and right to see centered and justified text, margins, page breaks, broken paragraphs, orphan lines etc.

PRINTING VERSATILITY

VIP Writer III prints TWICE as fast as any other CoCo word processor! It supports most serial or parallel printers using J&M JFD-CP or Rainbow interface and gives you the ability to select baud rates from 110 to 19,200. You can imbed printer control codes anywhere in your text file EVEN WITHIN JUSTIFIED TEXT! VIP Writer III also has TWENTY programmable printer macros which allow you to easily control all of your printers capabilities such as bold, underline, italics and superscript using simple key strokes. Other features include: multiple copy printing • single sheet pause • line feeds.

BUILT IN PRINT SPOOLING

VIP Writer III has a print spooler with a 57,000 character buffer which allows you to print one document WHILE you are editing another. You don't have to wait until your printer is done before starting another job! Some word processors DO NOT include this feature!

50,000 WORD SPELLING CHECKER

VIP Writer III includes VIP Speller (not FREEWARE) to check your text for misspelled words It has a 50,000 (not 20,000) word dictionary that can be added to or edited.

QUALITY DOCUMENTATION

VIP Writer III comes with a well written 125 page manual which is Laser printed, not dot-matrix like the competition. It includes a tutorial, glossary of terms and examples for the beginner as well as a complete index! VIP Writer III is truly the BEST you can buy.

VIP Writer III includes VIP Speller 1.1. **DISK $79.95**

VIP Writer owners: Upgrade to the Writer III 2.0 for $49.95 + $3 S/H. Send original disk and $52.95 total.

VIP Database III *Cat. #90-915

VIP Database III features selectable screen displays of 40, 64 or 80 characters by 24 lines with choice of 64 foreground, background, hilite and cursor colors for EASY DATA ENTRY. It uses the CoCo 3's hardware screen and double clock speed to be the FASTEST database available! VIP Database III will handle as many records as will fit on your disks and is structured in a simple and easy to understand menu system with full prompting for easy operation. Your data is stored in records of your own design. All files are fully indexed for speed and efficiency. IN-MEMORY SORT of records is LIGHTNING FAST and provides for easy listing of names, figures, addresses, etc., in ascending or descending alphabetical or numeric order. Records can be searched for specific entries using multiple search criteria. The built-in mail-merge lets you sort and print mailing lists, print form letters, address envelopes - the list is endless. The built-in MATH PACKAGE even performs arithmetic operations and updates other fields. VIP Database III also has a print spooler and report generator which uses print forms you create. **DISK $69.95**

VIP Database owners: Upgrade to the VIP Database III for $39.95 + $3 S/H. Send ORIGINAL disk and $42.95 total.

VIP Library /WDCE $179.95

The VIP Library /WDCE (Writer Database Calc Enhanced) combines all six popular VIP application programs - VIP Writer III, Database III, Calc III, Speller, Terminal and Disk-ZAP - into one program on one disk called VIP Desktop. For VIP Library shipping please add $4 USA. $5 Canada. $10 Foreign.

VIP Library owners: Upgrade to the VIP Library /WDCE for $99.95 + $3 S/H. Send ORIGINAL disk and $102.95 total.

VIP Library /WDE owners: Upgrade to the VIP Library /WDCE for $10 + $3 S/H. Send ORIGINAL disk and $13 total.

VIP Calc III *Cat. #90-916

FAST 4-color POPUP menus • PRINT SPOOLER
32, 40, 64 and 80 Column HARDWARE display!
Runs VERY VERY FAST at double clock speed!

Now every CoCo 3 owner has access to a calculating and planning tool better than VisiCalc™, containing all its features and commands and then some. VIP Calc III allows a large worksheet with up to 512 columns by 1024 rows! In addition, VIP Calc III has up to 16 windows which allow you to compare and contrast results of changes. Other features include 8 AND 16 digit precision • trig. functions • averaging • algebraic functions • column and row ascending and descending SORTS • locate formulas or titles in cells • block move and replicate • global or local column width • limitless programmable functions • create BAR charts. Embed printer control codes for customized printing. Combine spreadsheet data with VIP Writer documents to create ledgers, projections, statistical & financial budgets and reports. **DISK $69.95**

VIP Calc owners: Upgrade to the VIP Calc III for $29.95 + $3 S/H. Send original disk and $32.95 total.

Buy RGB-DOS for $29.95,

Get Hard Disk support, new commands and a Disk Drive FREE!*

Sounds too good to be true? If you own a Radio Shack FD 502 or other double sided Disk Drive, using RGB-DOS, you can access the other side of your Disk Drive giving a second disk drive absolutely free!* RGB-DOS also supports up to 2 Hard Drives that can be used by DISK BASIC as well as OS-9. RGB-DOS works with CoCo 1, 2 and 3 and supports double sided drives and faster stepping rates. Other features include: Full screen directory display shows drive #, free space and even a disk name! • RUNM command and FLEXIKEY Last Command Recall and Edit system • EPROM version executes any program when CoCo is turned on for hands free start-up. 64K Req'd.

SD Enterprises credit card / COD order line.

1-800-322-9873 EXT 3

SD Enterprises info line (805) 566-1317
P O Box 621 Carpinteria, Ca. 93013

Non VIP Library orders add $3 for shipping and handling in USA. Canada $4. Foreign $6. COD orders add an additional $2.75. Checks allow 3 weeks for delivery. California residents add 6% sales tax.

* Available through your nearby Radio Shack Computer Center® and participating Radio Shack stores and dealers or order direct from Express Order℠ by dialing 1-800-321-3133.

Assembly Line continued from 33

Looking up the timing for each instruction (on a 6809 spec sheet), the total number of CPU cycles for the first square wave is 5+2+(196*(2+3))+2+3=992 CPU cycles.

For the second wave the loop sums to 5+2+(98*(5+2+3))+2+3=992 CPU cycles.

Since the total cycles between sending two $FF's is 992*2=1984, the period is 1.13177 microseconds/cycle * 1984 cycles, which is .00225 seconds. Inverting gives about frequency=440Hz, (445Hz actually, but my CPU cycle time of 1.13177 is not exact, as I measured it by making a sound on the CoCo and comparing it by ear to a piano.)

Changing the frequency means to simply change the number of instructions between sending two $FF volumes. I should point out that the bulk of the delay is due to a counting loop, which executes for 980 CPU cycles. Since any instructions that take 980 CPU cycles would suffice here, graphics animation or anything else can be performed while generating this sound. I may present an example of this in a future article.

3: Diminishing Square Wave--Figure 5, 1290-1450.

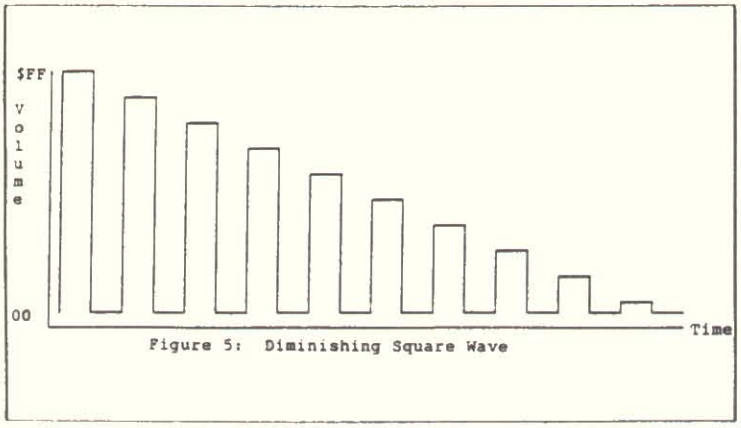


Figure 5: Diminishing Square Wave

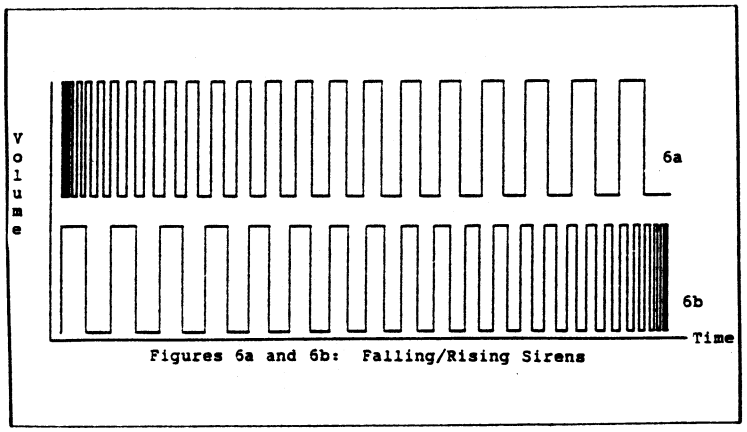
This sound is similar to a bell, with a frequancy of around 412Hz, produced by decreasing the volume with time. This is done by sending the A register to the sound port, waiting, sending a zero, waiting again, and then decrementing A. If A is not zero, the above loop repeats, otherwise there's a long delay after which the entire sound is repeated. The A register starts at maximum volume ($FF) and diminishes to 0. When I made this it reminded me of a new car telling me that my door is ajar, my door is ajar...

4 and 5: Falling/Rising Sirens--Figure 6, a and b. 1460-1700.

Instead of changing the volume as in exampIle 3, these two 'sirens' decrease or

Assembly Line continued on 36

Assembly Line continued from 35



Figures 6a and 6b: Falling/Rising Sirens

increase the frequency with time. The delay parameter is stored in the B register, and the volume follows a simple square wave. The first siren starts with B=1 (short delay -> high frequency) and continually increases, thus lowering the frequency. When B changes from 0 to 1 (0 is longest delay), the delay is again short, thus the frequency goes high again. The second siren is the direct opposite, starting with a low frequency and getting higher. Both pieces of code should be fairly easy to follow in the assembly listing.

6: Noise--Figure 7. 1710-1800.

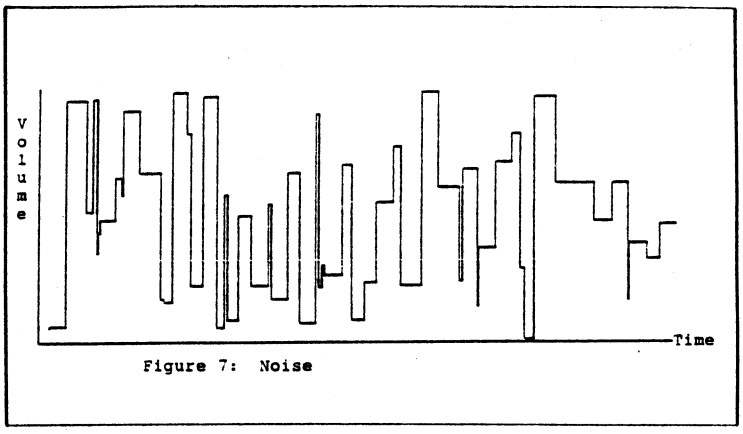


Figure 7: Noise

Noise is generated by varying the volume and frequency of the sound randomly. For the example I have simply taken numbers from the computer's memory for the parameters, and play them over and over. There's really nothing more to it than that...

7: Combination--1810-1950

This odd sound is a combination of changing the volume and frequency in a strange way. For the square wave a COMA was used to change the volume and the siren either incremented or decremented the delay time. This sound demonstrates what results by exchanging the parameters, complementing one and adding 3, and negating the other and adding 4. I encourage you to change the instructions in lines 1850-1930 and see what other sounds result. Note that whatever instructions you chose should not drive the volume parameter to a constant, since no sound will be generated. In my example, I used and EXG, COM, NEG, and ADD instructions, which together will keep registers A and B continually changing. Instructions to avoid are LSR, AND, and CLR.

8: Pure (more or less) Sine Wave-See Figure 1. 1960-2250.

Beyond the realm of square waves and sirens are a vast multitude of complex sounds, which are best generated by use of a waveform stored in memory. This example shows how to generate a good sine wave sound. You should immediately notice the difference in timbre between the sine wave and the square waves of 1 and 2.

The waveform for the sine wave is stored in the table labelled SINE, which contains 256 volume values. For ease of entering the values, I initially added $20 to each value, which converted made most them to ASCII characters and thus could be entered using FCC pseudo-ops. In addition, only half of the wave is initially defined. So at the start of the program, this table is completed by 'mirroring' the first half to the second, then subtracting $20 from each value. Finally, each value is multiplied by 4, resulting in a full 256-byte table with values from 0 to 255, in steps of 4, (the lower two bits of each value do not affect the sound). The value at FLAG is changed to indicate that the table has already been completed, so that subsequent EXECs of the program do not repeat the alterations. If you substitue your own waveform in place of the sine wave, you will want to initially store anything besides 0 in FLAG so the specific changes for my sine wave do not ruin your waveform.

It should be apparent that the short loop in lines 2200-2250 simply extracts a volume from the sine wave table, sends it to the output, increments the B register, and repeats. The B register is used as an index of the current position in the table. Since it is used as a two's complement value (-128 to +127) in the LDA B,X instruction, X is set to the middle of the table.

This loop effectively moves the speaker in a sine wave pattern, at the calculated frequency of 230.1Hz. But wait! What happened to a delay parameter? Well, with waveform sounds there is no real need for a delay parameter. A delay was used to define the length of a period in the previous sounds using square waves. A waveform in a table will define some number of periods of that waveform. In the example, exactly one period of a sine wave is defined. Since a fixed number of instructions are executed for each repetition of the waveform, the delay is inherent. The particular code in the listing takes 15 CPU cycles per value extracted from the table and sent to the speaker. There are 256 values in the table, it takes 256 loops or 3840 CPU cycles to

Assembly Line continued on 37

Assembly Line continued from 36

sound one period of the waveform. Multiplying by the CPU cycle time (1.13177 microseconds/CPU cycle) gives .0043459 second/period, or an inherent frequency of 230.1Hz, which is almost exactly the Bb note below middle C on a piano. I compared this to my electronic keyboard that can emit pure sine waves, and the frequency was very close.

How then can we vary the frequency? This is done simply by changing the way in which we extract values from the table. In the example, the values are taken one at a time in sequence, i.e. the ADDB #1 instruction moves the index pointer by one for each loop. Let's see what happens when 2 is added to B--instead of sending each value in the table consecutively, we send every other value. Thus it takes only 128 loops to go through the entire table and complete one period of the waveform, which is equivalent to 1920 CPU cycles. This generates a sine wave with a frequency of 460.2Hz. It can also be looked at that in 256 loops, the table is read through twice, as only every other value is used. Since 2 periods are completed in 3840 CPU cycles, each period is 1/2 * .0043459 second, so the frequency is 2*230.1 =460.2Hz, which is Bb above middle C.

Continuing these calculations, if we add some number N to the B register for each loop, the frequency of the sine wave is 230.1Hz*N. Note that as N grows larger, the sound actually generated starts to deviate from a good sine wave. For example, at N=64, the values sent to the sound output are 0, 255, 0, 255, 0, and so on, which looks an awful lot like a square wave (with a frequency of 14700Hz, just about the limit of a normal person's hearing!) Also, since the inherent frequency is so high (230.1Hz), and since we can only use multiuples of that frequency, the code shown in the example has quite limited uses as a music synthesizer. Besides having around a 6-note capability, the high notes start to sound like square waves because of the jump in the volume values sent to the speaker.

The way around this problem is to simply make a loop that uses more CPU cycles per iteration. If each loop used 38400 CPU cycles instead of 3840, then the inherent frequency would be 23.01Hz, which is very low on a piano keyboard.However, to obtain high frequencies, large values of N must be used, which destroys and sine wave qualities of the sound. Unfortunately, the only way to get a good sound at a high frequency is to play a sharper waveform in a very short time. Since the 6809 can only execute so many instructions per second, most high frequencies do not follow a waveform very well. In short, a faster microprocessor is needed...

Assembly Line continued on 38

The problem that still remains is that the frequencies of musical notes are not in integral multiples of 23.01. What is needed then, is a method of counting N in fractions, which is easily accomplished by a bit of assembly code. The trick is to use a two byte counter where the low byte is the fractional part of N, and the upper byte the whole part. For example, if N=18.19 is needed, then add 18 to the upper byte and 50 to the lower (50=255*.19). Of course, the entire 16-bit count must always be retained. This little trick is also useful in other application where fractions are needed.

The only subject left on sound is generating multi-voice music, which is done with the waveform method above, but I'll leave that to a future article.

I hope these examples and explanations will aid everyone in understanding sound and how it is generated by computers. It was certainly educational for me (I started this article with only vague ideas of how CoCo sound worked), and I had a lot of fun programming the examples.

Listing # 1 - Produces the 7 graphs shown in the program on your screen.

```
5 POKE&HFFD9,0
6 ON BRK GOTO 2000
10 WIDTH80:ATTR3,2:RGB:CLS
20 INPUT"WHICH FIGURE";A
25 HSCREEN3:PALETTE0,8:PALETTE1,
63
30 ON A GOTO 100,200,300,400,500
,600,700
40 GOTO 10
100 GOSUB 1000 'DRAW AXIS
105 X1=0:Y1=95
110 FOR X=0 TO 580
120 Y=95-SIN(X/23.07)*85
130 GOSUB999:NEXT X
140 GOTO 140
200 GOSUB1000
205 X1=0:Y1=95:A=85
206 FOR X=0 TO 580
210 Y=95-SIN(X/9.23)*A
220 A=A-.16:IF A<0 THEN A=0
230 GOSUB999:NEXT X
240 GOTO 240
300 GOSUB1000
310 X1=0:Y1=95
315 FOR X=0 TO 580
320 Y=95-((2*SIN(X/50)+2*COS(X/2
00)-4.5*SIN(X/10)*COS(X/6))*10)
330 GOSUB 999:NEXT X
340 GOTO 340
400 GOSUB1000:X1=10:Y1=10:X=10:Y
=10
410 HLINE(50,170)-(50,10),PSET
420 HLINE-(X+40,Y),PSET:X=X+58:H
LINE-(X+40,Y),PSET
430 Y=180-Y:IF X<580 THEN 420
440 GOTO 440
500 GOSUB1000:X=10:Y=10
510 HLINE(50,170)-(50,10),PSET
520 HLINE-(X+40,Y),PSET:X=X+29:H
LINE-(X+40,Y),PSET
530 HLINE-(X+40,170),PSET:X=X+29
:HLINE-(X+40,170),PSET
535 Y=Y+17
540 IF X<580 THEN 520
550 GOTO 550
600 GOSUB1000
610 B=0:I=.5:X=10
620 HLINE(50,85)-(50,10),PSET
630 HLINE-(X+40,10),PSET:X=X+B:H
LINE-(X+40,10),PSET
640 HLINE-(X+40,85),PSET:B=B+I:X
=X+B:HLINE-(X+40,85),PSET
650 B=B+I:IF X<550 THEN 630
660 B=23.5:I=.5:X=10
670 HLINE(50,175)-(50,105),PSET
680 HLINE-(X+40,105),PSET:X=X+B:
HLINE-(X+40,105),PSET
682 HLINE-(X+40,175),PSET:B=B-I:
X=X+B:HLINE-(X+40,175),PSET
684 B=B-I:IF X<573 THEN 680
690 GOTO 690
700 GOSUB1000
705 X=10
710 HLINE(50,170)-(50,170),PSET
715 Y=RND(170)+10
720 HLINE-(X+40,Y),PSET
730 X=X+RND(20):HLINE-(X+40,Y),P
SET
740 IF X<580 THEN 715
750 GOTO 750
999 HLINE(X1+40,Y1)-(X+40,Y),PSE
T:X1=X:Y1=Y::RETURN
1000 HLINE(40,10)-(40,180),PSET:
HLINE-(630,180),PSET:'HLINE(40,9
5)-(630,95),PSET
1010 RETURN
2000 POKE&HFFD8,0
2010 STOP
```

Listing # 2 - Basic program to load in Machine Language Program.

```
10 '*************************
20 '*    CoCo Sound Demo    *
30 '* By Kraig Brockschmidt *
40 '* 14506 NE 37th Pl. F-8 *
50 '*Bellevue, WA 98007-3486*
60 '*  For CoCo Clipboard   *
70 '*************************
80 CLEAR200,&H7000
90 CLS:PRINT"POKING PROGRAM INTO
 MEMORY..."
100 FORT=&H7000 TO &H7302:READ A
$:POKET,VAL("&H"+A$):NEXT
110 INPUT"PRESS ENTER TO SAVE TO
 DISK";A$
120 SAVEM"SOUND.BIN",&H7000,&H73
02,&H7100
130 PRINT"PROGRAM IN MEMORY AND
SAVED TO DISK.":STOP
140 DATA 20,20,20,20,20,43,4F,43
,4F,20,53,4F,55,4E,44,20,44,45,4
D,4F,4E,53,54,52,41
150 DATA 54,49,4F,4E,0D,41,53,53
,45,4D,42,4C,59,20,4C,49,4E,45,2
D,4B,52,41,49,47,20
160 DATA 42,52,4F,43,4B,53,43,48
,4D,49,44,54,0D,0D,31,29,20,20,4
6,49,52,53,54,20,34
170 DATA 34,30,48,5A,20,53,51,55
,41,52,45,20,57,41,56,45,0D,32,2
9,20,20,53,45,43,4F
180 DATA 4E,44,20,34,34,30,48,5A
,20,53,51,55,41,52,45,20,57,41,5
6,45,0D,33,29,20,20
190 DATA 44,49,4D,49,4E,49,53,48
,49,4E,47,20,53,51,55,41,52,45,2
0,57,41,56,45,0D,34
200 DATA 29,20,20,46,41,4C,4C,49
,4E,47,20,53,49,52,45,4E,0D,35,2
9,20,20,52,49,53,49
210 DATA 4E,47,20,53,49,52,45,4E
,0D,36,29,20,20,4E,4F,49,53,45,0
D,37,29,20,20,43,4F
220 DATA 4D,42,49,4E,41,54,49,4F
,4E,20,53,4F,55,4E,44,0D,38,29,2
0,20,50,55,52,45,20
230 DATA 53,49,4E,45,20,57,41,56
,45,0D,0D,45,4E,54,45,52,20,59,4
F,55,52,20,43,48,4F
240 DATA 49,43,45,3A,20,00,1A,50
,7F,FF,40,6D,8D,00,EA,26,18,63,8
D,00,E4,5F,30,8D,00
250 DATA E0,33,8D,01,DC,A6,84,80
,20,A7,80,A7,C2,5C,2A,F5,17,00,4
2,30,8D,FE,D6,A6,80
260 DATA 27,06,AD,9F,A0,02,20,F6
,BD,A1,B1,81,31,25,F9,81,38,22,F
5,AD,9F,A0,02,F6,FF
270 DATA 23,CA,08,F7,FF,23,80,31
,27,23,4A,27,2D,4A,27,37,4A,27,5
2,4A,27,61,4A,10,27
280 DATA 00,6D,4A,10,27,00,77,16
,01,8C,0D,E7,27,03,7E,F6,E0,7E,A
9,28,86,FF,B7,FF,20
290 DATA C6,C4,5A,26,FD,43,20,F5
```

continued on 39

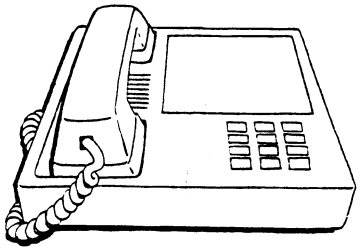
continued from 38

```
,86,FF,C6,62,B7,FF,20,5A,26,FA,4
3,20,F5,86,FF,B7,FF
300 DATA 20,C6,FF,5A,26,FD,7F,FF
,20,C6,AA,5A,26,FD,4A,26,ED,8E,2
5,00,30,82,26,FC,20
310 DATA E2,86,FF,C6,01,B7,FF,20
,34,04,5A,26,FD,35,04,43,5C,20,F
2,86,FF,5F,B7,FF,20
320 DATA 34,04,5A,26,FD,35,04,43
,5A,20,F2,8E,00,00,A6,80,B7,FF,2
0,E6,80,5A,26,FD,20
330 DATA F4,CC,07,23,B7,FF,20,34
,04,5A,26,FD,35,04,1E,89,43,8B,0
3,50,C0,04,20,EC,00
340 DATA 20,20,20,20,20,20,20,20
,20,20,21,21,21,21,22,22,22,23,2
3,23,24,24,24,25,25
350 DATA 26,26,27,27,28,28,29,29
,2A,2B,2B,2C,2C,2D,2E,2E,2F,30,3
0,31,32,32,33,34,35
360 DATA 35,36,37,38,38,39,3A,3B
,3C,3C,3D,3E,3F,3F,40,40,41,42,4
3,43,44,45,46,47,47
370 DATA 48,49,4A,4A,4B,4C,4D,4D
,4E,4F,4F,50,51,51,52,53,53,54,5
4,59,56,56,57,57,58
380 DATA 58,59,59,5A,5A,5B,5B,5B
,5C,5C,5C,5D,5D,5D,5E,5E,5E,5E,5
F,5F,5F,5F,5F,5F,5F
390 DATA 5F,5F,5F,,,,,,,,,,,,,,,
,,,,,,,
400 DATA 00,,,,,,,,,,,,,,,,,,,,,
,,,
410 DATA 00,,,,,,,,,,,,,,,,,,,,,
,,,
420 DATA 00,,,,,,,,,,,,,,,,,,,,,
,,,
430 DATA 00,,,,,,,,,,,,,,,,,,,,,
,,,
440 DATA 00,,,,,,30,8D,FF,7C,5F,
A6,85,B7,FF,20,CB,01,20,F7,00
```


```
00010 ************************
00020 *    CoCo Sound Demo    *
00030 * By Kraig Brockschmidt *
00040 * 14506 NE 37th Pl. F-8 *
00050 *Bellevue, WA 98007-3486*
00060 *  For CoCo Clipboard   *
00070 ************************
00080
00090 LFCR    EQU     13      *Line feed/carriage return.
00100 CHROUT  EQU     $A002   *Print Character ROM call.
00110 CHRCUR  EQU     $A1B1   *ROM call to do cursor and wait for a key.
00120 HRCLS   EQU     $F6E0   *40/80 column screen CLS.
00130 LRCLS   EQU     $A928   *32 column screen CLS.
00140 PORT    EQU     $FF20   *Audio sound port.
00150 ENABLE  EQU     $FF23   *6-bit sound enable/disable port.
00160
00170         ORG     $7000   *Start at $7000.  Program is relocatable.
00180
00190 *Text for menu screen.  This is all simply read and fed to
00200 *CHROUT.  A LFCR forces a new line with CHROUT so we don't need
00210 *to pay attention to line lengths or anything.
00220 MENUTX FCC      '     COCO SOUND DEMONSTRATION'
00230         FCB     LFCR
00240         FCC     'ASSEMBLY LINE-KRAIG BROCKSCHMIDT'
00250         FCB     LFCR
00260         FCB     LFCR
00270         FCC     '1)  FIRST 440HZ SQUARE WAVE'
00280         FCB     LFCR
00290         FCC     '2)  SECOND 440HZ SQUARE WAVE'
00300         FCB     LFCR
00310         FCC     '3)  DIMINISHING SQUARE WAVE'
00320         FCB     LFCR
00330         FCC     '4)  FALLING SIREN'
00340         FCB     LFCR
00350         FCC     '5)  RISING SIREN'
00360         FCB     LFCR
00370         FCC     '6)  NOISE'
00380         FCB     LFCR
00390         FCC     '7)  COMBINATION SOUND'
00400         FCB     LFCR
00410         FCC     '8)  PURE SINE WAVE'
00420         FCB     LFCR
00430         FCB     LFCR
00440         FCC     'ENTER YOUR CHOICE: '
00450         FCB     0       *End of menu text.
00460
00470 START   ORCC    #$50    *Turn off IRQ and FIRQ for better sound quality.
00480         CLR     $FF40   *Turn off disk motor.  Since FIRQ is
00490                         *disabled it won't turn off by itself.
00500
00510 *Next 12 lines finish generating the sine wave waveform at SINE, if
00520 *needed.  This is done so one does not have to type in the full wave.
00530 *If you change the waveform, set FLAG to FCB $FF.
00540 MAIN    TST     FLAG,PCR    *Check if already changed.
00550         BNE     MENUST      *Skip changing if so.
00560         COM     FLAG,PCR    *Toggle flag.
00570         CLRB                *Set counter to zero.
00580         LEAX    SINE,PCR    *Start of waveform to copy from.
00590         LEAU    WAVE,PCR    *End of waveform to copy to.
00600 MAKWAV  LDA     ,X      *Get a value.
00610         SUBA    #$20    *Change values $20-$5F to 0-$3F.
00620         LSLA            *Shift left twice to make 0-$FF in
00630         LSLA            *steps of 4, i.e. upper 6 bits.
00640         STA     ,X+     *Store changed value back.
00650         STA     ,-U     *Store at end of waveform.
00660         INCB            *Increment counter.
00670         BPL     MAKWAV  *Counting to $80=counting until negative.
00680
```

```
00690 *Print the menu and get a selection.
00700 MENUST LBSR    CLS      *Clear the screen, CoCo I, II, or III.
00710        LEAX    MENUTX,PCR  *Get menu text pointer.
00720 MENU   LDA     ,X+      *Get a character.
00730        BEQ     DONE     *Brif end of text.
00740        JSR     [CHROUT]    *Print character in A.
00750        BRA     MENU     *Continue.
00760 DONE   JSR     CHRCUR   *Do cursor and wait for input.
00770        CMPA    #$31     *Is it lower than a '1'?
00780        BLO     DONE     *Brif so.
00790        CMPA    #$38     *Greater than '8'?
00800        BHI     DONE     *Brif so
00810        JSR     [CHROUT]    *Print the character '1'-'8'.
00820
00830 *Enable sound and branch to the selected demonstration sound.
00840        LDB     ENABLE   *Get value in $FF23
00850        ORB     #8       *Turn on 6-Bit sound capability.
00860        STB     ENABLE   *Store it back.
00870 DOSOUN SUBA    #$31     *Convert ASCII 1-8 to numerical 0-7.
00880        BEQ     SQUAR1   *0=First 440Hz square wave.
00890        DECA             *Easiest way to check if A=1.
00900        BEQ     SQUAR2   *If so, the DECA made 0, so do 2nd sq. wave.
00910        DECA             *Check for A=2.
00920        BEQ     DIMSQR   *If so, do diminishing square wave.
00930        DECA
00940        BEQ     SIRFAL   *A=3, falling siren.
00950        DECA
00960        BEQ     SIRRIS   *A=4, rising siren.
00970        DECA
00980        LBEQ    NOISE    *A=5, noise.
00990        DECA
01000        LBEQ    COMBO    *A=6, combination sound.
01010        LBRA    WAVE     *A must be 7, so do pure sine wave.
01020
01030 *CLS function for any text screen.
01040 CLS    TST     $E7      *Check screen mode: 0=32 col, 1=40, 2=80.
01050        BEQ     CLS+7    *Do 32 column CLS if needed.
01060        JMP     HRCLS    *Hi-res text screen CLS.
01070        JMP     LRCLS    *32 Col CLS.
01080
01090 *First method of producing a 440Hz square wave.
01100 SQUAR1 LDA     #$FF     *Byte to store: maximum volume.
01110        STA     PORT     *Send to sound port.
01120        LDB     #$C4     *440 Hz delay for this particular code.
01130 DL0    DECB             *Wait for a while.
01140        BNE     DL0      *Brif not done waiting.
01150        COMA             *If A=$FF then A=0 else A=$FF.
01160        BRA     SQUAR1+2    *Continue with square wave.
01170
01180 *Second method of producing a 440Hz square wave, this time
01190 *Storing A to $FF20 each loop.
01200 SQUAR2 LDA     #$FF     *Maximum volume.
01210        LDB     #$62     *$C4.DIV.2 makes 440 Hz in this case
01220        BRN     FILLER   *The BRN and NOP instructions provide
01230 FILLER NOP              *5 CPU cycles of filler to make both square
01240                         *waves the same.
01250 DL1    STA     PORT     *Send to sound port.
01260        DECB             *Decrement counter.
01270        BNE     DL1      *If not 0, store A to $FF20 again.
01280        COMA             *If A=$FF then A=0 else A=$FF.
01290        BRA     SQUAR2+2    *Continue with square wave.
01300
01310 *Diminishing square wave; sounds a little like a bell.
01320 DIMSQR LDA     #$FF     *Start with maximum volume again.
01330        STA     PORT     *Send to sound port.
01340        LDB     #$FF     *Some delay, partially determines frequency.
01350 DL2    DECB             *Wait a while.
01360        BNE     DL2      *Keep waiting if needed.
01370        CLR     PORT     *Put speaker back to zero.
01380        LDB     #$AA     *Another delay, other factor in frequency.
01390 DL3    DECB             *Wait again.
01400        BNE     DL3
01410        DECA             *Decrease volume one-quarter step.
01420        BNE     DIMSQR+2    *Not zero yet, continue sound.
01430        LDX     #$2500   *Otherwise, wait a bit before doing it again.
01440 WAIT   LEAX    ,-X      *Pause.
01450        BNE     WAIT
01460        BRA     DIMSQR   *Repeat sound.
01470
01480 *Falling siren characteristic of several arcade games for the CoCo.
01490 SIRFAL LDA     #$FF     *Maximum volume.
01500        LDB     #1       *Start with minimum initial delay.
01510 FALL1  STA     PORT     *Send to sound port.
01520        PSHS    B        *Save B.
01530 DL4    DECB             *Delay.  For small B, frequency is higher.
01540        BNE     DL4      *For large B, frequency is lower.
01550        PULS    B        *Restore B.
01560        COMA             *Reverse volume to produce square wave.
01570        INCB             *Increase delay which lowers the frequency.
01580        BRA     FALL1    *Continue sound.  B will roll over continually.
01590
01600 *Rising siren. Very similar to falling siren except frequency
01610 *increases.
01620 SIRRIS LDA     #$FF     *Maximum volume.
01630        CLRB             *Start with maximum delay.
01640 RISE1  STA     PORT     *Send to sound port.
01650        PSHS    B        *Save B.
01660 DL6    DECB             *Delay.  As before, smaller B means
01670        BNE     DL6      *higher frequency.
01680        PULS    B        *Restore B.
01690        COMA             *Reverse volume for square wave.
01700        DECB             *Decrease delay to increase frequency.
01710        BRA     RISE1    *Continue to produce sound.
01720
01730 *Use everything from memory as volume and delay parameters to
01740 *produce random noise.
01750 NOISE  LDX     #0       *Start at memory address 0.
01760        LDA     ,X+      *Get a volume.
01770        STA     PORT     *Send to sound port.
01780        LDB     ,X+      *Get a delay.
01790 DL8    DECB             *Wait.
01800        BNE     DL8
01810        BRA     NOISE+3  *Continue.
01820
01830 *Combination sound:  made by an arbitrary square wave with
01840 *volume and delay parameters changing in jumps.
01850 COMBO  LDD     #$0723   *Arbitrary value (actually my wedding day).
01860        STA     PORT     *Send to sound port.
01870        PSHS    B        *Save B.
01880 DL9    DECB             *Wait.
01890        BNE     DL9
01900        PULS    B        *Restore B.
01910        EXG     A,B      *Do a few strange operations.
01920        COMA             *Just make sure that both A and B do not
01930        ADDA    #3       *converge to some constant value or no
01940        NEGB             *sound results.
01950        SUBB    #4
01960        BRA     COMBO+3  *Continue sound.
01970
01980 *Following 256 bytes defines the waveform. This data will be changed
01990 *To be values from $00 to $FF instead of $20 to $5F as it is
02000 *initially.  If FLAG is initially set to anything besides 0,
02010 *then no change is made to the waveform, allowing one to change
02020 *it at will.
02030 FLAG   FCB     0        *Flag to mark need to change data.
02040 SINE   FCC     @           !!!!""""####$$$%%&&''(())*++,,-../@
02050        FCC     #001223455667889:;<<=>??#
02060        FCC     #@@ABCCDEFGGHIJJKLMMNOOPQQRSSTTYVVWWXXYYZZ#
02070        FCC     #[[[\\\]]]#
02080        FDB     $5E5E
02090        FDB     $5E5E
```

```
02100          FDB     $5F5F
02110          FDB     $5F5F
02120          FDB     $5F5F
02130          FDB     $5F5F
02140          FDB     $5F5F
02150          RMB     128     *Space to mirror waveform int.
02160
02170 *Sine wave sound produced from the table of volume parameters with
02180 *the only delay being the time it takes to get the next parameter.
02190 *Since the loop WAVER is so short, only multiples of 230.1 Hz can
02200 *be obtained.  If we had a longer loop, a smaller multiple can
02210 *be used to give a greater variety of frequencies.
02220 WAVE     LEAX    SINE+128,PCR   *Get pointer to center of waveform.
02230          CLRB                   *Clear index.
02240 WAVER    LDA     B,X            *Load volume from table.  Note that B is
02250          STA     PORT           *taken as two's complement (-128 to 127).
02260          ADDB    #1             *Add some value to index.  Frequency here is
02270          BRA     WAVER          *230.1 Hz * number added above.
02280          END     START          *End
```

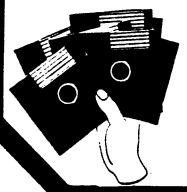


Just One Call Does It ALL

(716) 679-0126 to place your order for Clipboard, ClipDisk, Baseball Hat '89 and Back Issues.

Monday thru Saturday - 9am to 9pm Easterm

"Glad You Asked . . ."



Robert Gault

(Editors Note: We are extremely please and fortunate to have Robert Gault contributing to CoCo Clipboard Magazine. Many of you who subscribed to 'Hot CoCo' will recognize his name as author of a highly detailed graphics article called "Dawn At Crossing Gate." Still others will remember his article called "The Final Fix."

Bobs expertise runs from hardware to software and programming under RS-DOS, OS9, BASIC09, C and BASIC. We're sure you'll appreciate his insights and we encourgae you to write to Mr. Gault here at the Magazine.)

One of the oldest and most infamous bugs of the CoCo disk system is the headbanger bug. An unusual manifestation of this bug was mentioned by Paul Anderson in the Jan/ Feb, 1989 issue of CoCo Clipboard Magazine. Paul found that occasionally double sided drives read the wrong track after switching sides.

Both of the above problems are caused by a lack of foresight when RSDOS was written. Tandy did not anticipate the use of double sided drives (not to mention 6ms 40 or 80 track drives and head solenoids.) The disk operating system at startup, assumes that the drive heads are located at track 0. Tandy DOS 1.0 and 1.1 (and probably most off brand DOSes) do not test where the drive heads are located. Generally the heads are at track 17 but hardly ever at track 0. At the first disk access, the head is sent flying across the disk to bang into the inner drive stop. This happens because RSDOS issues an incorrect SEEK command to the drive controller. RSDOS can recover from this error (even though your drive may not) by causing the head to reset to track 0 after an I/O error. At this point, the softwear and the head are synchronized and usually all is well.

The DOS keeps a record at $97E to $981 of the track number for each of four drives. Each time the head is moved, the image in RAM is updated. This works well with single sided drives but not with double sided drives. As Paul said, if a double sided drive is used, RSDOS will assume that there are two separate heads, where actually there is only one. The front and back heads are mounted on a single support and move as one. If you read track 0 front side and then read track 34 back side, $97E still points to track 0 which is not where the front drive head is located. It is also at track 34. The next use of the front side of the drive will make you want to curse Tandy and Microsoft.

Paul suggested using a two byte drive table as a solution. But this would not be compatible with single sided drives. Since we must modify RSDOS to solve the problem, I suggest that DSKCON be changed as follows: every time a new drive is accessed, DOS should send a READ ADDRESS command to the drive controller. If this is done, it is impossible to lose track of the drive head location. As a side benefit this will allow head settling if you are using head load solenoids.

To implement this change with complete success it will be necessary to patch RSDOS and produce an EPROM for your drive controller. We need to do several things with our new code. Create a new variable in low RAM which the system can use to store the number of the last drive used. This variable should be set to $FF at turn on as this will force an address check at the first drive access. We must compare this value against the current drive at each drive access and take appropriate action. And we must update the new variable as needed.

Below is the code required. I will leave it to the reader to decide where the actual patch and variable are to be located. Be advised that this patch is intended for the RSDOS 1.1. If you have DOS 1.0 or an off brand DOS, you will probably need to do some dissassembling.

```
                ORG     $F3         suggested location
LASTDRIVE       RMB     1           unused by the system

                ORG     $C0E4       END OF COLD START
                JMP     COLDPATCH

                ORG     $D899       DSKCON JUMP TABLE
                FDB     REASEC      READ SECTOR PATCH
                FDB     WRISEC      WRITE SECTOR PATCH

                ORG     $DF60       suggested location
                                    END OF DOS
REASEC          LDA     #$80        CODE FOR READ
                BRA     CONT
WRISEC          LDA     #$A0        CODE FOR WRITE
CONT            PSHS    A
                LDX     #$97E       DRIVE TRACK TABLE
                LDB     <$EB        CURRENT DRIVE #
```

continued on 44

continued from 43

```
            CMPB    <LASTDRIVE                      * N
            LBEQ    $D804     = RETURN TO ROM       * E
            STB     <LASTDRIVE SAVE IT              * W
            LDA     #$C0      GET ADDR              *
            STA     $FF48     COMMAND               * C
            JSR     $D7D1     WAIT UNTIL READY      * O
            LDA     $FF4A     GET TRACK             * D
            STA     $FF49     SET TRACK             * E
            LDX     #$97E
            ABX
            JMP     $D80A     RETURN TO ROM

COLDPATCH   LDA     #$FF
            STA     <LASTDRIVE MAKE FIRST DRIVE
                               ACCESSED A NEW DRIVE
            JMP     $A0E2     RETURN TO ROM
```

I have been using this patch in EPROM form for over a year with no problems. The only adverse effect that I have noticed is a slight hesitation as a disk is verified after a DSKINI operation. Note that any CoCo that is operating in the all RAM mode can use this patch without being EPROMed.

Super Controller Modification

How do you make a good disk controller better? Several companies have done it by designing "no-halt" controllers. I am the happy owner of one of these, the Super Controller II by CRC Computers Inc.

A sacrifice was made in the design of this unit. You no longer have the option of three selectable DOSes as with the older Super Controller I.

How dou you make a good disk controller better? By giving the S.C.II the extra DOS option. It is as simple as drilling a hole in the controllers case to mount a switch and wire-wrapping three wires.

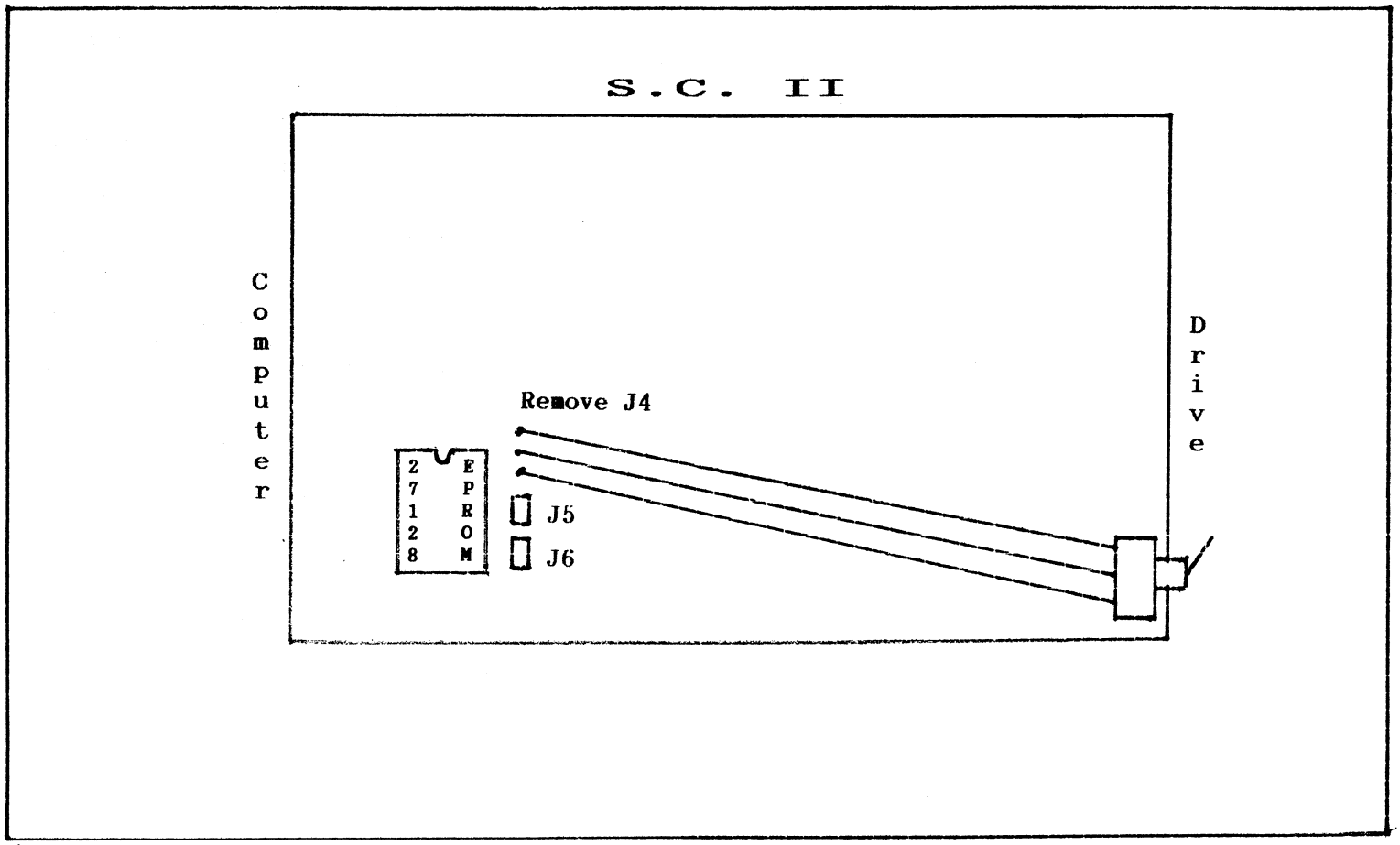
You will need the following items:

1. Any small single pole double throw switch

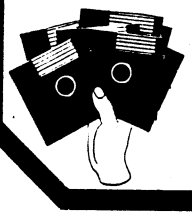
2. Wire wrap wire (prefered)

With the parts installed as shown in the diagram, and with a properly programed 27128 EPROM, the switch selects the top or bottom bank of the EPROM. Your DOS can be in the lower bank and Tandy's RSDOS 1.1 in the upper bank. To change DOSes, flip the switch and do a hard reset (ALT+CTRL+Reset simultaneously) or turn the computer off/on.

Where do you get the EPROM? Well, hmmm,.... CRC does sell an EPROM programmer board for their controller.



"C" and Basic 09



Bob van der Poel

Last issue we discussed interfacing assembly language routines with your Basic09 programs. This month we'll look at the marrying of routines written in the 'C' language with Basic09.

If you are lucky enough to have the 'C' compiler package from Radio Shack you'll have seen the instructions in the appendix on how to interface 'C' with Basic09. They make it all sound so simple . . . and once you get on the problems, it is. But first, why would we even want to write a function in 'C'? Most of the time it probably won't be necessary; however here's a real world application: Adding long integers to Basic09's numeric types.

Basic09 knows about three different types of numbers:

Byte:	unsigned integers in the range 0 to 255,
Integers:	integer numbers from -32768 to 32767,
Reals:	floating point numbers.

As nice as this may seem, there are limitations. Sometimes we need to calculate EXACT numbers outside the range of integers, and we can't use reals since numbers are constantly rounded to enable the number to fit in 5 bytes of storage. 'C' partially solves this problem with a data type called a long integer. Long integers store values in 4 bytes of memory and can represent numbers from -2,147,483,648 to 2,147,483,647 -- and of course, no rounding occurs when long integers are manipulated.

All we need to do then is write a 'C' function (actually a program) which will permit us to communicate values between 'C' and Basic09 and permit 'C' to manipulate the variables. The 'C' program at the end of this article is the result of my labors. I'm not a 'C' expert and am still not sure if I really like the language, but the code works.

The following functions have been implemented: conversion of ASCII strings to longs ("#"), conversion of longs to ASCII strings ("$"), conversion of Basic09 integers to longs ("="), addition ("+"), subtraction ("-"), multiplication ("*"), division ("/") and modulus ("%"). The Basic09 program "ltest" lets you test the code and shows how the various calls are made.

The first thing the Basic09 program does is define a new variable type. LONG is defined as an array of 4 bytes. We could just as easily have used an array of 2 integers or a string, so long as the resultant variable uses 4 bytes of storage. Next we allocate storage for 3 LONG variables: v1, v2 and result. Note the allocation of the variable cmem. This is a working area for 'C' to use.

Ltest asks for 2 values, it converts the strings to 'C' longs, asks for the operation, calls the 'C' function to do the operation, converts the result back to an ASCII string, and finally prints the string. Each time we call the function "longs" we must pass four or five variables:

Cmem:	a 32 byte working area for the 'C'function,
Operation:	a 1 character string representing the operation,
	variable1 (and variable2): either strings or longs,
result:	either a string or long variable.

If you don't pass the correct number of variables, or if the variables are not the correct size "longs" will generate an error 56 (parameter error). If you attempt an operation which "longs" does not know about an error 48 (unimplemented routine) will be returned. You can, if you wish, set up Basic 09 error trapping to deal with these.

The 'C' program presented some real problems in developing -- mainly because our program does not have a "main" and as a result the code which sets up variables (cstart) is not included in the assembly. Also, since the final code is a subroutine, no static variables are permitted. I have used the program "CC" and an enhanced 'C' library (both available on Compuserve) to compile the program. It may require changes for other libraries, however this month's issue of ClipDisk contains the source and compiled code.

If you examine the 'C' source code you will see that we have had to revert to

"C" and Basic 09 continued on 46

"C" and Basic 09 continued from 45

assembler code for two sections. First, we have a "ldy 6,s" instruction. This sets up the Y register for 'C'. Second, the function "seterr" is written in assembler. This function sets the B register and the carry flag so that Basic09 can recognize errors (when Basic09 returns from a RUN the carry flag is checked, if it is set the value in B is used as an error code). The rest of the code is well commented -- if you want to learn some 'C' wade through it.

If you wish to write me you can now use either of two addresses:

P.O. Box 57
Wynndel, B.C.
Canada V0B 2N0

or

P.O. Box 355
Porthill, ID
USA 83853


```
\v+Listing 1, 'C' source code for "longs"\v-

#include <stdlib.h>

/* This is a Basic09 subroutine package to add long integers
   to Basic09.

   Note: In the following examples "Cmemory" is a storage area
         assigned by Basic09. It is needed by  Longs...Failure
         to define Cmemory will cause stack crashes.

   The defined type LONG must be created in your Basic09
   programs with the statement:

          TYPE LONG=LongDummy(4):byte

   Now you can create a long variable with

          DIM a,b,c:LONG

   The following functions are supported:

   + add Var1 and Var2, result in Var3
         RUN (Cmemory,"+",v1,v2,v3)

   - subtract Var2 from Var1, result in Var3
         RUN (Cmemory,"-",v1,v2,v3)

   * multiply Var1 and Var2, result in Var3
         RUN (Cmemory,"*",v1,v2,v3)

   / divide Var1 by Var2, result in Var3
         RUN (Cmemory,"/",v1,v2,v3)

   % return modulus of Var1 % Var2, result in Var3
         RUN (Cmemory,"%",v1,v2,v3)

   = assigns Basic09 integer to long
        RUN (Cmemory,"=",v1,integervalue)

   # converts basic09 string to long
        RUN (Cmemory,"#","1234"+chr$(0),v1)

   $ converts long to string
        RUN (Cmemory,"$",v1,string)


*/


/* these 2 defs are needed for the linker, normally they
   appear in cstart--but we don't link that code...*/

int errno;
double _flacc;

long tonum();

longs (count,cmemory,msize,code,s1,var1,size1,var2,size2,result,size3)
```

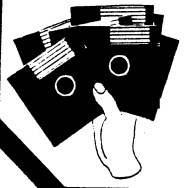
```
/* these are all global variables */

int count,s1,msize,size1,size2,size3;
char *code,*cmemory;
long *var1,*var2,*result;



#asm
 ldy 6,s get static memory address
#endasm

int error=0;

if (msize<20)
  error=53; /* stack routine overflow */

else

/* check for correct params for math functions */

if (*code=='+'  *code=='-'  *code=='*'
              *code=='/'  *code=='%')
   if ( count !=5  size1 != 4  size2 != 4  size3 !=4 )
    error = 56;


if (error == 0)

 switch(*code)
  case '+':
    *result=*var1+*var2;
    break;

  case'-':
    *result=*var1-*var2;
    break;

  case'/':
    *result=*var1 / *var2;
    break;

  case'*':
    *result=*var1 * *var2;
    break;

  case'%':
    *result=*var1 % *var2;
    break;

  case'=':
    if ( count !=4  size1 != 4  size2 != 2 )
      error=56;
    else
      setint(var1,var2);
    break;
 case'#':
    if ( count !=4  size2 != 4 )
      error=56;
    else
      *var2=tonum(var1);
    break;

  case'$':
    if ( count !=4  size1 != 4  size2 <= 14 )
      error=56;
    else
     tostr(var1,var2);
    break;

  default:
    error=48; /* unimplemented routine */



seterr(error); /* return error status to Basic09 */



/*******************************************************/


/* convert the basic09 ascii string to a long,
   actually this is a standard routine (atol) but
   the one in the library uses initialized data
   so must be avoided in this package. */


long tonum(s)
char    *s;

    int     sflag = 1;
    long    value=0;

    while (*s==' ') ++s;
    if (*s == '+'  *s == '-')
        if (*s++ == '-') sflag = -1;
    while (*s > 47 && *s < 58) value = 10 * value + (*s++ - 48);
    return value * sflag;


/* convert the basic09 'long' to an ascii string
   actually this is a standard routine (ltoa), but
   we can't use the one in the library since it
   requires the use of initialized data.  */

tostr(value, string)
long    *value;
char    *string;


 char *ptr; /* needed to save start for reverse() */
 int sflag=0;
 ptr=string;
 *ptr++ = 0xff; /* set basic09 terminator */
 if (*value < 0)
  if (*value == -2147483648) return
    strcpy(string, "-2147483648\\xff");
  sflag = '-';
  *value = -*value;
```

```
    do
     *(ptr++) = *value  % 10 + 48;
       while ((*value /= 10) != 0);

    *ptr++ = sflag; /* add sign */
    *ptr   = 0;
    reverse(string);


/* this routine assigns the basic09 integer variable
   (var2) to the 'C' long (var1)
*/

setint(var1,var2)
long *var1;
int *var2;


 *var1=*var2;


/* Set the CC and B registers for returning to Basic09 */

seterr(error)
 int error;

#asm
 ldd 4,s fetch error value
 tstb if 0
 beq noerr then no error
 coma else set error flag
 bra errx
noerr
 clra no error
errx
#endasm
```

\v+Listing 2, Basic09 Demo Program\v-

```
PROCEDURE ltest
 0000
 0001       TYPE LONG=dlong(4):BYTE
 0011       DIM cmem:STRING[32]
 001D       DIM v1,v2,result:LONG
 002E       DIM a$:STRING[32]
 003A
 003B       PRINT
 003D       PRINT "Long integer 'C' interface demo."
 0061
 0062       LOOP
 0064         PRINT
 0066         INPUT "Enter 1st number: ",a$
 0080       EXITIF a$="" THEN  \\ ENDEXIT
 0090
 0091         (* 1st convert the ASCII string to a long
 00BA         RUN longs(cmem,"$",a$,v1)
 00D2
 00D3         INPUT "       2nd number: ",a$
 00ED
 00EE         (* now convert 2nd string to long
 010F         RUN longs(cmem,"$",a$,v2)
 0127
 0128         INPUT "operation (+,-,*,/ or %): ",a$
 014A
 014B         (* do the requested operation in A$
 016E         RUN longs(cmem,a$,v1,v2,result)
 018C
 018D         (* now convert the result to ASCII so we can print it
 01C2         RUN longs(cmem,"$",result,a$)
 01DA
 01DB         PRINT "Result: "; a$
 01EB
 01EC       ENDLOOP
```

Master Basic 09



Bill Brady

In this column, I will try to introduce you to the concept of 'resources', and how separate storage of variable resources yields great flexibility in customization of programs and user satisfaction. We will also peek into Wiz Pro equates and prostuff files in order to help you do your own customization of Pro. If you are interested in more detail in this area, drop me a note via the Clipboard.

I have a scanner. (a device to 'scan' documents and objects, converting their image to a bit-mapped graphics file). The software that came with this scanner assumes that the scanner itself is hooked up to the modem port. I have a modem there, but am not using the printer port, so that is where I plugged in the scanner. The program allows a switch to the printer port, but forgets the switch each time I exit. So, each time I use the scanner I go through this idiot cycle of trying to scan... waiting.... getting the message 'scanner not responding'.. then remembering to make the switch... etc.

Most good new programs don't have this limited memory. Todays programming practice is tending towards a strict segregation between code and data. 'Don't it always?' you say. But I mean ALL data, including even simple variables.

For example, to send a message "Printer Not Ready" including the line:

```
PRINT "Printer Not Ready"
```

This usually takes care of it. The problem with this is that if you want to change the message to something else, or perhaps get rid of it altogether, you must edit the program source code.

"What's wrong with that?" you say. Well, as long as you are the only one who uses the program, nothing. But suppose someday you decide to commercialize the program, go shareware with it, or just upload it to the services to let others see and use your work. Now, in order to make the change, you must 1: upload a new version of the program, or 2: upload a 'patch', (Ugh). But this is just to change a message! A new version of the program? No!

What usually happens is that you just leave it be. So the message "Printer Not Ready" appears when the Disk is full. So what do you do instead? Take a look at this approach:

First, in your main program, DIM the variable:

```
PROCEDURE main_program
DIM msg(4):STRING
.....
```

Next you create an 'equates' program:

```
PROCEDURE equates
PARAM msg(4):STRING
.....
```

Note that in the equates program, the variable msg is a parameter. main_program creates and holds the space for it, but does not initialize it. Now we add the actual message:

```
PROCEDURE equates
PARAM msg(4):STRING
.......
msg(1)="Printer Not Ready"
......
```

But how does msg(1) get initialized in main_program? Here is how:

```
PROCEDURE main_program
DIM msg(4):STRING
DIM proc:STRING
.....
proc="equates"
RUN proc
KILL proc
.....
(* here is where we actually print it
PRINT msg(1)
......
```

At this point, when we return from equates, the variable msg(1) is set to "Printer Not Ready".

To change "Printer Not Ready" to "Disk full!", we change the procedure equates, not main_program. But so what? We are still changing code right? Yes, but not the main_program, and there's more, we also make equates save the variables and re-load them each time like so:

Master Basic 09 continued on 50

Master Basic 09 continued from 49

```
PROCEDURE equates
PARAM msg(4):STRING
.......
msg(1)="Printer Not Ready"
......
(* First see if we have a 'stuff' file
(* if we do, use it, if not go 1000
(* yup, we replace msg(1).
ON ERROR GOTO 1000
OPEN #dpath,"/dd/stuff":read
GET #dpath,msg
END

(* No 'stuff file? Make one
1000 en=ERR
IF en=215 OR en=216 then
CREATE #dpath,"/dd/stuff":UPDATE
PUT #dpath,msg
END
```

Why write out the variables? First consider what these variables may be: Screen codes, Window types, Fonts, path names, Operating modes, Macro strings etc. By writing them out to a file, you give the user the ability to change them in one session, and have the changes carry through to future sessions. But more: you can write a program that reads in the 'stuff' file, and modifies it directly, thus making major customizations to the program without changing a single line of code, *and* you can revert to the 'original' at any time by simply deleting the 'stuff' file!

Wiz Professional does just this. It creates a file called 'prostuff' in the directory /DD/COM/PRO. Each time Pro is run, it tries to read the prostuff file, if it cannot, it creates it. Almost every variable used by Pro is kept in this file. By reading in the prostuff file, changing variables, and writing it back out, you can customize Pro.

Below, I have given an example. Install_Sterm is a basic09 procedure that, (naturally), installs Sterm, (Mark Griffiths terminal program that includes the CIS-B protocol), right in Wiz Professionals menu. If you look at the code carefully, you will be able to make your own customizations. (Of course it helps if you have the Wiz Pro Unabridged manual).


```
PROCEDURE install_sterm
(* a proc to install Sterm in Wiz Pro
(* created 10/16/88 by Bill Brady
ON ERROR GOTO 100
TYPE wpaths=sp,spa,wpa,dpa,ppa:BYTE; siop,piop:STRING[3];
nextproc,ho,rxfile,host:STRING; spd(32),oldesc(32),newdesc(32):BYTE
DIM paths:wpaths
(* **
TYPE
calls=ststat,gtstat,cursr,screen,reed,wrt,wrtln,creat,sleep,pid,setpri,pag,bau,t
yp,pau,alf,ech,eor,endof,qut,intrpt,opt,bsp,bso,bse:BYTE
DIM caw:calls
(* **
TYPE
flg=acia6551,TEXT,gmode,roll,bufroll,chdfl,vt52,vtansi,OLOPEN,spopen,WPAOPEN,SPA
OPEN,PSP,BLDF,conf,pfl,dfl,exit,ex,dup,mu:BOOLEAN
DIM flags:flg
(* **
TYPE con=esc,CR,lf,BS,BL,home,cls,tnum,lbrk,BK:STRING[1]; IAm,Mkey,Estep:BYTE
DIM cntrl:con
(* **
TYPE commands=dwend,dwsel,owend,defcolor,ulon,uloff,pal,hdds:STRING[2];
DLpos,PRpos,ATpos,br,bufnum,group,revoff,rev,prop,propoff,bold,boldoff:STRING[3]
; four,palette:STRING[4]
DIM cmds:commands
(* **
TYPE buffers=buf(10240):BYTE; klin:STRING[256]; abuf(256):STRING[1];
mac(2):STRING[80]; bsiz,absiz,in:INTEGER
DIM bufs:buffers
(* **
TYPE windows=flat1,flat,stat,bottom:STRING[9]; vt80,vt40,gwin,main:STRING[10];
menu:STRING[63]; mesw:STRING[42]; tmenu,tmes:STRING[15]; ask,prmpt:STRING[9]
DIM win:windows
(* **
TYPE
PRNeq=forePRN,backPRN,menuforePRN,menubackPRN,otherforePRN,otherbackPRN,curPRN,m
enucurPRN,othercurPRN:STRING[1]
DIM PRNs:PRNeq
(* **
TYPE
cnos=fore,back,curs,menufore,menuback,menucur,otherfore,otherback,othercur:BYTE
DIM colors:cnos
(* **
TYPE fx=one,two,GRP,BFN:BYTE
DIM font:fx
(* **
TYPE regs=ccode,a,b,dp:BYTE; x,y,u:INTEGER
```

Burke & Burke

. . . has relocated to Renton, Washington. We pledge to continue to offer Color Computer owners the high quality, affordable, and innovative products that have built our reputation.

To order, use our NEW TOLL FREE ORDER HOTLINE: 1-800-237-2409

Real BASIC for OS9!

There is nothing wrong with your Color Computer. Do not attempt to adjust it.

Only $39.95

```
OS-9 LEVEL TWO VR. 02.00.00
COPYRIGHT 1986 BY
MICROWARE SYSTEMS CORP.
LICENSED TO TANDY CORP.
ALL RIGHTS RESERVED

July 11, 1988  14:37:30

Shell

OS9: xmode /w5 type=0
OS9: Inlz /w5
OS9: rsb <>>>/w5 &
&007
```

CLEAR

```
COPR. 1988 BURKE & BURKE
DISK EXTENDED COLOR BASIC 2.1
COPR. 1982, 1986 BY TANDY
UNDER LICENSE FROM MICROSOFT
AND MICROWARE SYSTEMS CORP.

OK
LOAD "DEMO"
OK
LIST
10 PMODE 4:SCREEN 1,1
20  X=RND(256)-1:Y=RND(192)-1
30  A=RND(256-X)-1:B=RND(192-Y)-1
40  LINE (X,Y)-(X+A,Y+B),PSET,BF
```

Burke & Burke's new R.S.B. program gives you a complete, OS9-compatible version of Disk Extended Color BASIC. We've added new software for OS9-style graphics, sound, printer, and disk I/O. The BASIC you know and love is now running under Level 2 OS9 windows!

R.S.B. loads and saves files using OS9's file format, so we've also included utilities to transfer BASIC programs and data files betwen OS9 and BASIC disks. Of course, you can't use R.S.B. to run machine language programs, and some BASIC commands work slightly differently under R.S.B.

Your BASIC programs can take full advantage of great OS9 features like hard disks, no-halt floppies, multi-tasking, and 2 MHz operation.

R.S.B. requires a CoCo 3 with at least 128K RAM (512K recommended), a floppy controller with either Tandy Disk Extended Color BASIC or DISTO CoCo 3 CDOS, and Level 2 OS9.

R.S.B. Version 1.2 $39.95

CoCo-XT Hard Disk Interfaces

We've sold hundreds of our affordable, high-performance hard disk interfaces to Color Computer enthusiasts worldwide!

Each includes a durable, fully enclosed metal housing, 100 page user manual, and software for use with OS9. The CoCo XT-RTC adds a battery-powered real time clock / calendar for OS9 and BASIC.

- **A true "NO HALT" hard disk system**
- **Controls 1 or 2 hard drives, which may be different sizes**
- **Full ECC / CRC error correction**
- **Average access 30% faster than SASI systems**
- **Uses PC-type hard disk drives & controllers**
- **Full 5 Meg to 120 Meg storage per hard drive**
- **Does not use or disable interrupts**
- **Compatible with most RS-232 interfaces**
- **20 Meg system cost: under $450**
- **Requires Multi-PAK or "electric" Y-cable**
- **EZGen Boot File Editor software included with each interface**
- **Use with HYPER-I/O to share your hard disk between BASIC and OS9**

Buy a hard disk kit and a case/power supply from the PC dealer of your choice. Plug them into the CoCo XT, plug the CoCo XT into your Multi-PAK, and install the OS9 or BASIC software. Presto!

***CoCo XT* $69.95**
***CoCo XT-RTC* $99.95**

Handyman's note: A hard disk kit includes a hard drive, cable set, and Western Digital, DTC, Adaptec, or equivalent PC-compatible hard disk controller.

The Profesional Touch: XT-ROM 2.3

Install XT-ROM in your CoCo XT hard disk controller's BIOS ROM socket. It automatically boots and reboots OS9 from your hard disk.

XT-ROM $19.95

Select among any of two different hard disk boot files, two different floppy boot files, or your BASIC ROM at power-up. XT-ROM gives your system that "professional touch". Great for unattended BBS, home security, or other fail-safe CoCo applications.

OS9 Utilities

Wild & MV Version 2.1

Use "wildcards" with most OS9 commands, or rearrange your directory tree. Features recursive directory searches. A hard disk must! $19.95

EZGen Version 1.06

Powerful OS9 bootfile editor. Change module names, add or delete modules, patch bytes, or rearrange modules. Works on other files, too. $19.95

Hardware, or What?

68B09E 2MHz Microprocessor	$14.95
4' Hard Disk Cable Set	$17.50
Blank 27128 EPROM (for HYPER-I/O)	$9.95
Hard Disk BIOS Socket Installed	$7.50

HYPER-I/O — *Now BASIC runs hard drives, big floppies, and more!*

HYPER-I/O modifies the Disk BASIC in your CoCo 1, 2, or 3 to provide a "Dynamic Disk Interface". Use your existing BASIC and M/L software with hard disk interfaces (CoCo XT, DISTO, LR), RAM Disks, and any mix of floppy drives from 160K to 720K each. Fully RESET protected, user configurable, expandable, EPROM- able HYPER-I/O V2.6 is the most versatile hard / floppy disk DOS available. ***Please specify HYPER-I/O, DISTO HYPER-I/O, or LR HYPER-I/O when ordering.***

$29.95

HYPER-III (Adds RAM Disk and Print Spooler to CoCo 3 HYPER-I/O. ***$12.95***

HYPER-I/O & HYPER-III work with your B&B, RGB, LR, or DISTO Hard Disk

HYPER-I/O Utilities

by Kevin Berner

The HYPER-I/O HARD DISK UTILITIES let you perform wildcard copy, delete, and search operations on your HYPER-I/O directories. Great timesaver for moving data from floppy disk to hard disk, or for BBS maintenance. Kevin's DISK DOCTOR will lock out bad sectors on your hard or floppy disks, and includes a disk-zap utility designed specifically for use with HYPER-I/O.

DISK Doctor $17.95 HYPER-I/O Hard Disk Utilities $21.95

WOW! Both Great Utility Packages $37.95

PERTASCII $19.95

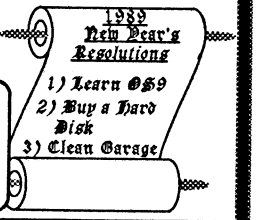
PERTASCII is a multi-user word game for Level 2 OS9. The players are yourself, the computer, other users on your system, or even friends that call in on a modem.

The game is played in timed rounds, until a certain score is reached. Players can join or leave the game at the beginning of any round. The players make words from random letters during each 3 minute round, and then argue over whether or not to accept each other's words.

Great for BBS and multi-user systems . . . or play practice rounds against the computer to hone your skills!

Includes a user-expandable 3500 word dictionary.

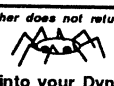
512K CoCo 3, Level 2 OS9, and one disk drive required. CoCo 2 and 128K owners: watch for our 128K / 64K version!



Daggorpatch

Yet another does not return!

Don't be afraid of the dungeons . . .



Only $9.95!

DAGGORPATCH puts the thrill back into your Dyna Micro Dungeons of Daggorath™ game cartridge by patching it to run from disk. Includes disk load & save, auto-repeat command, pause, DMP-100 screen dump, tape-to-disk, and more!

Burke & Burke
P.O. Box 58342 Renton, WA 98058
(206) 235-0917

WASHINGTON RESIDENTS PLEASE ADD 7% SALES TAX. COD's add $2.20. Shipping (within the USA) $2.50 per CoCo XT; $2.00 per disk or ROM. Please allow 2 weeks for delivery (overnight delivery also available for in-stock items).

Telephone orders call (800) 237-2409.
Technical support (206) 235-0917.

```
DIM s:regs
(* **
TYPE strings=ramf,nofont,nod,mdmclr,ddir,crtc,aktc,pnr:STRING; ontime:STRING[8];
mdmdly,hh,mm:INTEGER; te2:REAL
DIM strg:strings
(* **
TYPE portset=bb,pp,ss,ll,sbaud,stype:BYTE; baud(7):STRING[4]; bcmd(7):BYTE;
PAR(5):STRING[4]; SB(2):STRING[3]; word(2):STRING[1];
pcmd(5),scmd(2),wcmd(2),ppmax:BYTE; typcmd(2,3,2):BYTE
DIM port:portset
TYPE menuentry=parm:STRING; sho:STRING[16]; PRO:STRING[12];
pa,funct,trig1,trig2:BYTE; trig3(2):BYTE
DIM mnu(26):menuentry
1 (* DIM Local Vars here
DIM char:STRING[1]
DIM epa:BYTE
10 (* proc start
ON ERROR GOTO 100
PRINT CHR$($0C)
PRINT " This program opens the file /DD/COM/PRO/ProStuff and installs the STerm"
PRINT " in WizPro as an XPROG. It installs STerm in slot 20 and shows <s>Term in
the"
PRINT " Menu. It instructs Wiz Pro to pass the param '/T2' to STerm at runtime."
PRINT
PRINT " If you would also like Wiz Pro to xmode T2 at autolog time, edit your
CIS"
PRINT " autolog file to include your xmode command as follows: \$xmode /t2
baud=nn"
PRINT " etc. You may have more than one shell command in your autolog file,
using"
PRINT " \$ to preface each.  Tap enter when you are ready to proceed."
INPUT "    ",a$
20 (* go do it
OPEN #epa,"/dd/com/pro/prostuff":UPDATE
SEEK #epa,0
GET #epa,paths \ GET #epa,caw \ GET #epa,flags \ GET #epa,cntrl
GET #epa,cmds \ GET #epa,win \ GET #epa,PRNs
GET #epa,colors \ GET #epa,font \ GET #epa,strg
GET #epa,port \ GET #epa,mnu
IF mnu(20).sho>"" THEN
PRINT " Slot 20 already contains : "
PRINT
PRINT USING "'Sho ',s20,' Proc ',s20,' type
',i4",mnu(20).sho,mnu(20).PRO,mnu(20).pa
PRINT
PRINT " Do you want to replace it Y/N" \ INPUT a$
IF a$="N" OR a$="n" THEN 50
ENDIF

(* here is where the actual changes take place
mnu(20).sho="s<T>erm    " \mnu(20).PRO="STerm    "
mnu(20).pa=1 \mnu(20).parm="/t2"

(* sTerm shows in the menu now, and the proc that is
(* run is STerm. The pa is 1 which means this is a
(* stand alone program and the parameter passed to
(* sterm when run will be /t2. All of this is installed
(* in slot 20 which is the "T" key.

SEEK #epa,0
PUT #epa,paths \ PUT #epa,caw \ PUT #epa,flags \ PUT #epa,cntrl
PUT #epa,cmds \ PUT #epa,win \ PUT #epa,PRNs
PUT #epa,colors \ PUT #epa,font \ PUT #epa,strg
PUT #epa,port \ PUT #epa,mnu
PRINT " Sterm Installed"
50 (* done
CLOSE #epa
END
(*
(*
END
100 (* error
en=ERR
IF en=215 OR en=216 THEN PRINT " Unable to open /DD/COM/ProStuff"
ELSE
PRINT " I am unable to install STerm in Wiz pro"
PRINT " The system error is : "; en
ENDIF
ON ERROR GOTO 120
CLOSE #epa
120 END

To help you customize Pro, I have included some of the standard
equates below:

DIM big,mes,mes1:STRING[9]
DIM menwin,point2,point,point1,box,box1,box2:STRING[6]
DIM ch,windTYPE,borderPRN:STRING[1]
DIM dwset,owset,cwarea:STRING[2]
DIM epa:BYTE
DIM pathopen:BOOLEAN
DIM ppd(32):BYTE
DIM phost:STRING

(* start equating
flags.TEXT=TRUE
paths.nextproc="WizAuto"
(* enable text mode & x-off
paths.spd(8)=$FF \paths.spd(9)=$FF \paths.spd(22)=3
FOR i=1 TO 32 \paths.newdesc(i)=0 \NEXT i
pathopen=FALSE
bufs.mac(1)=" No Macro "
bufs.mac(2)=bufs.mac(1)
timeout=1000 \
flags.chdfl=FALSE \flags.roll=FALSE \flags.bufroll=FALSE
flags.gmode=FALSE \flags.vtansi=FALSE
flags.BLDF=FALSE \flags.PSP=FALSE
strg.mdmclr="+=+=+==ATZ"
strg.mdmdly=40
strg.ddir="Current Data Directory "
strg.nod="No modem CD, you cannot receive"
strg.ramf="Proc Missing or My space is full"
strg.nofont="The font is not in memory"
cntrl.Estep=255
cntrl.tnum=CHR$(0)
cntrl.cls=CHR$($0C) \cntrl.home=CHR$(1)
cntrl.CR=CHR$(13) \cntrl.lf=CHR$(10)
cntrl.BS=CHR$(8) \cntrl.BL=CHR$(7)
cntrl.lbrk=CHR$(133)
cntrl.esc=CHR$(27)
cntrl.Mkey=3
cntrl.IAm=0 \cntrl.Estep=0
cmds.hdds="02"
cmds.br="15"
cmds.four=cntrl.CR+cntrl.CR+cntrl.CR+cntrl.CR
strg.aktc=" Any Key to Continue "
strg.crtc=" <ENTER> to Continue"
strg.pnr="   Path not Ready "

(* System Equates
caw.screen=$8C \ \caw.reed=$89 \caw.cursr=$25
caw.ststat=$8E \caw.sleep=$0A \caw.gtstat=$8D
caw.wrt=$8A \caw.wrtln=$8C \caw.creat=$83
caw.pau=8
caw.eor=12 \caw.endof=13 \caw.qut=18 \caw.intrpt=17
caw.opt=0 \caw.bsp=10 \caw.bso=3 \caw.bse=19
caw.alf=6 \caw.bau=22 \caw.typ=21
caw.pag=9 \caw.ech=5
caw.pid=$0C \caw.setpri=$0D
paths.newdesc(caw.eor)=$0D
paths.newdesc(caw.bsp)=8
paths.newdesc(caw.bse)=8
paths.newdesc(caw.bso)=1 \paths.newdesc(caw.alf)=1

2 (* Port Equates

(* set acias6651 to TRUE for RS-232 Pak Hemphill Disto etc
flags.acia6551=TRUE
```

```
port.bb=3 \port.pp=1 \port.ss=1 \port.ll=2
paths.siop="mwp" \paths.piop="p"
paths.rxfile=""
(* 6850 equates NOTE typcmd becomes typcmd(ll,pp,ss)
IF NOT(flags.acia6551) THEN
port.ppmax=3
port.baud(1)=" 300" \port.bcmd(1)=$00 \port.baud(2)=" 600" \port.bcmd(2)=$01
port.baud(3)="1200" \port.bcmd(3)=$02 \port.baud(4)="2400" \port.bcmd(4)=$03
port.baud(5)="4800" \port.bcmd(5)=$04 \port.baud(6)="9600" \port.bcmd(6)=$05
port.baud(7)="19.2" \port.bcmd(7)=$06
port.typcmd(1,1,1)=$15 \port.typcmd(1,1,2)=$15 \port.typcmd(1,2,1)=$0D
\port.typcmd(1,2,2)=5
port.typcmd(1,3,1)=$09 \port.typcmd(1,3,2)=1 \port.typcmd(2,1,1)=$15
\port.typcmd(2,1,2)=$11
port.typcmd(2,2,1)=$1D \port.typcmd(2,2,2)=$15 \port.typcmd(2,3,1)=$19
\port.typcmd(2,3,2)=$15
port.PAR(2)=" ODD" \port.PAR(3)="EVEN" \port.PAR(1)="NONE" \port.PAR(4)="N/A "
\port.PAR(5)="N/A "
port.SB(1)="ONE" \port.SB(2)="TWO"
port.word(1)="7" \port.word(2)="8"
ENDIF

(* S6551 equates
IF flags.acia6551 THEN
port.ppmax=5
port.baud(1)=" 300" \port.bcmd(1)=$01 \port.baud(2)=" 600" \port.bcmd(2)=$02
port.baud(3)="1200" \port.bcmd(3)=$03 \port.baud(4)="2400" \port.bcmd(4)=$04
port.baud(5)="4800" \port.bcmd(5)=$05 \port.baud(6)="9600" \port.bcmd(6)=$06
port.baud(7)="19.2" \port.bcmd(7)=$07 \port.PAR(1)="NONE" \port.pcmd(1)=$00
port.PAR(2)=" ODD" \port.pcmd(2)=$20 \port.PAR(3)="EVEN" \port.pcmd(3)=$60
port.PAR(4)="MARK" \port.pcmd(4)=$A0 \port.PAR(5)="*SPC" \port.pcmd(5)=$E0
port.SB(1)="ONE" \port.scmd(1)=0 \port.SB(2)="TWO" \port.scmd(2)=$80
port.word(1)="7" \port.wcmd(1)=$20 \port.word(2)="8" \port.wcmd(2)=$00
ENDIF

3 (* Color Equates
font.GRP=$C8 \font.BFN=1
font.one=27 \font.two=$3A
colors.fore=63 \colors.back=8 \colors.curs=36
IF flags.TEXT THEN
colors.menufore=0 \colors.menuback=36 \colors.menucur=63
colors.otherfore=0 \colors.otherback=63 \colors.othercur=0
ELSE
colors.menufore=63 \colors.menuback=8 \colors.menucur=63
colors.otherfore=63 \colors.otherback=8 \colors.othercur=63
ENDIF
cmds.defcolor=CHR$(27)+CHR$($30)

4 (* Screen Equates
cntrl.BK=CHR$(15)
cmds.hdds="2"
cmds.owend=cntrl.esc+CHR$($23) \cmds.dwend=cntrl.esc+CHR$(24)
\cmds.pal=cntrl.esc+CHR$($31)
cmds.defcolor=cntrl.esc+CHR$($30) \cmds.dwsel=cntrl.esc+CHR$($21)
cmds.DLpos=CHR$(2)+CHR$(32+16)+"!"
cmds.PRpos=CHR$(2)+" "+"!"
cmds.ATpos=CHR$(2)+CHR$(32+48)+"!"
cmds.rev=CHR$($1F)+CHR$($20)+CHR$(0) \cmds.revoff=CHR$($1F)+CHR$($21)+CHR$(0)
IF flags.TEXT THEN
cmds.bold=CHR$(0)+CHR$(0)+CHR$(0)
cmds.boldoff=cmds.bold
cmds.prop=cmds.bold
cmds.propoff=cmds.bold
ELSE
cmds.bold=cntrl.esc+CHR$($3D)+CHR$(1)
cmds.boldoff=cntrl.esc+CHR$($3D)+CHR$(0) \cmds.prop=cntrl.esc+CHR$($3F)+CHR$(1)
cmds.propoff=cntrl.esc+CHR$($3F)+CHR$(0)
ENDIF
cmds.ulon=CHR$($1F)+CHR$($22) \cmds.uloff=CHR$($1F)+CHR$($23)

5 (* Window Equates
PRNs.forePRN=CHR$(0) \PRNs.backPRN=CHR$(1) \borderPRN=PRNs.backPRN
IF flags.TEXT THEN
windTYPE=CHR$(2)
PRNs.curPRN=CHR$(0) \PRNs.forePRN=CHR$(8)
PRNs.backPRN=CHR$(1)
PRNs.menucurPRN=CHR$(2) \PRNs.menuforePRN=CHR$(10)
PRNs.menubackPRN=CHR$(3)
PRNs.othercurPRN=CHR$(4) \PRNs.otherforePRN=CHR$(12)
PRNs.otherbackPRN=CHR$(5)
ELSE
windTYPE=CHR$(5)
PRNs.curPRN=CHR$(0) \PRNs.forePRN=CHR$(0)
PRNs.backPRN=CHR$(1)
PRNs.menucurPRN=CHR$(1) \PRNs.menuforePRN=CHR$(1)
PRNs.menubackPRN=CHR$(0)
PRNs.othercurPRN=CHR$(0) \PRNs.otherforePRN=CHR$(0)
PRNs.otherbackPRN=CHR$(1)
ENDIF

cmds.owend=cntrl.esc+CHR$($23) \cmds.dwend=cntrl.esc+CHR$($24)
\cmds.dwsel=cntrl.esc+CHR$($21)
dwset=cntrl.esc+CHR$($20) \owset=cntrl.esc+CHR$($22) \cwarea=cntrl.esc+CHR$($25)
win.ask=owset+CHR$(1)+CHR$(18)+CHR$(18)+CHR$(44)+CHR$(3)+PRNs.menucurPRN+PRNs.me
nubackPRN
win.prmpt=owset+CHR$(0)+CHR$(24)+CHR$(19)+CHR$(33)+CHR$(1)+PRNs.curPRN+PRNs.back
PRN
point=cntrl.esc+CHR$($40)+CHR$(0)+CHR$(12)+CHR$(0)+CHR$(4)
point1=cntrl.esc+CHR$($40)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)
point2=cntrl.esc+CHR$($40)+CHR$(0)+CHR$(16)+CHR$(0)+CHR$(5)
box=cntrl.esc+CHR$($49)+CHR$(2)+CHR$($7F)+CHR$(0)+CHR$($BF)
box1=cntrl.esc+CHR$($49)+CHR$(2)+CHR$($69)+CHR$(0)+CHR$($B7)
box2=cntrl.esc+CHR$($49)+CHR$(2)+CHR$($60)+CHR$(0)+CHR$($B4)
big=owset+CHR$(1)+CHR$(22)+CHR$(1)+CHR$(36)+CHR$(18)+PRNs.menucurPRN+PRNs.menuba
ckPRN
IF flags.TEXT THEN
win.flat=owset+CHR$(1)+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(12)+PRNs.otherbackPRN+PRNs.
othercurPRN

win.flat1=owset+CHR$(1)+CHR$(0)+CHR$(12)+CHR$(80)+CHR$(9)+PRNs.curPRN+PRNs.backP
RN

menwin=cwarea+CHR$(2)+CHR$(1)+CHR$(33)+CHR$(16)

win.menu=big+point1+box+point+box1+point1+point2+box2+point1+menwin

mes=owset+CHR$(1)+CHR$(8)+CHR$(2)+CHR$(64)+CHR$(15)+PRNs.othercurPRN+PRNs.otherb
ackPRN

mes1=cwarea+CHR$(2)+CHR$(1)+CHR$(60)+CHR$(13)

win.mesw=mes+point1+box+point+box1+mes1

win.bottom=dwset+CHR$(0)+CHR$(0)+CHR$(23)+CHR$(80)+CHR$(1)+PRNs.otherbackPRN+PRN
s.othercurPRN

win.stat=dwset+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(2)+PRNs.othercurPRN+PRNs.ot
herbackPRN

win.main=dwset+windTYPE+CHR$(0)+CHR$(2)+CHR$(80)+CHR$(21)+PRNs.curPRN+PRNs.backP
RN+borderPRN

ELSE

win.flat=owset+CHR$(1)+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(12)+PRNs.otherbackPRN+PRNs.
otherforePRN

win.flat1=owset+CHR$(1)+CHR$(0)+CHR$(12)+CHR$(80)+CHR$(9)+PRNs.forePRN+PRNs.back
PRN

menwin=cwarea+CHR$(2)+CHR$(1)+CHR$(33)+CHR$(16)

win.menu=big+point1+box+point+box1+point1+point2+box2+point1+menwin

mes=owset+CHR$(1)+CHR$(8)+CHR$(2)+CHR$(64)+CHR$(15)+PRNs.otherforePRN+PRNs.other
backPRN

mes1=cwarea+CHR$(2)+CHR$(1)+CHR$(60)+CHR$(13)
```

Master Basic 09 continued on 55

CoCo 'N Amateur Radio

Mike Dooley KE4PC

This time around let's look at something that's related to Coco's and Amateur Radio, but which doesn't have a radio attached to it... Actually, this is good information for all Coco users.

What I'm talking about is connecting devices to the Coco via the built-in Serial Port or the RS232 ROMPack. If you buy your equipment and cables from Radio Shack, there's usually no problem. When you decide to connect up to that printer or MODEM you bought at the flea market then, sometimes, there is a problem.

Another area I use a home-built cable in is to connect a PC to my Coco for exchanging files. I use a Terminal program on each computer, Mikeyterm on the Coco and PC-Talk on the PC, to transfer files between the two.

The solution to any of these problems is to build your own cable. In this issue we'll look at the connections necessary to build a cable and connect your Coco to most any other Serial RS232 Device.

Although most any connector can be used, most devices using RS232 use a 25 pin connector called a DB25. The connector on the RS232 ROMPack is a typical example. That connector is a DB25S (S for socket), while its mate is a DB25P (P for plug).

There are two types of communications equipment used in the computer world. One is DTE which stands for Data Terminal Equipment. The Coco is a DTE device. The other type is DCE which is Data Communications Equipment. A printer or MODEM is a good example of this type of device. Figure One shows some of the pins on an RS232 connector used for DTE and their function.

Figure 1
DTE Equipment

Pin Number	Signal
1	Frame Ground
2	Transmit Data
3	Receive Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Carrier Detect
20	Data Terminal Ready

Figure Two shows the pinout for a DCE piece of equipment. As you can see, there's no difference between the pins and the names of the signals there. The difference is in the direction of signal flow.

Figure 2
DCE Equipment

Pin Number	Signal
1	Frame Ground
2	Transmit Data
3	Receive Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Carrier Detect
20	Data Terminal Ready

Figure Three shows the direction of signal flow for DTE and Figure Four shows the same for DCE.

Figure 3
DTE Signal flow

Transmit Data - Output
Receive Data - Input
Request to Send - Output
Clear to Send - Input
Data Terminal Ready - Output
Data Set Ready - Input
Data Carrier Detect - Input

Figure 4
DCE Signal Flow

Transmit Data - Input
Receive Data - Output
Request to Send - Input
Clear to Send - Output
Data Terminal Ready - Input
Data Set Ready - Output
Data Carrier Detect - Output

Now comes the fun part... is that piece of flea market equipment a DCE or DTE device? Depending on what it is, and if the

CoCo 'N Amateur Radio continued on 54

```
win.mesw=mes+point+box+point+box1+mes1

win.bottom=dwset+CHR$(0)+CHR$(0)+CHR$(23)+CHR$(80)+CHR$(1)+PRNs.otherbackPRN+PRN
s.otherforePRN

win.stat=dwset+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(2)+PRNs.otherforePRN+PRNs.o
therbackPRN

win.main=dwset+windTYPE+CHR$(0)+CHR$(2)+CHR$(80)+CHR$(21)+PRNs.forePRN+PRNs.back
PRN+borderPRN
ENDIF

win.gwin=dwset+CHR$(7)+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(24)+CHR$(0)+CHR$(2)+CHR$(0)

win.vt80=dwset+CHR$(2)+CHR$(0)+CHR$(0)+CHR$(80)+CHR$(24)+CHR$(1)+CHR$(0)+CHR$(0)

win.vt40=dwset+CHR$(6)+CHR$(0)+CHR$(0)+CHR$(40)+CHR$(24)+CHR$(0)+CHR$(1)+CHR$(2)

win.tmenu=big+menwin
win.tmes=mes+mes1

7 (* Menu Entries
FOR i=1 TO 26
mnu(i).parm=""
mnu(i).sho="" \mnu(i).PRO="" \mnu(i).pa=255
mnu(i).funct=255
mnu(i).trig1=0 \mnu(i).trig2=0 \mnu(i).trig3(1)=0 \mnu(i).trig3(2)=0
NEXT i
mnu(1).sho="<A>utolog      " \mnu(1).PRO="WizAuto" \mnu(1).pa=4
mnu(2).sho="<B>reak Value  " \mnu(2).PRO="WizUtils" \mnu(2).pa=4
mnu(3).sho="<C>onfigure    " \mnu(3).PRO="WizConfig" \mnu(3).pa=4
mnu(4).sho="<D>ownload     " \mnu(4).PRO="WizUtils" \mnu(4).pa=4
mnu(5).sho="<E>nables      " \mnu(5).PRO="WizUtils" \mnu(5).pa=4
mnu(6).sho="<F>ull dux/Echo " \mnu(6).PRO="WizUtils" \mnu(6).pa=4
mnu(7).sho="<G>cal " \mnu(7).PRO="gcal" \mnu(7).pa=2
mnu(8).sho="<H>elp " \mnu(8).PRO="WizHelp" \mnu(8).pa=4
mnu(9).sho="<I>nfo " \mnu(9).PRO="WizInfo" \mnu(9).pa=4

mnu(14).sho="<N>ew Host " \mnu(14).PRO="newhost" \mnu(14).pa=4
mnu(16).sho="<P>rint ON/OFF " \mnu(16).PRO="WizUtils" \mnu(16).pa=4
mnu(17).sho="<Q>uit Wiz " \mnu(17).PRO="WizQuit" \mnu(17).pa=4
mnu(18).sho="<R>eset " \mnu(18).PRO="WizUtils" \mnu(18).pa=4
mnu(19).sho="<S>hell " \mnu(19).PRO="WizShell" \mnu(19).pa=4 \mnu(19).funct=0

mnu(21).sho="<U>pload Text " \mnu(21).PRO="WizSend" \mnu(21).pa=4
mnu(23).sho="<W>orking Dir " \mnu(23).PRO="WizUtils" \mnu(23).pa=4
mnu(24).sho="<X>modem      " \mnu(24).PRO="WPXmod" \mnu(24).pa=4
lode(1)="WizUtils" \lode(2)="WizConfig" \lode(3)="WizClipper"
lode(4)="WpXmod" \lode(5)="WizAuto"
lode(6)="" \lode(7)="" \lode(8)=""

Make a note if the 'lodes' above. Next time I will show you how to 'get around'
the so-called 64k limit by using non-mapped loads and the RUN/KILL statements.
```

CoCo Clipboard Magazine

CoCo 'N Amateur Radio continued from 54

manufacturer followed the RS232 guidelines, the information contained here should help you connect the two together. As a general rule, the connections shown in Figures Five and Six show how a DTE is connected to a DTE and a DTE is connected to a DCE.

Figure 5
DTE to DTE

```
 1 ---------------  1
 2 ---------------  3
 3 ---------------  2
 4 ---------------  5
 5 ---------------  4
 6 ---------------  20
20 ---------------  6
```

Figure 6
DTE to DCE

```
 1 ---------------  1
 2 ---------------  2
 3 ---------------  3
 4 ---------------  4
 5 ---------------  5
 6 ---------------  6
20 ---------------  20
```

We've seen the connections on the DB25 connector so let's remind ourselves of the connections on the Serial Port connector. Figure Seven will show those pins and their designations.

Figure 7
Coco Serial Port

1 - Carrier Detect
2 - Receive Data
3 - Signal Ground
4 - Transmit Data

Figures Eight through fifteen show connections between the Coco RS232 ROMPack, or the Coco Serial Port, and various other devices. The connections to a PC were shown since some (like me) like to exchange data between PC's and their Coco's. Some PC's have a DB25 and others have a DB9 so I included both.

CoCo 'N Amateur Radio continued on 56

CoCo 'N Amateur Radio continued from 55

Figure 8
Coco RS232 to MODEM

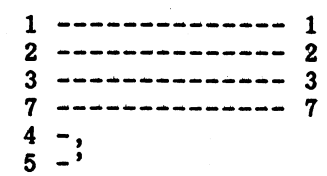


Figure 9
Coco RS232 to PC DB25

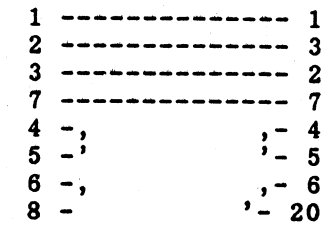


Figure 10
Coco RS232 to PC DB9

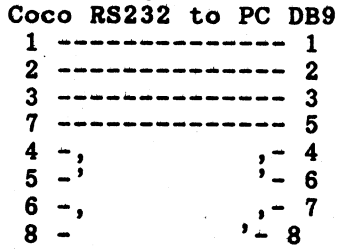


Figure 11
Coco RS232 to DB25 Printer

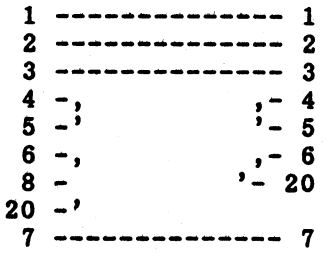


Figure 12
Coco Serial Port to MODEM

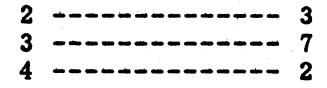


Figure 13
Coco Serial Port to DB25 PC

Figure 14
Coco Serial Port to DB9 PC

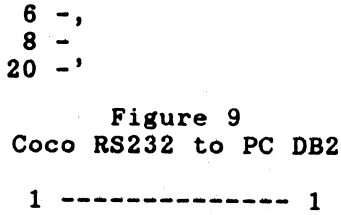


Figure 15
Coco Serial Port to DB25 Printer

If you'd like to contact me with questions or comments there are three ways to do it.

1 - Write in care of Coco Clipboard
2 - Compuserve 73367,632
3 - Home address
Mike Dooley
3801 E 14th #1401
Plano, Tx 75074

73's and thanks... DE KE4PC

OS9 Comments

Shelby LaMont

Editors Note: Last issue Shelby was somewhere in or on the Bay of Fundy in Canada. Now our world wide correspondent has found his way out west. This article was written after a marathon session of watching "Meet Mr. McNutley" video's. Reporting from the Vista Dome of the California Zephyr via radio-phone here is Mr. LaMont.

In our last article, we covered some of the basics of an OS9 module and the different parts of the module. Covering items from the header to the CRC, but how does the assembler know to put those particular items in a particular place in the finished product? It does this by certain LABELS we place in our OS9 assembly language code, labels that can't be used as a 'placement' label in the rest of the code. And it's this information which makes it quite different from RS-DOS assembly language programs.

Quite a few folks started in RS-DOS assembly language before progressing to OS9. EDTASM+ and Disk EDTASM+ (and other commercial assemblers) gave us our first taste of assembly language programming on the Color Computer. Moving on to OS9, we used the module 'ASM' to transform our OS9 assembly language code into modules. Now with OS9 Level 2 and the OS9 Developer's System, we now have an RMA Assembler (RMA stands for Relocatable Macro Assembler), which allows us to develop a library of assembly language routines and include/call them from various different assembly language products.

With RS-DOS assemblers, you began your program with the proverbial ORG statement (ie.. ORG $7000). This was the memory location where your program would reside. If the code was relocatable (using BRAnches instead of JuMPs, etc.), you could easily change the location of where you wanted the program to reside with simple reference in the (C)LOADM command line. With the RS-DOS assembly language program, you could place your DATA information space in the front of the program or the end. As long as you knew where the entry point to the program was, it really didn't matter. The EXEC command would allow you to start the program at that specified entry point. With OS9...it IS different!

Here's a brief skeleton of the structure of an OS9 module:

```
 1)       ifp1
 2)        use .../defs/os9defs.a
 3)       endc
 4)  TYP set  PRGRM+OBJ
 5)  REV set  REENT+1
 6)       MOD
LEN,NAM,TYP,REV,BEG,MEM
 7)  one RMB  1
 8)  two RMB  2
 9)  buf RMB  50
 10) stk RMB  256
 11) MEM equ  *
 12) NAM FCS  /DEMO/
 13) BEG lda  #20
 14) <the rest of the code>
 15)     emod
 16) LEN equ  *
```

Now that you see the structure, we'll briefly explain what each area of this code stands for. The ASM compiler is a two-pass compiler. During the first pass it creates a symbol table, which means it looks at all the labels and EQUates and prepares for assembly. It's during the second pass of the code where it actually does the heavy work.

Lines 1 through 3 are used during the first pass and tells the compiler that "IF it's Pass 1" then include the file in Line 2 (the USE statement tells what file to include). Line 3 says that's the end of the USE files.

Line 4 sets the TYPE of the module. In this case, we tell the assembler that it is a program (PRGRM) and it's an assembly language module (OBJ). Instead of PRGRM, you could have indicated to the assembler that it was a data module, Basic09 I-code, Pascal P-code, or even Cobol I-Code. Module types could have been a Subroutine, data, system, file manager, device descriptor, and other various modules.

Line 5 instructs the assembler that this particular module will be a re-entrant module, and also indicates the edition of the module.

Line 6 is the portion of the code that instructs the module how to set up the header of the module (that we explain in the last issue of the CoCo Clipboard). In the order in which the appear in the MOD directive, LEN is the end of the module, NAM is

OS9 Comments continued on 58

OS9 Comments continued from 57

the name of the module, TYP is the type of module, REV indicates it is a re-entrant module and the edition, BEG tells where the actual program execution begins, and MEM indicates where the END of the data area.

Lines 7 and 8 are simple data storage areas (RMB standing for Reserve Memory Bytes). Lines 9 and 10 and areas for the stack. It is always a good idea (but not required) to include a 256 byte area, as in Line 10, for the stack, unless you know exactly how much stack space the computer is going to use.

Line 11 tells the MOD line, through the use of an EQUate, where the end of the data storage area is.

Line 12 tells the assembler what the name of the module is. As we mentioned in our last issue, the module header will point to an address that contains this name. We use the FCS instruction because it will add a value of 128 to the last character of the name, and the system will know where the end of the module's name is.

Line 13 contains the label BEG, and this indicates the entry point of the module's program code. Line 14 is shorthand for where the rest of the module's code will be located.

Lines 15 and 16 indicate where the end of the module is. The EMOD compiler instruction tells the assembler that this is the End of the MODule. The LEN label is used with the EQUate for the purpose of telling the MOD line where the module's end is, thus providing the module's header with the proper information for the length of the module.

There are some vast differences between the structure of the Level 1 OS9 ASM module structure and those used by the Level 2 OS9 Development System's RMA assembler, but that's a whole different story!

Intro to MIDI

Dr. Lester Hands

If you like music, MIDI provides you with a way to perform even if your keyboard skills are almost nonexistent. Traditionally, music making has been exclusive domain of the very gifted. Developing the skills needed to make music took years to develop. Now with the marriage of computers and synthesizers, that has all changed. Even musicians can benefit from this. Instead of concentrating most of their energy on producing technically correct music, they can now spend more time on musicality.

Everywhere you listen, MIDI is behind the sounds. Most of the sound tracks of TV and radio commercials are made with the help of MIDI. Many professional recording artists are producing albums that rely heavily on MIDI.

MIDI is an acronym, which means that each letter stands for the first letter in a series of words. In this case, it stands for Musical Instrument Digital Interface. MIDI is a means by which musical instruments can interface or connect together. MIDI makes it possible for a computer to communicate with a synthesizer.

All sorts of interesting possibilities come into being with MIDI. Now you can record music from a synthesizer and store it in digital form inside a computer. Once inside a computer, you can do all sorts of interesting things with the musical data. You could display it on the screen in standard notation, print it out, edit it, and even send it back to the synthesizer.

The Color Computer is quite capable of acting as a MIDI controller, provided that you have the right equipment. The first thing you need is a MIDI equipped synthesizer. Not all synthesizers have MIDI built into them. The easiest way to tell is to look at the back of the synthesizer. You should see at least two black sockets labeled MIDI IN and MIDI OUT.

The other thing that you need is a MIDI program and connector for your CoCo. The simplest and least expensive arrangement is to use a "player" program such as Lyra. The CoCo can generate MIDI signals without any assistance through the serial port on the back of the computer. You just need a cable that will connect correctly to the synthesizer. Such a cable consists of merely two wires and can easily be made with parts from your local Radio Shack store.

Load in your program, make sure the cable is correctly connected, and you are ready to go! Your CoCo now can send MIDI data to the synthesizer. The player program (such as Lyra) can send the right MIDI data to produce amazingly good sounding music. You can play music that sounds like a small band, complete with percussion parts. Of course, exactly what sounds you get will depend on what type of synthesizer you have.

Nowadays a reasonably good synthesizer can be had for well under $200. A favorite choice of mine is the Yamaha PSS-480 which sells for as little as $129 if you strike the time right. It has a good MIDI interface and the ability to play 12 simultaneous notes. Each of those notes can be on a separate channel and each can have a different sound.

In this type of a setup, the CoCo is unique in that it is very inexpensive. The only other computer that I know of that has built-in MIDI capability is the Atari ST, which costs a good deal more.

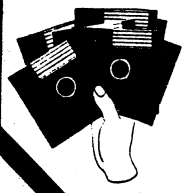
If you want to be able to record music from your synthesizer, then you will need to invest in a special program pack that gives you MIDI IN and MIDI OUT connectors. The CoCo is quite capable of sending MIDI OUT, but MIDI IN is another story. Keeping track of MIDI data that is both coming and going takes a lot of effort on the part of the computer. There just isn't enough time to do it all unless you have the MIDI program pack.

Once you have the MIDI pack, then all sorts of new possibilities open up. You can still use it with the Lyra program to play all your favorite songs. With a good recorder/sequencer program, you can now record what you play on the synthesizer. Once the music is recorded you can edit it, removing or changing bad notes, or fixing timings, or whatever else your imagination comes up with.

The best MIDI recorder currently available for the CoCo is CoCo MIDI 3. It uses special techniques so that it is very efficient when recording. In fact, a CoCo running this program does a better job of recording than an IBM PC/XT (running at 10 megahertz using an MPU-401 and the "Cake-

Midi continued on 62

Invoice Program



Paul Bornemann

The documentation for my Invoice program is very simple because the program is very simple to use. I purposely wrote it to be short, sweet, and at the same time professionally functionary.

With this program you could be a president of a major corporation invoicing an overseas client for products or services rendered, or maybe a paperboy invoicing a subscriber who seems to forget that newspapers cost money. If someone owes you some money don't get mad, send them an invoice.

The very first thing you must do is configure a portion of this program by inserting your company name and address. You can do this by loading "INVOICE/BAS" and EDIT lines 410 and lines 1510 to 1530 using basic. After you have resaved the program with your information in it, you will be ready to begin zapping out your invoices.

Follow with me now while I briefly run through the program.

1) The program prompts you to enter your baud rate by pressing the corresponding number (1-5) on the left side of the screen.

2) Follow the prompt and enter your invoice or reference number. Then enter the invoice date. mm/dd/yy works nicely.

3) The "bill to" party is the company or person you want to send this invoice.

4) Keep your item description brief, 25 or less characters. When entering your dollar amounts do not use commas.

5) Now you have the option of entering up to 3 lines of remarks or comments. For example "Please pay this invoice in 15 days" or you might want to add additional information concerning your reasons for sending this invoice.

6) Finally, you have the option to do another invoice, or address an envelope or simply quit.

Use carbon paper or multistrike paper if you want copies.

I wish to thank Randy Krippner for his article in the CoCo Clipboard Magazine Vol.1 #5 May/June 1988, titled "The Basic Idea" on page 16. It was his idea of the Universal Input Routine that I used several times throughout my program. Nice work, Randy what's next?

Any questions please contact me via Compuserve ID # 75340,513 or write to me Paul E. Bornemann c/o Born-Air Freight, P.O. Box 7567, West Trenton, New Jersey 08628.

Please feel very free to customize any areas of this program, and have fun using it.


```
10 REM **** FILENAME: INVOICE/BA
S
20 REM **** DATE:     AUGUST 22,
 1988
30 CLS
40 REM ****** GREEN SCREEN DISPL
AY
50 POKE359,57:POKE65314,65
60 REM ****** TITLE SCREEN
70 PRINT@42,STRING$(11,36)
80 PRINT@74,"* invoice *"
90 PRINT@106,STRING$(11,36)
100 PRINT@165,"BY: PAUL E. BORNE
MANN"
110 PRINT@197,STRING$(21,131)
120 PRINT@262,"BILL YOUR CUSTOME
RS"
130 PRINT@294,STRING$(19,131)
140 PRINT@388,"press enter to co
ntinue"
150 X$=INKEY$
160 IFX$=""THEN150
170 CLEAR5000
180 CLS
190 REM ****** SET PRINTER BAUD
RATE
200 PRINTSTRING$(32,159)
210 PRINT" SELECT YOUR PRINTER B
AUD RATE"
220 PRINTSTRING$(32,159)
```

Invoice continued on 61

Invoice continued from 60

```
230 PRINT" <1>     600 BAUD - POK
E 150,87
240 PRINT" <2>    1200 BAUD - POK
E 150,41
250 PRINT" <3>    2400 BAUD - POK
E 150,18
260 PRINT" <4>    4800 BAUD - POK
E 150,7
270 PRINT" <5>    9600 BAUD - POK
E 150,1
280 PRINTSTRING$(32,159)
290 PRINT@422,"make your selecti
on"
300 Y$=INKEY$
310 IFY$=""THEN300
320 IFY$="1"THENPOKE150,87
330 IFY$="2"THENPOKE150,41
340 IFY$="3"THENPOKE150,18
350 IFY$="4"THENPOKE150,7
360 IFY$="5"THENPOKE150,1
370 CLS
380 REM ****** PRINT INVOICE HEA
DING
390 PRINT@297,"PRINTING ..."
400 PRINT#-2,TAB(6)STRING$(71,"*
")
410 PRINT#-2,"       TYPE  IN  YO
UR  NAME  OR  COMPANY  NAME  -
ADDRESS  -  ZIP  CODE  HERE "
420 PRINT#-2,TAB(6)STRING$(71,"*
")
430 PRINT#-2,CHR$(10)
440 PRINT#-2,TAB(32)"***** INVOI
CE *****"
450 PRINT#-2,CHR$(10)
460 CLS
470 REM ****** INVOICE NUMBER OR
 REFERENCE NUMBER - AND DATE
480 PRINTSTRING$(32,239)
490 PRINT"  INVOICE OR REFERENCE
 NUMBER       AND DATE IN ANY FO
RMAT
500 PRINT@160,STRING$(32,239)
510 PRINT"invoice number";:INPUT
IN$
520 PRINT:PRINT"invoice date?
";:LINEINPUTDT$
530 PRINT#-2,TAB(50)"INVOICE NO:
 ";IN$
540 PRINT#-2
550 PRINT#-2,TAB(50)"DATE: ";DT$
560 PRINT#-2
570 PRINT#-2,"       BILL TO:",
580 PRINT#-2, CHR$(10)
590 CLS
600 REM ****** BILL TO PARTY - N
AME - ADDRESS - CITY - STATE - Z
IP
610 PRINT@8,"bill to party"
620 REM ****** LO=LOCATION AND L
N=LINE LENGTH (UNIVERSAL INPUT R
OUTINE)
630 LO=69:LN=58
640 PRINT:PRINT"name: "
650 GOSUB1900
660 LO=142:LN=49
670 PRINT:PRINT"street or pob: "
680 GOSUB1900
690 LO=207:LN=48
700 PRINT:PRINT"city/state/zip:
"
710 GOSUB1900
720 PRINT#-2, CHR$(10)
730 LO=293:LN=58
740 PRINT:PRINT:PRINT"attn: "
750 GOSUB1900
760 PRINT#-2,CHR$(10)
770 CLS
780 PRINT@297,"PRINTING ..."
790 PRINT:PRINT
800 PRINT#-2,CHR$(10)
810 PRINT#-2,TAB(6)STRING$(71,"-
")
820 PRINT#-2,TAB(6)"ITEM#";:PRIN
T#-2,TAB(18)"DESCRIPTION";:PRINT
#-2,TAB(65)"AMOUNTS"
830 PRINT#-2,TAB(6)STRING$(71,"-
")
840 CLS
850 REM ****** ITEM DESCRIPTION
AND DOLLAR AMOUNTS
860 PRINT"       BILL UP TO <5> I
TEMS"
870 PRINT"  LIMIT 25 CHARACTERS
PER LINE"
880 PRINTSTRING$(32,159)
890 A$=CHR$(45):B$=CHR$(45):C$=C
HR$(45):D$=CHR$(45):E$=CHR$(45)
900 PRINT"item #1 ";:INPUTA$
910 PRINT"AMOUNT ";:PRINT"$";:IN
PUTA
920 GOSUB1960
930 PRINT:PRINT"item #2 ";:INPUT
B$
940 PRINT"AMOUNT ";:PRINT"$";:IN
PUTB
950 GOSUB1960
960 PRINT:PRINT"item #3 ";:INPUT
C$
970 PRINT"AMOUNT ";:PRINT"$";:IN
PUTC
980 GOSUB1960
990 PRINT:PRINT"item #4 ";:INPUT
D$
1000 PRINT"AMOUNT ";:PRINT"$";:I
NPUTD
1010 GOSUB1960
1020 PRINT:PRINT"item #5 ";:INPU
TE$
1030 PRINT"AMOUNT ";:PRINT"$";:I
NPUTE
1040 PRINT:PRINT"      press ente
r for total"
1050 T$=INKEY$:IFT$=""THEN1050
1060 PRINT#-2,CHR$(10)
1070 PRINT#-2,"       ITEM 1:";:P
RINT#-2,TAB(18)A$,
1080 PRINT#-2,TAB(63)
1090 PRINT#-2,USING"$###,###.##"
;A
1100 PRINT#-2,"       ITEM 2:";:P
RINT#-2,TAB(18)B$,
1110 PRINT#-2,TAB(63)
1120 PRINT#-2,USING"$###,###.##"
;B
1130 PRINT#-2,"       ITEM 3:";:P
RINT#-2,TAB(18)C$,
1140 PRINT#-2,TAB(63)
1150 PRINT#-2,USING"$###,###.##"
;C
1160 PRINT#-2,"       ITEM 4:";:P
RINT#-2,TAB(18)D$,
1170 PRINT#-2,TAB(63)
1180 PRINT#-2,USING"$###,###.##"
;D
1190 PRINT#-2,"       ITEM 5:";:P
RINT#-2,TAB(18)E$,
1200 PRINT#-2,TAB(63)
1210 PRINT#-2,USING"$###,###.##"
;E
1220 PRINT#-2,TAB(63)"==========
="
1230 LETT=(A+B+C+D+E)
1240 PRINT#-2,CHR$(10)
1250 PRINT#-2,TAB(18)"BALANCE DU
E THIS INVOICE:",
1260 PRINT#-2,TAB(63)
1270 PRINT#-2,USING"**$#,###.##"
;T
1280 PRINT#-2,TAB(63)"----------
-"
1290 PRINT#-2,CHR$(10)
1300 CLS
1310 REM ****** ENTER UP TO 3 LI
NES OF REMARKS/COMMENTS ETC.
1320 REM ****** LO=LOCATION AND
LN=LINE LENGTH (UNIVERSAL INPUT
ROUTINE)
1330 PRINT#-2,TAB(6)"< REMARKS >
"
1340 PRINT "ENTER UP TO 3 LINES
OF REMARKS"
1350 PRINT STRING$(30,131)
1360 LO=106:LN=73
1370 PRINT:PRINT"REMARKS:1 "
1380 GOSUB1900
1390 LO=202:LN=73
1400 PRINT:PRINT"REMARKS:2 "
1410 GOSUB1900
1420 LO=298:LN=73
1430 PRINT:PRINT"REMARKS:3 "
1440 GOSUB1900
1450 PRINT#-2,CHR$(10)
1460 PRINT#-2,TAB(6)STRING$(71,"
-")
1470 CLS
1480 PRINT@297,"PRINTING ..."
1490 PRINT#-2,CHR$(10)
1500 PRINT#-2,TAB(17)STRING$(48,
"*")
1510 PRINT#-2,"
   PLEASE REMIT TO: YOUR NAME OR
 COMPANY NAME"
1520 PRINT#-2,"
                    ADDRESS OR P
O BOX"
1530 PRINT#-2,"
                    CITY / STATE
 / ZIP CODE"
1540 PRINT#-2,CHR$(26)
1550 PRINT#-2,"
    IT IS ALWAYS OUR PLEASURE TO
 SERVE YOU!!"
1560 PRINT#-2,TAB(17)STRING$(48,
"*")
1570 FORI=1TO5
1580 PRINT#-2,CHR$(10)
1590 NEXTI
1600 CLS
1610 REM ****** OPTIONS MENU
1620 PRINT"     INVOICE COMPLETED
1630 PRINT"     YOUR OPTIONS ARE:
1640 PRINT
1650 PRINT"     <a> DO ANOTHER IN
VOICE"
1660 PRINT"     <m> ADDRESS ENVEL
OPE"
1670 PRINT"     <q> QUIT"
1680 X$=INKEY$
1690 IFX$=""THEN1680
1700 IFX$="A"THENGOTO30
1710 IFX$="M"THENGOTO1730
1720 IFX$="Q"THENGOTO1880
1730 PRINT:PRINT"     SET ENVELOP
E IN PRINTER
1740 PRINT"     PRESS <ENTER> WHE
N READY
1750 X$=INKEY$
1760 IFX$=""THEN1750
1770 PRINT:PRINT"name: ";:LINEIN
PUTNA$
1780 PRINT"address: ";:LINEINPUT
AD$
```

Invoice continued on 62

Midi continued from 59

walk" sequencer)! It also has special editing features that are normally found only on much larger systems such as the MacIntosh.

A MIDI recorder makes the most sense if you have some keyboard skills. But even if you don't, there are things you can do with it that can't be done with a program like Lyra. Some examples are smooth tempo and volume changes or controlling tempo from the synthesizer (or with a drum machine). It is possible to convert music files from Lyra to CoCo MIDI 3 so you can take advantage of these extra features.

In summary, the Color Computer has some unique advantages for making music with a MIDI synthesizer. It can be done for much less than most other systems (under $400 for the computer and necessary programs) and still is quite capable. This fact is well recognized by recording studios around the country who use Color Computers to produce music. And then there are many thousands of hobbyists who have more fun making music with their trusty CoCo than Heinz has pickles!

CoCo Clipboard Magazine

Invoice continued from 61

```
1790 PRINT"city/state/zip: ";:LI
NEINPUTCS$
1800 PRINT
1810 PRINT"attn ? ";:LINEINPUTAT
$
1820 PRINT#-2,TAB(35)NA$
1830 PRINT#-2,TAB(35)AD$
1840 PRINT#-2,TAB(35)CS$
1850 PRINT#-2
1860 PRINT#-2,TAB(35);:PRINT#-2,
"ATTN: ";AT$
1870 GOTO1600
1880 EXEC40999
1890 REM ******* UNIVERSAL INPUT
 ROUTINE *******
1900 PRINT@LO,STRING$(LN,".");:P
RINT@LO,">";:TM$="":LE=0
1910 AK$=INKEY$:IFAK$=""THEN1910
ELSEIFAK$=CHR$(13)THEN1950
1920 IFAK$=CHR$(8)ANDTM$=""THENG
OTO1910
1930 IFAK$=CHR$(8)ANDLEN(TM$)>0T
HENTM$=LEFT$(TM$,LEN(TM$)-1):LO=
LO-1:LE=LE-1:PRINT@LO,">";:IFLE=
LN-1THEN1910ELSEPRINT@LO+1,".";:
GOTO1910
1940 IFLE=LNTHENGOTO1910ELSEPRIN
T@LO,AK$;:TM$=TM$+AK$:LO=LO+1:LE
=LE+1:IFLE=LNTHENGOTO1910ELSEPRI
NT@LO,">";:GOTO1910
1950 IFLE<LNTHENPRINT@LO,STRING$
(LN-LE,32);:PRINT#-2,TAB(6);TM$;
:PRINT#-2:RETURNELSERETURN
1960 PRINT"     <m> more OR <t>
total"
1970 I$=INKEY$:IFI$=""THEN1970
1980 IFI$="M"THEN RETURN
1990 IFI$="T"THEN GOTO 1040
```

Simply Better Word Processor

**Setting New Standards . . .

**Separate configuration program provides three screens of parameters that can be changed and saved to disk as separate modules (versions). Select a module when program boots.

**Single command creates second interacting word processor in a variable size alternate window on the same screen. Load and work on two different files at the same time. Transfer blocks of text between word processors.

**Automatic print spooling. Simply initiate any of the print commands while using the word processor in one window, and then move to the word processor in the opposite window and work on another file while text is being printed.

**Display five pre-defined print fonts in pre-selected colors.

**Display underlining on the screen.

**Mark up to 79,000 blocks of text in a pre-selected color. Blocks can be saved to disk, output to a printer, copied, moved, erased, or moved to text in another file. (All print fonts and/or underlining are maintained.)

**Automatically create a table of contents for your text.

**Automatically create an alphabetized index for your text.

**Use the Mail-Merge feature to print numerous copies of text. Insert text into each copy from a separate list.

**We do not "print to disk." We went one step further. Print all or any section of text to memory first; make any necessary changes; and then save the formatted listing to disk.

**Place pre-printed forms into your printer, and using the Print Fill feature, fill in only the areas where you want text.

**Help is available instantly in overlay windows.

**Create up to 10 tasks (macros). Each task can be saved to disk for use later. Build a library of frequently used tasks.

**Compatible with RGB Computer Systems hard drive.

**If desired, program will automatically save your text to disk if no key is pressed for 5 minutes.

**Sort sections of text. All print fonts and/or underlining are maintained in the sort.

**Print all text or blocks of text using serial or parallel output.

**Has a type ahead buffer, key beep, alert beep, and error beep.

**Hearing impaired mode flashes the screen's border if an alert or an error occurs.

**Headers, footers, and auxilliary lines can contain print fonts and/or underlining while maintaining whatever print font and/or underlining is active in the body of your text.

**Insert the current page number anywhere inside your text. Place it inside headers, footers, or auxilliary lines to automatically number pages.

**All print fonts and/or underlining are maintained regardless of where you begin printing.

**Set the default justification to left, right, center, or both left and right. Change individual lines to left, right, or center.

**Provides over 90k of text storage with 128k systems, and over 480k of storage with 512k systems.

**Separate mode allows you to view your text as nearly as possible to how it will be printed on paper.

**Use the Go To Page Number feature to quickly move to the beginning of a specific printed page number.

**The Get Directory command appends a formatted directory listing to your text. Easily keep a record of your text files.

**Easily set or reset up to 250 individual tab stops.

**Uses the 40 and 80 column screens for super fast display.

Included is a concise and complete 138 page spiral bound manual for your ***Simply Better Word Processor.***

Setting New Prices . . . ~~$79.95~~

Special Intro Price . . . $29.95 *(add $3 S/H)*

EXAMINED BY CLIPBOARD

When It Comes To Word Processing, We're . . . "Simply Better"

MasterCard VISA

Simply Better Software *P.O. Box 20726* *Portland, OR 97220*	*All Orders,* *Call . . . 1-800-248-8420*	*Technical Assistance* *9 AM - 5 PM* *(503) 254-7225*

Painless OS9

Randy Krippner

What a computer does can be broken down into three basic steps: Getting data for processing (Input), processing that data, and returning the processed data to the user or storing it for later use (Output). That's the topic for today, I/O.

Under RS-DOS, the Coco is limited as to where input may come from or where output may go to. Input is limited to the keyboard or data stored in a disk file, depending on the whims of the programmer. Output is limited to the screen, a disk file, or the printer. Again, depending on what the programmer has decided the program should do.

But what if your input isn't a format that's useful to you? Suppose you have a 1500 name mailing list stored on disk, and the only sorting program you have wants the data items to be typed in? What if you want to take that printed report from your database program and stick it in a disk file so you can merge it into a document you're writing? Sorry, Charlie. No can do.

But with OS9 things are considerably different.

OS9 doesn't care where the data comes from or where it is sent to. To OS9, all I/O devices look the same. It treats them all as if they were just simple data files. It works as a sort of pipeline between devices, sending a stream of data to, or getting a stream of data from, any device hooked to the computer.

It can do this because of special OS9 modules known as device drivers. OS9 sends the data to the specified device driver, and that's it. It's up to the device driver to do whatever is necessary to store or fetch the data in question. We, the users, and OS9, don't have to worry about data format, special control codes, timing or anything else. The device drivers do all of that for us.

This means that we are no longer restricted to the whims of the programmer. Input and output can usually be redirected anywhere you want. And if you add a different device to your computer, such as a hard drive or a RAM disk, all you have to do to get OS9 to use it is to add the necessary device driver.

Let's take a look. Boot up OS9. After the OS9: prompt appears, type this:

```
mdir <ENTER>
```

The MDIR command shows you the module directory. This is a listing of OS9 modules and device drivers that are already in your computer's memory. You should see something like this:

```
IOMan       RBF      D0
D1          TERM     VDGInt
W           W1       W2
PRINTER     P        T1
```

This abbreviated listing of my module directory shows some of the device drivers and descriptors loaded into my computer's memory. What is shown when you type MDIR will differ a great deal from this, but these should appear with the other files in your module directory.

It's not necessary to understand exactly how device drivers and descriptors work. Just think of them as a sort of translator. They take the data sent to them by OS9 and translate it into a form the device can understand or, if the device is sending data to OS9, the process is reversed and it translates the data sent by the device to a format OS9 can deal with.

OS9 sends and receives data along input and output paths that lead to or from various files, programs or devices. (Remember pathlists from last time?)

There are three standard paths under OS9.

1. Standard Input Path: This is path # 0. It routes data from the keyboard to programs.

2. Standard Output Path: This is path #1. It routes data from the program to your display screen.

3. Standard Error Output Path: This is path #2. It routes routine messages such as prompts and error messages to your display screen.

Painless OS9 continued on 65

Painless OS9 continued from 64

Where these paths lead can be redirected. The less than symbol (<) redirects the standard input path. The greater than symbol (>) redirects the standard output path. The standard error output path is redirected with two greater than symbols (>>).

By using these symbols, you can get your input from, or send your output to, just about any device or file you wish.

Before we look at how to do that, we have to do some setting up. Since this is a column about I/O, this is as good a place as any to give you a few tidbits about how to customize OS9 to work with various I/O devices.

OS9 defaults to a printer baud rate of 600, which just won't do. So we'll change that first. But before we get started, make a backup copy of your boot disk just in case something goes wrong.

To change the baud rate, we'll use a command called XMODE. This displays or changes the parameters of any sequential access device, such as the terminal or printer. The format for using XMODE is this:

```
xmode devname [paramlist]
```

The device we want to change is the printer, so our "devname" is going to be "/p". We want to change baud rate, so the "paramlist" will be "baud=nn", where "nn" is the code for the desired baud rate. The codes are listed on page 6-103 of the System Command Descriptions section of your documentation. Since I run my printer at 2400 baud, I'd use 4. Here's the entire command, as I'd type it for my 2400 baud printer:

```
xmode /p baud=4 <ENTER>
```

This change is not permanent. It will remain in effect only as long as you have your computer turned on. In order to make this change permanent, one more step is required. But before we do that, let's make a change to two other devices while we're at it, drives 0 and 1.

Every disk drive manufactured in the past five years or so can run at a 6ms step rate, but Tandy insists on setting the step rate to 25 or 30 ms. Because OS9 uses the drives so often, upping the step rate can improve the performance of the system enormously. So here's how to change the step rate of your drives to 6ms.

We're going to build a text file called PATCH, containing the information needed to patch the D0 and D1 device descriptors so a 6ms step rate is used. We'll use an OS9 utility called BUILD, which takes anything you type and stores it in a disk file. First type this line:

```
build patch <ENTER>
```

After you do this, the drive will whir, a file called PATCH will be created and a ? will pop up on your screen telling you it's ready for you to start typing. Type in the following six lines, pressing ENTER at the end of each line. After typing in the last line, press ENTER alone. This will close the file and return you to the OS9: prompt.

```
l d0
c 14 00 03
v
l d1
c 14 00 03
v
```

The first character in lines 1 and 4 is an "L", not a 1.

If you type DIR when the OS9: prompt returns, you should see a file called PATCH stored on the disk. Now we're going to use another OS9 utility called MODPATCH. Type this line:

```
modpatch patch <ENTER>
```

After a few seconds the OS9: prompt will return. If you get any error messages, delete the PATCH file and use BUILD to type it in again, being careful not to make any mistakes. If you are running only one disk drive, you should not type in lines 4 through 6.

After you do this, you should notice a big improvement in speed whenever OS9 has to load something from disk.

This patch, like baud rate change, is temporary. In order to make the changes permanent, we need to use another utility, COBBLER. After using Xmode and Modpatch, type this:

```
cobbler <ENTER>
```

Cobbler will create a new OS9 boot file with the same modules currently in memory. The changes that were made with Modpatch and Xmode are now permanent.

If you get any error messages or your disk will not boot after using cobbler, make a fresh backup of your OS9 master disk, then go through this process again. Cobbler can be finicky sometimes, so make sure you always use backup disks when fooling around with it.

Now that this is out of the way, it's time to have a little fun and take a look at some simple examples of I/O redirection.

If you do a DIR of your boot disk, you should see a file called START UP on the disk. Type this:

```
list START UP <ENTER>
```

You should see something like this appear on your screen:

```
*Echo welcome message
```

Painless OS9 continued on 66

Painless OS9 continued from 65

```
echo *Welcome to OS-9 LEVEL 2*
echo *on the Color Computer 3*
*Lock shell and std utils into memory
link shell
*Start system time from keyboard
setime </1
date t
```

START UP is a special text file called a Procedure file. It contains commands that are automatically executed by OS9 whenever OS9 is booted up. We'll go into more detail about procedure files in a later column.

If you have a printer, make sure you've used Xmode to set the baud rate properly, then type this line:

```
list START UP >/p <ENTER>
```

The List utility uses the standard output path to send information to the screen display. But here we used the standard output redirection symbol, >, to route the output to the printer, /p, instead of the screen. You could, if you wished, route the output of List into another disk file.

```
list START UP >/d0/test_file <ENTER>
```

This would cause OS9 to create a new disk file, called test_file, and send the listing of the START UP file into it.

Remember, just about anything can be rerouted simply by using the proper redirection indicators. Try this:

```
dir /d0/cmds >/p <ENTER>
```

This will print the directory of the CMDS subdirectory instead of listing it on the screen.

My word counter tells me I'm out of room for this month. Until next time, play around with I/O redirection. Remember, you can't hurt anything if you make a mistake. The worst that can happen is that you may ruin a file or scramble a disk, and even then, all you have to do is make another backup of your master and start again.

In the next issue we'll be looking at I/O redirection again, this time putting it together with OS9's multi-tasking capabilities to do some very interesting things. If, like many Coco 3 owners, you have an old Coco 2 laying around (or any computer with an RS-232 port), you may want to dust it off, make up a null-modem cable for it, and pick up a term program. I'll show you how you can turn your Coco 3, with OS9 L2, into a real multi-user computer system with nothing but a cable, a public domain terminal program, and a Coco 2 or other computer with an RS-232 port. It's so simple you won't believe it.

Questions, comments, spare change, proposals of marriage, etc. should be addressed to Randy Krippner, 1014 W. Hwy. 114, Lot 29, Hilbert, WI 54129. Please include an SASE if you wish a reply.

CoCo Clipboard Magazine is proud to be associated with the "Successful Magazine Publishers Group". This voluntary organization assists magazine publishers in a wide range of areas from specialized accounting systems to production techniques and magazine distribution. Our participation in S.M.P.G. will help us produce a better magazine for you the reader and once again demonstrates our commitment to the Color Computer market.

At last you can show everybody you know about ***Clipboard*** with the ***CoCo Clipboard Magazine*** baseball cap!

This is a summer weight cap in red with a white panel and our now famous *". . . because no one is a novice forever"* slogan on the front!

One size fits all and it's made in the U.S.A.!

The price is $8.00 and includes shipping anywhere in the U.S.A. or Canada. Phone orders accepted and can be charged (10% service charge) on VISA, MasterCard or Discover or send your check or money order to:

CoCo Clipboard Magazine
Baseball Hat '89
3742 U.S. 20 - Box 3
Fredonia, NY 14063
(716) 679-0126

Allow 6-8 weeks for delivery.

CoCo Clipboard Magazine
3742 U.S. 20 Box 3, • Fredonia, N.Y. 14063

BULK MAIL
U.S. POSTAGE
PAID
PERMIT NO. 18
FREDONIA, N.Y.
14063