

the world of 68' micros

Supporting Tandy Color Computer Disk BASIC, CoCo OS-9, and OS-9/68000



MONSTER
 HARD DRIVES (Part II)
for your CoCo (OS-9)
 page 4

The Chicago
 CoCoFest!
 Observations from
 James Jones and
 Brother Jeremy!
 Page 15

CONTENTS

<i>The Editor Speaks</i>	2
<i>Letters to the Editor</i>	3
<i>Monster CoCo HDs</i>	4
(article) James W. Cross	
<i>CoCo 3 Emulator</i>	6
(article) Steve Ostrom	
<i>Universal Bas Conversions</i>	7
(article) James Toth	
<i>BASIC In Color</i>	9
(column) Fred Remin	
<i>Programming with System IV</i>	10
(article) David Wordell	
<i>The Hardware Hacker</i>	12
(column) Dr. Marty Goodman	
<i>Chicago CoCoFest</i>	15
(article) J. Jones & Br. Jeremy	
<i>Using Syscalls in Basic09</i>	14
(article) James Jones	
<i>Programming in C</i>	16
(column) Henry Harwell	
<i>Basic09 In Easy Steps</i>	19
(series) C.hris Dekker	
<i>Micro News</i>	21
<i>Advertiser's Index</i>	25

DON'T PANIC!

Rick Ulland (Operating System Nine) and Joel Hegberg (OS-9/OSK Answers!) should be back with us in the next issue. We should even hear from David Graham (MM/1 News) again!

POSTMASTER:

If undeliverable return to:
 FARNA Systems PB
 Box 321
 Warner Robins, GA 31099

the world of 68' micros

Published by:
FARNA Systems
P.O. Box 321
Warner Robins, GA 31099-0321

Editor: F. G. Swygert

Subscriptions:

\$25/year (8 issues) US; \$32/year for Canada/Mexico (\$13 US, \$17 C/M for six months- four issues). Overseas \$45/year (\$23 for four issues) AIR; \$37/year, \$18 six months for surface mail. microdisk: \$40 per year, \$21 six months, or \$6 per issue. Overseas add \$10/year, \$5/ six months, \$1/single issue for air mail delivery. Contains programs and source listings from magazine; not stand-alone.

Advertising Rates:

\$15 1/6 page, \$20 1/4 page, \$35 1/2 page, \$60 full page, copy ready. Add \$10 for special placement, \$10 for typesetting (\$5 1/4 or less). Dot matrix will be typeset if deemed unacceptable and submitter billed. 10% discount for four or more appearances.

All trademarks/names property of their respective owners.

Any and all contributions welcomed. Submission constitutes warranty on part of the author that the work is original and not copywritten by another party. All opinions expressed herein are those of the individual writers, not necessarily the publisher or editor. FARNA Systems reserves the right to edit or reject any submitted material without explanation. Remuneration discussed on an individual basis.

Back issues are \$4 per copy. Overseas add \$1 each for surface, \$2.50 airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via e-mail at dsrtfox@Delphi.com. The Delphi CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are still sponsored by Falsoft).

ENTIRE CONTENTS COPYRIGHT
1995, FARNA Systems

The editor speaks...

F.G. Swygert

Well, the "Fourth Annual Last CoCoFest" (Chicago) is now over. And we had a blast! The Chicago Fest, being held at a Holiday Rec Center, is always fun. There is a big lobby area around the heated indoor pool for socializing, and one can always find the regular attendees easily. I took advantage of the pool myself after the eighteen hour drive up there... drove fourteen hours the first day, finished up with four on Friday. Most vendors put up signs in their pool-side windows, making them easy to find also.

I'll leave most of the Fest news to the review articles in this issue. I will relate an almost tragic event that happened after hours though.

Saturday night many of us took a trip across Chicago to a Q-Zar fun center. This is a laser-tag arena and was great fun! If you think this is just for kids you're wrong. Many in the group were 30+ (including myself), and all thoroughly enjoyed the arena. Besides, it was a good aerobic workout (and there were some complaints about sore muscles later).

After spending about an hour at Q-Zar (two fifteen minute games), we all split into different groups. Most were going back to the hotel, but I had promised to get some Hard Rock Cafe or Planet Hollywood T-Shirts for a couple friends. Allen Huffman wanted some "real Chicago pizza". So we spoke with one of the attendants at the Q-Zar center and were directed to a place downtown. That fit perfectly with stopping by the bar at the Hard Rock or Planet Hollywood (both are only a block apart, and not too far from the pizza place), have a drink, and get some T-shirts.

Linda Podraza (Tony's wife), heard Allen and I speak of going, so she came bopping along like a college girl, grabbed Allen and I each by an arm (we were walking side by side), and announced that she was going too. Hmmmm.... no problem there! We DID ask if anyone else was interested in going, I think. At the last minute she decided to turn and ask Tony if he minded... nah, not Tony! By the way, these people (the Podrazas), like most of the CoCo community, are great! Allen and I got to know Linda much better as the night wore on....

We never made it to get pizza. As we were coming off of I-290 (it sort of just ends and becomes a four-lane street in downtown Chicago... Congress Avenue, I think), I hit the back of a taxi. We were on a concrete bridge going down hill, and it was wet from a steady drizzle that had pretty much fizzled out by that time. We went through one

traffic light, and about 150-200 feet later there was another, which really couldn't be seen until one was almost upon it. The taxi sped up at first to go through that light, which was yellow, then suddenly decided he didn't want to do that after all. Instead he stopped HARD. Luckily I had already started slowing, but when I applied brakes harder, the front wheels locked, causing the Rambler to slide. For those who don't know, you can't steer a car with locked brakes, so I let OFF the brakes for a moment and found a hole to escape through. I did this automatically, as I put about 5,000 miles yearly on a motorcycle. One NEVER wants to hit anything on a bike, so you ALWAYS keep a watch for possible escape/avoidance routes.

Unfortunately, the Rambler is about seven feet wide. I missed avoiding the taxi altogether (by running into a parking lot entrance) by about a foot. He was still on his brakes, and I had reapplied mine, so his rear was high while my front was low. I caught his rear bumper on the inside edge of my left front headlight.

Because of the way the old 63' American's front end is made, the only real damage was the fender itself. The bumper was slightly twisted, but we actually pulled that back on sight. The radiator was also punctured, having been pressed into the fan. That was the most serious damage at this point, as we couldn't drive very far without having to add water. At first I couldn't tell how the radiator was damaged, as the mounts (which bolt to the fender) had popped back into place after impact.

Well, the police eventually arrived and took information from both drivers. There really was no dispute as to what happened.. the taxi driver and I both agreed it was really an unavoidable accident under the conditions, and no one was charged. After all, it was obvious that I did try to avoid the accident.

Linda and Allen went to call the hotel, and the others hadn't returned yet. We left directions where we were and told someone at the hotel we'd call back in 30 minutes. I had thoroughly inspected the damage at this point, and decided to move the car some more. We backed under a parking garage and I hooked my tow strap to a support column and convenient crease now in the fender. By backing up I was able to pull the fender away from the tire enough to safely drive the car.

To make a long story short, we drove over to Planet Hollywood, got T-shirts and something to eat at a hot-dog stand, waited

on someone to come out for about an hour (they never found us, stopped just ONE BLOCK short!), then drove the car back to the hotel by stopping 3-4 times and filling with water. We finally made it back to the hotel about 5:00 AM.

I stayed another night with Tony and he helped me run around and fix the car enough to drive home Monday. A nearby radiator shop actually had a core (center part of the radiator) that fit the car, so I replaced that (\$150... not bad since they fixed it in three hours!). I also heard something in the left rear wheel. We removed the brake drum and discovered that the adjuster had fallen out. This means that I only had about 50% of rear brake power. The loose parts were worn enough that they had probably been that way for at least a few weeks. I had never noticed any fade in the brakes, however. This would have still put more of the braking effort on the front discs, which is probably why the wheels locked on very hard braking.

You don't decide to drive a 30 year old car daily and not at least know where to get parts. I have two parts cars at my dad's in South Carolina. One is now missing a left front fender, which happily resides on my driver. I also totally rebuilt the rear brakes when I replaced the fender. I can now tell there is a difference!

I luckily made enough at the fest to cover expenses and the extra cost of repairs. Unfortunately, I'll have to wait a while before a coat of paint graces the Rambler and returns it to two-tone. The pinkish fender does lend a little character to the green with silver top car though. I'll gladly accept paint donations....

It was a great fest nonetheless. Even that night was great. The three of us talked a lot and learned a good deal about each other, more than our few hours twice a year (Chicago and Atlanta fests) for the last 5-6 years could do. And I have to thank the Podrazas for all their assistance in getting me back on the road, the understanding from Linda and Allen, and all the fest attendees who were so concerned when they heard what happened. All know that the old Rambler, which I've had for about 13 years now, is like a family member to me... I'd have to give up the CoCo before my Rambler! I didn't feel bad about the accident until several days later though, and then I was only a day or two from making repairs. I'm just glad I was able to avoid a more serious accident and that no one got hurt.

Take care all, and DRIVE SAFELY!!!

< 268'm >

Letters to the Editor

At last! I've been able to upgrade to Disk BASIC. But, I need some clarification about how to connect the drives to my CoCo 3. I have two old 5.25" drives, a 360K and a 720K.

I would like to know if I can just use the unmodified 720K as drive 1 for OS-9 and still use it to backup DECB programs. Can I still use the 360K drive as drive 0 and 2 under DECB and double sided for OS-9?

I have installed DECB 2.1 on the controller but don't yet have the Extended ADOS 3 package. I am planning on purchasing it.

I'm enclosing my renewal for another year. There has been some good news, good articles, and a lot of good learning in recent issues. So continue to provide a link between the whole CoCo community. Until next time!

Carlos Burgos
Urb. Miraflores, C.23 B12 #5
Bayamon, P.R. 00957

Carlos, thanks for the letter! In this issue you will find an article about disk drives.

In general, you can do exactly as you want. Without ADOS, you would only have two 35 track single sided drives. DECB will only read the one side and first 35 tracks on either of your drives. If you use the 80 tracker for backups under DECB, only your 80 tracker (or another setup the same way) will be able to read those disks, and the 80 tracker can't read disks made on the 40 track drive. You can get around this by backing up twice... first to the 80 track, then back to the 40. Would be faster and easier than swapping disks in a single drive anyway.

Extended ADOS 3 (\$50 from FARNA) will allow you to configure your drives as you want, and also to use both sides of the 80 tracker as drives 1 and 3 as full 80 track drives. You would still be limited to 78 file entries though (the same as a 40 track single sided drive). This is due to the limited room DECB allows for the number of files on a disk. OS-9 can be set to use the drives as two full capacity drives easily (360K and 720K). Good luck!

I received the following list of questions. I'll present each question and answer separately, then give the address of the writer at the end.

1) Is there a good M/L mailing list program out there?

If you use OS-9, I'd suggest getting Bob van der Poel's DML9. I use this for the magazine labels. MICCC sells a multi-use program called Home-Pac for DECB that does labels well.

2) How do I connect a 1200 baud modem to the CoCo?

In a previous issue, there is a complete article on connecting modems. If you send \$4 for a back issue, I'll look it up. To extensive for here!

3) Can I use programs such as Max-10, CoCoMax, etc., with OS-9? Would they all run at one time? Is OS-9 similar to MS Windows?

You can't run DECB based programs under OS-9. That's why a lot of people still use DECB along with OS-9. There aren't any really good graphics programs for CoCo OS-9, not comparable to CoCoMax anyway. OS-9 WILL run more than one program at a time. You should be able to run two with 128K, four or more with 512K (depending on memory needs of the program). OS-9 works sort of like MS Windows, but better! The CoCo disk controller does halt the CoCo during disk I/O, which slows things down a lot though.

5) The Hi-Res Interface works in Max-10, but there is a little switch on it. It sometimes won't let the mouse control the arrow.

The hi-res interface will only work with programs written specifically to use it. My guess is that the switch on the side turns the hi-res portion on and off so you don't have to plug/unplug it all the time. When not using a program that requires the hi-res, turn it off!

6) One of my programs says to type "DOS" to start. It then gives an I/O error. A directory gives a bunch of squares and "#". The last line says "FS ERROR". How do I fix this?

You don't fix this! The disk is an OS-9 disk, based on what DECB gives as a DIR listing. A backup copy is the only real fix.

7) One of my disk drives runs a little when inserting/removing disks. Normal? Some drives do this normally.

8) The belt on my DMP110 came off. How do I realign it properly?

My only suggestion is to keep trying by moving it a little at a time until you get it right. Maybe a reader has a better idea...

9) I could use a serial to parallel interface.

These are still made for PCs. SBUG (1251 W Sepulveda Blvd. - Suite 400, Torrance, CA 90502; 310-539-9702) has some great 128K printer buffers that also serve as adapters. I have one of these and added a switch to go between serial and parallel inputs (CoCo & PC) so it also acts as an A/B switch box. They are only \$35, a great deal! Any PC store should have one, but you'll have to make a 4 pin DIN to 25 pin cable to connect the CoCo (even for the 128K buffers noted above).

Jerry Alexander
208 E. Gaskins
Harrisburg, IL 62945

< 268'm >

Monster Hard Drives for the CoCo... Part 2

James W. Cross

Installing a large hard drive under your CoCo/OS-9 system.

I have been running my COCO 3, DISTO 4-N-1, with a 380 Megabyte Micropolis 1684-7 SCSI hard drive for 4 years. My experiences with a COCO hard drive are restricted to OS9 Level II. I do NOT use my hard drive under DECB ("RSDOS").

Before starting part 2, I would like to revisit one point from Part 1 (see Robert Gault's article in the previous issue). Just as 1 Kilobyte is often used to refer to 1024 bytes, 128 Megabytes may refer to as many as 134,215,680 bytes and 4 Gigabytes may refer to 4,294,967,326 bytes. The apparent discrepancies are due to the abbreviated transitions people use when converting from the base_2 to the base_10 number system. Similarly, disk sizes are not what they seem. It is common for drive manufacturers to specify the "raw" capacity of the drive (before formatting). Be aware that not all of the specified disk capacity is available for user storage. Using 256 bytes/sector, as much as 18 to 22% of the raw capacity can be lost to formatting. For example, my 380 Megabyte (raw) hard drive provides about 310 megabytes of usable storage.

Foreasy identification of files and modules to which I refer, I will follow each name with the CRC in parentheses. You can check your module's CRC using OS9's IDENT (\$548BB2) command. I can only "speak" for the modules with the specified CRC; modules with identical names but different CRC's may function differently with respect to the stated problem. Here is a list of OS9 utilities that I found invaluable in setting up and working with a "monster" hard drive:

1. HMODE (\$2F78CA) Displays/changes hard drive device descriptor (/h0)
2. MKDIR (\$8ABB69) Creates variable sized, contiguous sector directories
3. FMAP (\$A3C17C) Displays file/directory segment allocations & sizes
4. DCHECK (\$F4D5DD) Checks file system for allocation mismatches
5. DED (\$7C5F04) Disk sector editor
6. ASM (\$6996B8) Assembler from OS9 Level I also works for Level II.
7. MV (SCD7337) Moves a file to

a new directory without changing date.

Note: Do not confuse the MKDIR utility I used, written by Peter Lyall, with Tandy's original MAKDIR utility which can only create fixed size directories.

Next is a list of modules/programs with which I encountered hard drive related problems:

1. RMA (\$F83DD9) Creates device drivers/descriptors minus the MODE byte.
2. FORMAT (\$744368) Hard coded for 1 sector/cluster (DD.BIT = 1)
3. RBF (\$EFBE13) Creates 4 sector directories/file ignoring PD.SAS
4. RBF30 (\$4B1153) Fails to deallocate file descriptor for deleted files.

If you're going to use an assembler to create a device driver or device descriptor, use the older assembler, ASM, from OS9 Level 1; it runs fine under Level 2. The new assembler, RMA, does not provide for the MODE byte. Unlike the older assembler where you control the module header content, RMA depends on the linker, RLINK, to generate the module header. RLINK gets the header information in the source code file being assembled from the PSECT directive which has no field for the MODE byte. The MODE byte field is unique to device drivers/descriptors. Without a MODE byte in its device driver, the disk drive works for awhile and then mysteriously crashes; actually it trashes LSN0 & the allocation map. The DED utility was an invaluable help tracking this problem. You avoid the problem by using the older assembler, ASM.

Part 1 of this article covered the change required to the FORMAT command and the need to increase DD.BIT (the number of sectors in a cluster) to a value greater than 1. A cluster, you may recall, is the smallest number of sectors OS9 will allocate at one time. Each bit in the disk's allocation map represents 1 cluster. The number of sectors in a cluster may be different for each type of disk drive, but the number should always be an integral power of 2 (i.e.; 1,2,4,8,16,32,64, etc).

If you have a SCSI hard drive, remember to ALWAYS use the L (logical) option with the FORMAT command. Never

attempt a low level (physical) format of a SCSI drive with the FORMAT command. Unlike older drives, drives with embedded SCSI controllers perform the physical formatting within themselves upon receipt of a single command. The DISTO 4-N-1 comes with a BASIC program that can send the proper command, but TANDY'S FORMAT utility doesn't contain the code to properly initiate a physical format of a SCSI drive, only to logically format one.

Did you know that a directory is just a file with a predetermined data format? This fact allows OS9 to create and delete both files and directories with only minor differences between those two types of operations. When creating files or directories, the file manager, RBF, begins by allocating 1 sector for the 'file descriptor'. Then, the intent was to have RBF retrieve the minimum size segment (i.e. the number of sectors) to allocate for the "body" of the file/directory from the process variable, PD.SAS. PD.SAS gets its value from IT.SAS in the drive's device descriptor.

For a floppy disk, IT.SAS is usually set to 8; consequently, newly created directories are allocated 8 sectors (when DD.BIT = 1). Tandy's RBF code appears to have been altered to assume DD.BIT always = 1 (i.e. 1 sector/cluster). For "monster drives" where DD.BIT is greater than one, RBF reserves 1 sector for the file descriptor but then allocates the remainder of the sectors in the cluster without retrieving the minimum size segment from PD.SAS. For example; my hard drive has DD.BIT = 4 (sectors/cluster), all directories created by Tandy's MAKDIR on that drive will have only 4 sectors allocated instead of the 16 specified by my PD.SAS (usually 16,32 or 64 for a hard drive).

While this smaller allocation appears to function properly, it causes increased fragmentation when directories grow in size as files are added. Large directories like /H0/CMDS get spread all over the disk which greatly increases the time to find and execute the commands that you type in. Over time, the systems responds noticeably slower and slower. This same

“bug” prevents the MKDIR (not MAKDIR) utility from creating variable sized directories because it does so by modifying PD.SAS.

Note that users’ patches to RBF may have addressed this problem. The following patchfile maybe used as input to the MODPATCH (\$F3899A) utility to patch the original RBF (\$EFBE13) code to handle DD.BIT>1. This patch affects only drives with DD.BIT > 1; it does not change the way RBF handles disks with DD.BIT = 1.

rbf1.pat:

```
L RBF
C 0D5E 83 CC
C 0D5F 00 00
C 0D60 01 01
C 0D61 EB 10
C 0D62 0E A3
C 0D63 89 62
C 0D64 00 27
C 0D65 20 0B
C 0D66 02 E6
C 0D67 44 0E
C 0D68 56 10
C 0D69 64 A3
C 0D6A 6A 6C
C 0D6B 66 23
C 0D6C 6B 04
C 0D6D 24 ED
C 0D6E F8 6C
C 0D6F ED 12
C 0D70 E4 12
V
```

The SAVE (\$8F769A) utility can be used to save the modified RBF module for use in creating an updated os9boot file, or the command line “MODPATCH <rbf1.pat” can be added to your STARTUP file.

Some side effects of the large storage capacity offered by monster hard drives can detract from system performance, but with a little care and planning, most of these can be overcome. Note the techniques used below may benefit a floppy disk CMDS directory structure as well as for a hard drive, but of course the sizes may be smaller.

First, consider the general execution directory, /H0/CMDS; it will likely be the most often used directory on your hard drive. Every invoked command not resident in memory causes a search of the

execution directory. With plenty of storage space and access to the COCO community, you’ll constantly be adding new commands (executable files). If you don’t plan for some expansion, your CMDS directory can fragment as it grows causing slower response when you type in those newer commands.

Here is where the variable size feature of the MKDIR utility comes in handy. I used it to allocate 96 contiguous sectors (244_sector clusters) for my /H0/CMDS directory; that allows for 760 commands before the directory will fragment. I currently have 200 command files. Adding commands still increases the maximum search length, but considerable time is saved by minimizing the seek time. Additional time is saved by drives like mine that perform an automatic read of sequential sectors into the drive’s cache RAM after the first LSN (logical sector number) is sent by the controller; after that, the remaining sectors of a file or directory are read from RAM to RAM (ie. cache to CoCo) with no additional seek time.

Next, we look at the boot up delays, that annoying time one spends waiting after typing “DOS” until the OS9 command prompt is displayed. After OS9Boot is loaded, the next access to the hard drive is to load the GRFDRV module from the /H0/CMDS directory. I made GRFDRV the first file that I copied into my CMDS directory to minimize the directory search for it. The next file my boot up uses is LOGIN, so I placed that 2nd in the CMDS directory followed by SHELL then WCREATE. I use LOGIN because I have multiple users, but TSMON is much too slow so I modified CC3GO to call LOGIN instead of loading the immortal shell. This prevents errors or EX from dropping back into a super user shell.

My system uses a DISTO 4-N-1 with a real time clock (RTC) so I don’t need SETIME each time I boot, but those without an RTC may want to put SETIME near the top of their commands directory. The down side of placing these files first is that they are rarely used except at boot up, and all command searches after that must pass through them, but that minimal added time is not as noticeable as the long boot up delay.

Lastly, pay careful attention to your hard drive’s filing system, the directory

tree. Early planning of your directory tree structure will help you organize your ‘monster’ hard drive in a way that is meaningful to you. This is far more critical than it was with a floppy disk. Huge disk drives take a longer time to search. You can avoid these time delays if you store your files in ‘the tree’ where you would logically go to look for them. I try to keep my directory names between 3 and 6 characters long to reduce typing when using full path names, but I will make the names longer if I need to for clarity. I use MKDIR to create subdirectories only large enough to hold the contents of an expanded archive file when 8 or more files are included within the archive; this helps me keep them organized.

Editor: Hopefully, the information provided by Robert Gault in the previous issue and by James Cross here will help readers to get the most out of their CoCo OS-9 systems. With the falling cost of large hard drives, and the scarcity of smaller ones, more people should be using larger drives in the future. For your convenience, below is a couple sources for SCSI hard drives:

Hi-Tech Component Distributors
805-681-9961/ FAX 9971
(42MB Conner, \$55; 328MB Seagate, \$129; 525MB Seagate, \$199)

Jem Computers, Inc.
617-497-2500/ FAX 491-1006
Email: jem.computer@channell.com
127MB Quantum, \$89; 330MB HP, \$119; 670MB HP, \$249)

*James (Jim) Cross
may be reached for
comment at his e-mail
address on CompuServe:
740-40-3003@compuserve.com*

The CoCo 3 Emulator

Steve Ostrom

Running a CoCo 3 on an MS-DOS clone!

I sent for a registered copy of the CoCo 3 Emulator by Jeff Vavasour on December 15, 1994. A blank 5-1/4" 360K (IBM format) disk and a personal check for \$25 was included. In a few weeks I received a disk with multiple files on it... the CoCo 3 Emulator.

I have a 486DX2/66 PC and a 512K CoCo 3 with Extended ADOS 3 burned into a ROM chip in the disk controller. I used a PC file manager program to create a directory on my PC hard drive and copied all the files from the new disk to this directory. There are three documentation files on the disk, and I printed these from the PC. The 16 page COCO3.DOC file is the main source of information, containing detailed instructions on all aspect of the emulator. It is fairly well eritten and goes into quite a bit of detail on the various functions included in the emulator package.

I followed the step-by-step instructions but met with some major problems right away. I remember having had a lot of trouble with the original CoCo 2 Emulator because the disks made on the PC could no be read on the CoCo and vice-versa (I understand this problem was corrected later). So it didn't surprise me when I formatted a disk on the CoCo and it couldn't be read on the PC. I won't go into all the details here, but it is sufficient to say that after five hours of frustration I finally figure out that the problem was in my Extended ADOS 3 ROM -- the Emulator is just not compatible with it. After I exchanged my normal disk controller with a spare Tandy controller (with a normal DECB ROM), all the incompatibiliites disappeared. Nothing about this potential problem was in the documentation, so I hope Jeff will address this later.

The first step in the procedure is to use the PC to copy the file named GET3ROM.BAS to a CoCo disk and then run this program on a CoCo 3. This is a utility to copy the CoCo 3 ROM onto disk, where it can be ported back to the PC and used by the emulator. Once I switched to a disk controller with a standard DECB ROM, this went great. The program ran fine, and the ROMs

were copied to the CoCo disk. The PC PORT program does the transfer from the CoCo disk to the PC. This also worked well. There is a documented procedure to transfer the ROM by either a cassette recorder or serial port if needed.

The emulator is started by typing COCO3 at the DOS prompt in the emulator directory of the PC. A title banner appears, giving a list of available function keys and their tasks, and letting you know whether your emulator is operating as a 128K or 512K CoCo 3. Pressing any key puts you into the familiar CoCo green screen with the OK prompt. The cursor was flashing very fast so I returned to the option menu (F6) and made some adjustments. Near the bottom of the screen is an emulator speed indicator. I found that for my PC, I needed to set the level to about the 1/4 mark to slow the cursor to about the same speed as my CoCo 3, which is sitting right next to my PC for comparison. I also selected the PC keyboard layout, set up my PC mouse and joystick as left and right CoCo 3 Joystick emulation, selected my Soundblaster card to emulate the CoCo 3 speaker, selected the RGB color set with a black border, and selected the PC printer port to emulate the CoCo 3 serial port. The level of emulation is astonishing!

When I returned to the emulator by pressing <ESC>, the feeling was almost eerie. My CoCo screen was now on my PC! The fonts and colors were perfect. The only difference I could really see was the screen was a lot larger and the letters a lot clearer on my SVGA, 0.28 dot pitch PC monitor. But would the emulator really work, or does it just give the screen a pretty look?

Most of the CoCo programs I have tried worked perfectly in the emulator. But I did have some trouble with two in particular. I own one of the original "Marty's Nightmare" disks. This is a great, non-violent game, similar to Pacman in objective with voices and sound effects. This disk is very copy protected. It loads perfectly on a CoCo 3 but will not load on the PC. I'm sure this is due to the copy protection scheme used. I also tried FLIPPY, a cute game by

an old friend of mine, Rodger Smith. The graphical synchronization was off, making the game almost impossible to play, probably due to some special programming. OS-9 Level II will also run with few problems.

All in all, this is a fantastic program for all of us old CoCo fans who use both a PC and the CoCo. A person can still enjoy most of the old CoCo programs with super clarity, and without the fear of having the CoCo break down, losing the availability of all the software that person has invested in over the years. This program ranks near the top of my list for "best buys".

I would like to see some response to this review. Please write to me or this magazine with your comments and a list of programs that would not run, or that you were able to get running with some tweaking of some sort. I think this emulator is one of the most exciting things to happen in the CoCo community in a long time!

Steve Ostrom
12612 Cedar Lake Road
Minnetonka, MN 55305-3944

How to get the CoCo Emulators

The CoCo 2 emulator is shareware. It can be downloaded from several sources. The CoCo 3 emulator is a commercial product. It can only be obtained by sending \$25 to Jeff Vavasour, . This info is provided in the CoCo 2 emulator archive.

It is best to get the CoCo 2 emulator file first. There are some utilities to test your VGA card for compatibility. These are included on this month's "microdisk" (OS-9 side). These are PC programs, and will have to be transferred to a PC to run with a utility such as PCDOS.

For those without on-line access, a copy of the CoCo 2 emulator can be obtained from FARNA Systems for \$5.00. They will be in PKZIP format on a 360K 5.25" disk. You may specify 720K 3.5" format if you prefer.

Universal Base Conversions

James Toth

Easily convert between decimal, binary, and hexadecimal numbers

You can't go very far into the world of computing before you run into a variety of unusual number bases. Base ten is the most popular base in everyday life, probably because humanity's earliest counting was done on our ten fingers. Base two (binary), on the other hand, is better for computers, mainly because of the on/off two-ness of computer circuitry. And as a compromise of sorts between these two bases, there is base sixteen, which is compact like base ten but also is easy to convert into base two, the computer's favorite.

When working in peculiar bases, keep in mind that the number of available digits is equal to the number of the base. That is the only real difference between the different bases. All other rules and patterns stay the same — if it works in one base, it probably works in another.

Thus, in decimal (base ten) you have ten digits, 0 through 9. In binary (base two) there are only two digits, 0 and 1. And hexadecimal (base sixteen) has sixteen digits, which is more digits than there actually are, so some letters are used as digits to produce 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. "A" is equivalent to ten, and "B" is equivalent to eleven, and so on. If you work in hexadecimal long enough, you eventually get to the point where B52 no longer reminds you of airplanes or bingo.

You may wonder how it is possible to get along with only two digits as in the binary system. Whatever the number of allowed digits in a particular base, the same familiar everyday mathematical rules still apply, and you can do the same things.

For example, think about normal decimal counting: 0, 1, 2, etc. When you get to a count of 9, you have used up all allowed decimal digits, so you begin reusing digits — you write "1" to the left of the number (to indicate that you have used up all of the digits once) and then you begin again with "0" in the rightmost column. That is, you write down "10". If you study the "counting table" accompanying this article, you will notice that this is done in all bases.

Some bases run out of digits quicker than others. The binary system runs out of digits almost immediately. In such bases the numbers tend to be longer as the allowed digits are reused more often, creating numbers that are many columns long. The reverse is true of decimal and hexadecimal.

Technically there is an unlimited number of possible bases, but bases ten, two, and sixteen (decimal, binary, and hexadecimal) are the popular ones in the computer world. When working with these bases, it sometimes is helpful to be able to convert one base into another. There are various ways to accomplish this.

One popular method of base conversion involves repeatedly dividing the number you wish to convert by the base you wish to convert it into. By keeping track of the remainders obtained with each division, you eventually end up with the digits that comprise the number in its new base.

For example, suppose you wish to convert twenty (decimal) into its binary counterpart. To do this, first divide twenty by two and remember the remainder. Then take the results of that division and divide by two again, keeping the remainder again. Do the same with the results of that division. Continue in this manner until the results turn out to be less than two (the destination base), and then end it all by treating that final result as one more final remainder. Your list of remainders in reverse order is the decimal number twenty converted into binary, believe it or not.

Got all that? Let's try it. To convert decimal twenty into its binary equivalent, first divide twenty by two and keep the remainder:

$$20 / 2 = 10 \text{ remainder } 0$$

Now continue in this way, dividing the results of each preceding division by two:

$$10 / 2 = 5 \text{ remainder } 0$$

$$5 / 2 = 2 \text{ remainder } 1$$

$$2 / 2 = 1 \text{ remainder } 0$$

And since the result (1) is now less

than two, we have:

final "remainder" 1

Now, if you take that list of remainders in reverse order, you get 10100, and that happens to be decimal twenty converted to binary. No kidding.

Though interesting at first, the above procedure can get old and tedious in a hurry, and that is the kind of thing that computers do well. Someone should write a program to do this, I thought, and so I did. The listing accompanying this article, UBASE.BAS, is a universal base converter program. It is based on this repeated-division-with-remainder procedure, and it is general enough to convert any base into any other base — very cool.

Now that you know the basics of bases and conversions between them, let's look at some of the prominent lines of the base converter program UBASE.BAS.

Of course, the first things that UBASE needs to know are, what base are you starting out in, and what base would you like to end up in. That is the point of the queries in lines 30 and 50. Similarly, line 70 lets you enter the number that you want to convert.

The actual divide-and-keep-the-remainder process happens in lines 170 through 230. Variable Q is the result of the division — that's Q for quotient. R is the remainder. The various remainders are stored in the array D() — that's D for digits. As it is set in line 10, array D() is big enough to produce a 25-digit converted number, which should be plenty for any reasonable purpose.

Since UBASE is supposed to be a "universal" base converter, you may not be starting out in base ten as we did in our sample conversion. Lines 90 through 160 deal with this by guaranteeing that your starting number is in base ten, regardless of what base you actually enter it as.

Finally, lines 240 through 300 print out the converted result (remainders), digit by digit. As we said, the remainders are stored in array D(), and you can see in line 250 that this array is being

printed out in reverse order (STEP -1).

The experienced programmer will notice some references to the ASCII system such as ASC and CHR\$. The reason for this is to keep things general. 0,1,2,3,4,5,6,7,8,&9 normally are considered numeric values. A,B,C,D,E,&F generally are not considered numeric, but in this program they can be numeric. So rather than working with the characters themselves, we work with their ASCII values. All characters have numeric ASCII values, whether the characters themselves are numeric or not. Thus we can treat 0 - 9 and A - F the same.

Although UBASE does reject digits that are too big for the base being used (line 140), the program is not totally idiot-proof. It does not reject fractional input. It does not reject negative numbers. And breaking out of a for/next loop as in line 140 is not exactly good form. So if you plan to give the program frequent moronic input, then these things will give you something to do yourself in the way of improvement.

When actually running the program, you can convert number after number without retyping RUN because of line 310. If you need to switch to a different starting base or ending base, simply enter nothing at the number query. (Line 80 enables this.) And if you want to quit completely, you can enter nothing at both the number query and the starting base query. (Line 40's reason for existing.)

And so, you never need fear bizarre bases again. Happy converting!

```

5 REM UBASE.BAS
6 REM UNIVERSAL BASE
  CONVERTER
10 DIM D(25)
20 PRINT
30 INPUT "START BASE";SB
40 IF SB=0 THEN END
50 INPUT "END BASE";EB
60 PRINT
70 PRINT "BASE"SB"NUMBER";
  :INPUT SN$
80 IF SN$="" THEN 20
90 SN=0
100 FOR X=1 TO LEN(SN$)
110 Y$=MID$(SN$,X,1)
120 V=ASC(Y$)-48
130 IF V>9 THEN V=V-7
140 IF V>=SB THEN 70
150 SN=SN+V*SB^(LEN(SN$)-X)
160 NEXT X
170 X=1
180 Q=INT(SN/EB)
190 R=SN-Q*EB
200 D(X)=INT(R+.5)
210 X=X+1
220 IF Q>=EB THEN SN=Q:GOTO
  180
230 D(X)=Q
240 PRINT "BASE"EB "NUMBER:";
250 FOR Y=X TO 1 STEP -1
260 D(Y)=D(Y)+48
270 IF D(Y)>57 THEN D(Y)=
  D(Y)+7
280 PRINT CHR$(D(Y));
290 NEXT Y
300 PRINT
310 GOTO 60

```

COUNTING IN VARIOUS BASES

DECIMAL	BINARY	HEX
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

Questions/comments may be directed to the editor or directly to the author:

James Toth
R.D. 4, Box 230
Punxsutawney, PA 15767

< 268'm >

NOW AVAILABLE!

The long awaited update of Paul Ward's "Start OS-9"...

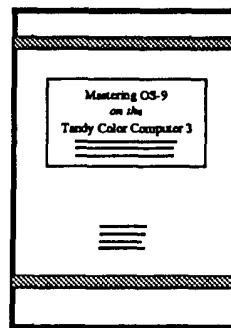
Mastering OS-9

Edited and revised by Francis G. Swygart

This new edition contains revised, easier to follow tutorials, an index, revised information articles, several NEW informative articles (including Rick Ulland on making an OS-9 boot disk), and several extra utilities on disk! 262 softbound pages (5.5"x8.5").

Price for book and disk is only \$35 post paid (US)

(Canada add \$2 for s&h, overseas add \$10 for airmail)



FARNA Systems

Box 321

Warner Robins, GA 31099-0321

912-328-7859

Over the last couple issues I covered quite a bit for new users of the Color Computer, starting with how to load and save a program from tape or disk through starting you onto writing your own programs. Just one thing I omitted in the segment about I/O errors with a tape system. If the remedies I gave still do not work, then it would be advisable to get the head of the tape deck aligned by someone. Either the Tandy service department or their equivalent elsewhere (ed: With the cost of portable tape decks now, it may be just as well to replace the deck).

In preparing to write the next segment of this column, I reread the last installment and there glaring at me from the page was a question I had failed to address. The simple statement "we must now decide what we want the computer to do" sent my mind back to 1984 when I first purchased my 16K CoCo 2. There I was staring at this powerful machine and thinking "well, what do I want it to do for me?"

My naive answer to myself went something like: "First, I want it to wake me up in the morning after having started the toaster and coffee pot, it should have also at this time turned on the iron to do my uniform and wakened the kids to get ready for school. Next it should be able to keep track of all my phone calls, bills and their payments, mileage on the car and service details, keep track of all my appointments, make a shopping list for my wife each fortnight, and keep track of our bank accounts every payday. It should also be able to keep track of everything we own in the form of an inventory, complete with updates and prices. Not to mention a database of all phone numbers which it can dial automatically whenever I want to make a phone call."

Do you get the idea? In my naivety about computers I was expecting my little 16K machine to be able to run my life for me without me lifting a finger. Not only that, but I should be able to get it to do all of this by simply coming up with one or two lines of program from the basic manual.

Well, I very quickly learned that yes, the CoCo could do all these things (within reason), but it would take some thought and work on my part. But where to start? The answers:

- 1) Buy a program for each job I want the computer to do.
- 2) Try to write a program from scratch.
- 3) Use the ideas of others and change

them to suit my needs.

I found, and I think that you will do the same, that the best way to go about it was answer number three, and this was achieved with invaluable assistance from two types of publications. The first was the manual that came with the system and the second was magazines dedicated to the CoCo.

So, now for the nitty-gritty. Have in the back of your mind what you want the computer to do, but temper that with a little constraint, be realistic. Next, see what the computer CAN do. How? Firstly read the manual from cover to cover, read it again while experimenting, then type in the programs at the back and see how and why they work. By taking this step first you will get a good idea of what the computer is capable of and how you go about getting it to work. Not only that, but a very good understanding of the available commands and how to use them will also be obtained.

Next the magazines. Not only this one, but get ahold of back issues of the Rainbow, Hot CoCo, Color Computer, and any others dedicated to the CoCo. Start typing in the short programs that are listed. You will more than likely make mistakes while doing this and by doing so will learn how the programs work and how to fix the problems (called debugging).

A word of advice: at this stage, do not try to do it all on your own. There are people who can help you as you go along. There are some user groups still active across the country, the on-line service, or you can always drop this magazine or one of the columnist a line.

By this time, a few weeks or even months down the track, you should become quite proficient with the BASIC programming language of the computer (as well as the language of swearing in frustration). DO NOT become discouraged because it will not happen over night. Remember, professional programmers spend years at a university and then months or even years on one program before it is ready for release to the public.

Now let's go back to the original question: "What do I want the computer to do?" In this case lets say that you want it to be a database for phone numbers and addresses. you will use the process that I explained in the last article. This time, however, you are more prepared.

Once you have decided exactly how you want this program of yours to work, you can utilise all the knowledge you've gleaned

out of the last few weeks or months of typing in and debugging programs from the magazines. By taking some of the routines used by others to do a particular task and then changing it to exactly what you want it to do, you will very quickly come up with your very own program custom made for your needs.

If you use someone else's routine in your program, then you must remember that the original writer has the copyright to that code. Ensure that if you are going to publish or sell the program that you either check with the originator to see if they are agreeable to your use of their code. If they cannot be contacted they should at least be given recognition for their part in the program comment lines and/or documentation. You could easily put in a line like:

10 REM Original code by J. Bloggs

20 REM Updated 1995 by F. Remin

or:

10 REM Program by F. Remin


20 REM Sort subroutine by J. Bloggs

Of course such statements could be made in the documentation as well.

By using the above procedure s you will learn a great deal more about how your CoCo and how it works compared to simply buying a commercial program and then adjusting your needs to what the program can do. You will also learn about computing in general, more so than someone who uses other systems such as Windows and Macs. Those systems teach you most how to push the right buttons and buy the extra software and peripherals to do what you want.

This does not mean that you will not need to buy commercial software. A spreadsheet program, for example, would be horribly slow and unwieldy if it were written in BASIC compared to a commercial program written in assembler. Writing such a program would also take a tremendous amount of time and effort on your part.

Oh, and by the way, don't forget to send your masterpiece to this magazine for publication. That way all the other budding programmers out there can benefit from your work just as you have done from the work of others.

Well, that's about all I have for this time. Remember, nothing can replace your physically going through the manual supplied with your computer. And the best source of background information comes from this magazine and back issues of those mentioned earlier. Happy CoC 

Programming with the Delmar System IV *David Wordell*

I have had a PT68K4 Single Board Computer from Peripheral Technology, in Marietta, Georgia for a little less than a year now, running Professional OS-9/68000 v2.4. For those who may not know, this is the board used in the System IV by Delmar. Ed Gresick of Delmar advises contacting Peripheral Technology for kits and this is the route I chose to take. Some of you may have read my article in the October 1993 issue of Metamorphosis magazine about how I put this computer together. Since writing that article, I have visited the Atlanta Fest, where I bought the IDE hard drive controller card for the PT68K4 and later installed a 245 Meg IDE hard drive. I have also discovered how to make a descriptor to allow my 3 1/2 inch 1.44 Meg Floppy Drive read 3 1/2 inch 720K floppies. I have therefore removed the 720K drive that I told about installing in the last article since I no longer need it. I also have a Color Computer III, running Nitros9 v1.16 by Gale Force Enterprises. It has a 3 1/2 inch 720K floppy also. I am able to easily try C code in both machines since both will read the same disk format.

I have been studying C language programming, trying to learn how to write something useful. I have written some utilities for the PT68K4 which work like those on the CoCo because I missed them on the new machine. The main focus of this article is about a "header" file I have written which allows control of the cursor on the screen and thereby control of where things are printed on the screen. There is a header file for the C compiler on the CoCo, named "screen.h", that appeared in an archive called header.ar that I got from our club Bulletin Board. It used all the familiar "display" codes on the CoCo III to control the cursor. I wanted to be able to write C code that will compile on both of my machines and obviously every program has something happening on the screen at some time. If I wrote code for the CoCo III that used "#include <screen.h>" and wanted to use that code on the PT68K4, I would have to have a similar header file for the new machine. I tried to follow as closely as possible the format and layout of the screen.h on the CoCo III. I also laid out the codes in the order that they appeared in my copy of the System IV manual, kindly supplied by Ed Gresick. I wish to thank him publicly for furnishing this to me. Without it I would never have been able to accomplish this task. Thank you Ed.

The first entry in the manual, under System IV Screen Control Codes, is Sound Bell. In the CoCo version of screen.h I found the statement "#define BEEP putc('\x07', stderr); /* Beep at me - beep at you */". Notice the "\x07" in the statement. If you type "display 07<enter>" on the CoCo, you hear the beep. Notice also that in C, the "display code" is being sent to "stderr" which is defined in the OS-9 Level II Manual as "The route through which your computer sends error codes and other messages to the screen". I decided to try this one first, since the code for Sound Bell on the PT68K4 was CNTL-G. This is in fact Hex 07. I wrote a short C program which defined BEEP and then called it several times. It went like this:

```
#include <stdio.h>
#define BEEP putc('\x07', stderr);

main()
```

```
{
    BEEP
    BEEP
    BEEP
    BEEP
}
```

Then came my first surprise, it worked. Imagine that. I'm not usually that lucky. I then proceeded to write the rest of the code for screen.h since I felt that I now had it figured out. Then I wrote a little routine to test the new screen.h. At first it wouldn't compile. I had a few things to clean up like leaving out */ (asterisk-slash) a few times and ; (semicolon) a few times. I finally got it to compile correctly. Then came the big test. Things seemed to print to the screen, and then disappear. It happened very fast. I couldn't really tell what was going on. Then I got to looking at some of my other header files for the PT68K4. I was listing "stdio.h" and I found the following statement. "#define putchar(c) putc(c, stdout)". This was the clue that I needed to figure this out. Notice the "stdout" with the "putc" function. I decided to try changing all the "stderr" statements to "stdout" in the version that I had written for the PT68K4. Then I tried my test file again. This time it worked perfectly with one exception.

There was one function that would not seem to work. At first I couldn't figure out what was meant in the System IV Manual by the statement "ESC [<row> ; <column> H" for the "Position Cursor" code. Nothing I tried would work. I gave up on that for a while and tried writing a pos_cur routine using a "for" loop and CUR_RGT and CUR_DWN. At first it seemed to work okay but further testing showed errors. I then went back to the code in the manual, determined to figure it out. I discovered that the row and column information had to be input in a special manner. An example or two will explain.

If we wanted to position the cursor to row 5, column 2, we would have to use the ASCII value for the character 5 and the character 2, not the numbers 5 and 2. If we now write the pos_cur function, as the code in the manual requires, we would write it like this. "putc('\x1B', stdout); putc('\x5B', stdout); putc(row+48, stdout); putc('\x3B', stdout); putc(col+48, stdout); putc('\x48', stdout)". The \x is the way to signify to the computer that the number following is in hexadecimal. The 1B is the ESC, the 5B is the [, the 3B is the ;, and the 48 in the last entry is the letter H. Now for the row+48 and col+48. The ASCII value for the character 0, in decimal, is 48. For a character 1, 49, a character 2, 50, etc. Can you see that to get the ASCII value for any numeric character, just add 48 to it? In the statement "putc(row+48, stdout)" we will be putting 5 plus 48, or 53 to stdout in the example above. The "col+48" would be 2 plus 48, or 50. This worked great until I tried a value above 9 for row or column. A new complication arose. I found that I had to "putc" the ASCII value of the left digit of a two digit number first as if it were a single digit, and then "putc" the right digit. So for a row value of 15, I had to use 1 plus 48, which is 49, then 5 plus 48, which is 53. That meant that I would need to use two variables for row and two for column. One each to handle the left digit, and one each to handle the right digit. Also, I would have to recognize when there

was only one digit, numbers 9 or less.

After much experimenting, I discovered that a two step process worked best for me. First I declared the needed variables and then set their values inside the function. I knew that if a variable is an integer, any value after the decimal point will be thrown away or truncated. Therefore, if the row were 24, row divided by 10 would be 2. This also held for column values. If the column were 75, col divided by 10 would be 7. This is how I extracted the "left digit" from the row, col value. I called them "rowa" and "cola". It is important to realize that I now have established a value that can be used for the next task. The next task is to extract the right digit. Looking at the above row value, 24, if you multiply the new "rowa" value, 2, by 10, getting 20, and subtract the 20 from row, which is 24, you have 4, the "right digit". I decided to call this "rowb". It naturally follows that the "right digit" for the column will be "colb".

The second step was to make the decision as to whether the row or column value was a single digit number or a two digit number and act accordingly. I used the statement "if (row > 9)" to make the decision. If this is true, it is a two digit number, and we "putc(rowa+48, stdout)". If it is not true, the putc is skipped. The next statement deals with the "right digit". It is simply "putc(rowb+48, stdout)". The column values are handled exactly the same way.

I have included the screen.h file that I wrote for the PT68K4. Once you have typed it with your favorite editor and saved it as screen.h, just put it in the directory that holds all your other header files. This is usually /dd/DEFS. Be sure to have the statement "#include <screen.h>" at the beginning of your program if you use any of the definitions in this file. I hope that you find it useful, whether or not you own a CoCo III, a TC-9, a PT68K4, a System IV or V, an MM-1, a KIX-30, a TC-70, or whatever OS-9 machine. Remember, "Friends don't let friends run DOS."

If I am not mistaken, the MM-1 is very similar to the CoCo in the area of Display Codes. I would imagine that, if it does not already exist, someone could easily write a version of screen.h for the MM-1. I firmly believe that if we in the OS-9 community are to survive, we must all strive to write code that can be easily ported to the various machines running OS-9. If we do not, we will splinter off into several even smaller groups running "Machine Specific OS-9" and that will be a loss to each of us. I keep this uppermost in my mind as I try to write programs in C for my two machines. I hope you will too.

Questions or comments regarding this article may be addressed to me at the following:

David Wordell
833 Woodhaven Ln.
Grand Prairie, TX 75052

/* Screen control definitions and macros for OS-9/68000
Ver. 2.4 on a PT68K4.

All control codes are in hexadecimal form */

/* NOTE: DO NOT append a semi-colon in the

```

program! */
/* Use on a single line as: "CLS" or
"CUR_OFF" */

/* Sound Bell */
#define BEEP    putc('\x07',stdout);

/* Backspace, destructive */
#define BK_SPC  putc('\x08',stdout);

/* Horizontal Tab */
#define TAB     putc('\x09',stdout);

/* Line-feed */
#define LFEED   putc('\x0A',stdout);

/* Cursor up */
#define CUR_UP  putc('\x0B',stdout);

/* Forward space */
#define FOR_SPC putc('\x0C',stdout);

/* Right Cursor */
#define CUR_RGT putc('\x1B',stdout);
putc('\x5B',stdout); putc('\x43',stdout);

/* Cursor down - no scroll */
#define CUR_DWN putc('\x1B',stdout);
putc('\x5B',stdout); putc('\x42',stdout);

/* Carriage Return (cursor to col 1) */
#define CAR_RET putc('\x0D',stdout);

/* Clear Screen, Home cursor */
#define CLS     putc('\x1A',stdout);

/* Home cursor */
#define CUR_HOME putc('\x1E',stdout);

/* Insert line at cursor row */
#define INS_LIN putc('\x1B',stdout);
putc('\x45',stdout);

/* Insert space at cursor */
#define INS_SPC putc('\x1B',stdout);
putc('\x51',stdout);

/* Delete line at cursor */
#define CLR_LIN putc('\x1B',stdout);
putc('\x52',stdout);

/* Clear from cursor to end of line */
#define CLR_EOL putc('\x1B',stdout);
putc('\x54',stdout);

/* Delete character at cursor */
#define DEL     putc('\x1B',stdout);
putc('\x57',stdout);

/* Clear from cursor to End of Display */
#define CLR_EOS putc('\x1B',stdout);
putc('\x59',stdout);

/* Blink ON */
#define BLINK_ON putc('\x1B',stdout);
putc('\x5E',stdout);

/* Blink OFF */
#define BLINK_OFF putc('\x1B',stdout);
putc('\x71',stdout);

/* Reverse Video ON */
#define REV_ON   putc('\x1B',stdout);
putc('\x6A',stdout);

/* Reverse Video OFF */
#define REV_OFF  putc('\x1B',stdout);
putc('\x6B',stdout);

/* Position Cursor */

/* Positions the cursor at the screen column and
row passed.
Variables col and row are passed as integer
variables and
must be in the range: col (0 - 80),row (0 - 25)
*/
pos_cur(col, row) /* Position the cursor to col
and row */

int col, row; /* Variables passed to the function
*/
{
int cola, rowa; /* Left digit */
int colb, rowb; /* Right digit */

if ((col >= 0 && col < 81) && (row >= 0 &&
row < 26))
{
cola = (col/10); /* Extract the left digit
*/
colb = (col - (cola*10)); /* Extract the right
digit */
rowa = (row/10);
rowb = (row - (rowa*10));

putc('\x1B',stdout); /* Put Escape code */
putc('\x5B',stdout); /* Put [ code */
if (row > 9) putc(rowa+48,stdout); /* Is
there left digit? */
putc(rowb+48,stdout); /* Put right digit
decimal code */
putc('\x3B',stdout); /* Put ; code */
if (col > 9) putc(cola+48,stdout);
putc(colb+48,stdout);
putc('\x48',stdout); /* Put H code */
}
}

```

INTRODUCING THE MM/1B!

A new machine based on a board produced by Kreider Electronics featuring :

- o 16 bit PC AT I/O Bus with 5 slots
- o MC68306 CPU at 16.67 MHz
- code compatible with 68000
- 2.4 MIPS
- o 0.5MB to 16MB DRAM (4 SIMM sockets)
- o IDE Hard Disk Interface (2 drives max)
- o 1.44MB Floppy Interface (2 drives max)
- o 2 16 byte FIFO serial ports (up to 115K baud)
- o Bi-directional parallel port
- o On board RS232 buffers
- o Battery Backed Real Time Clock
- o AT Keyboard Interface & Standard AT power connector
- o Baby AT Size Footprint
- o BASIC (resembles Microsoft Basic)
- o MGR - graphical windowing environment full documentation!
- o "Personal" OSK V3.0 (Industrial with RBF)
- Display drivers: Tseng 4K, generic inexpensive VGA
- SCSI card support: Future Domain 1680 & Adaptec AAH 15XX
- o OSK version 2.4 including network file manager, PCF, SCF, SBF, RBF, Pipeman RamDisk, MW C Compiler version 3.2 with r68/168, MW Basic, MW Debug, MW Programmers Toolkit (Mail, print spooler UMac)
- o UUCP package from Bob Billson
- o Ghostscript (PostScript interpreter)
- o Many other utilities and tools

Pricing as low as \$400!

(motherboard, Personal OSK, & MGR, no RAM)

**Call or write for quote
on full systems.**



**BlackHawk
Enterprises, Inc.**

P.O. Box 10552

Enid, OK 73706-0552

Phone 405-234-2347

Internet: nimitz@delphi.com

<268'm>

Recently on the Delphi CoCo SIG we had a very active discussion in the forum of different disk drives and the media for same, and how one would hook various drives to a CoCo, and what media one should use in them. Although our discussion is centered on the CoCo, much of the information applies to all computer types. In all cases, the OSK machines will use drives that the CoCo will not.

Let's talk about 3.5" drives

There are currently three common kinds of 3.5" drives in the world of PC compatibles. First is the 720K 3.5" drive, which accepts 720K 3.5" diskettes. These diskettes can be recognized in a number of ways. They often will call themselves "1 MB" diskettes, which signifies that their UNFORMATTED capacity is about 1 megabyte. After formatting in typical CoCo or PC compatible style, they will hold 720K of data. 720K 3.5" diskettes are distinguished from 1.4 meg diskettes by their NOT having a second "media type hole" punched at the edge opposite the write protect hole.

The diskettes have 80 tracks, and the data transfer rate is the SAME as for CoCo-style 360K drives: 250K bps. This sort of disk drive can be physically hooked to an ordinary Color Computer controller. All that is needed is an appropriate data cable (with proper connectors) and power supply connectors. These drives, unlike most 360K 5.25" drives, tend to all have a soldered-in 1000 ohm terminator resistor, so you do NOT need to worry about adding or removing terminator resistor paks with these drives. If you use one with a 5.25" drive, you should put a terminator resistor in the 5.25" drive. If possible, I suggest using a 220 or 330 ohm terminator resistor pak in this situation, instead of the usual 150 ohm pak.

ADOS (still sold and supported by SpectroSystems of Miami, Florida) supports such 720K drives under DECB (making each 720K drive into two

virtual 360K drives), and OS-9 can support 720K drives as 720K double sided drives.

720K 3.5" drives have become "obsolete", and the media for them is less and less commonly offered for sale. This is because they have been replaced in the world of the PC by the current standard, 1.4 meg drives and diskettes.

NOTE: These drives and diskettes are commonly and INCORRECTLY referred to as "1.44 meg". Their actual data capacity is 1.41 megabytes, and I would encourage folks to refer to them as "1.4 meg" drives and diskettes, which more correctly describes their data capacity. The reason they have been called "1.44 meg" devices is due to an arbitrary and INCORRECT re-definition of the word "megabyte" by those who introduced the drives. They defined, for purposes of marketing these drives, a megabyte as 1000 (base 10) Kilobytes. This is simply incorrect. A megabyte is 1024 kilobytes.

The 1.4 meg 3.5" drives and diskettes are 80 track devices, just like the 720K devices. However, they put twice as much data on a given track. Thus, for every revolution of the diskette, they can send or receive twice as much data. This results in a speed of data transfer twice that of the older drives, or 500K bps.

Ordinary Color Computer controllers cannot handle such data transfer speeds, and so cannot use 1.4 meg drives AS 1.4 meg drives. However, these drives are all also capable of reading and writing 720K type diskettes. In most modern situations in a PC compatible, these disk drives figure out whether to behave as 720K drives or as 1.4 meg drives based on whether or not they "see" the "media hole" in the diskette you put in them. Thus, MOST 1.4 meg drives can be hooked to a normal CoCo controller, and IF you feed them 720K diskettes, the controller will happily be able to read, write, and format diskettes in that drive.

Some early 1.4 meg 3.5" drives

(especially some used in IBM brand PS2 computers) did media type selection a bit differently. These required a signal on the data cable (often pin 34) to set them for 1.4 or 720K media.

Some 1.4 meg drives have jumpers that allow them to be set to either "look at" the media hole to determine disk drive type, OR to look at the signal on pin 34 of the data cable. Such drives often have an additional jumper to set the POLARITY of the signal on the data cable when they are set to receive a media type signal via their data cable.

Note that you can usually tell the difference between a 720 and a 1.4 meg floppy drive by carefully looking into the slot of the drive. If you see a microswitch sensor or LED-detector system on BOTH extreme edges of the slot, you have a 1.4 meg drive. If the drive only has such a detector arrangement on the side of the diskette that has its write protect hole, then you have a 720K drive.

What really IS the difference between 720K diskettes and 1.4 meg diskettes? The magnetic media on them is somewhat different. 720K diskettes use media with a magnetic coercivity of 600 Oersteads, and 1.4 meg diskettes use media with a magnetic coercivity of 720 Oersteads. These 1.4 meg diskettes, therefore, use media that is MORE RESISTANT to being magnetized, and require somewhat higher WRITE CURRENT by the disk drive to be properly written to.

If you can't get hold of a 720K diskette, you SHOULD be able to use a 1.4 meg diskette in a 720K drive AS a 720K diskette. The small difference in magnetic coercivity may make for slightly less reliable data recording, but you are not likely to notice the difference in most cases.

An interesting question came up on the Delphi CoCo SIG. Lonnie McClure asked: Why can't one use a 1.4 meg 3.5" drive with a 1.4 meg type diskette with a Color Computer type controller as a 720K drive system, and get a resulting 720K diskette (since the CoCo

is sending data at 250 BPS). He argued that it would be DESIREABLE to use 1.4 meg media in 1.4 meg drives as 720K diskettes with the Color Computer, because then the write current of the drive head will be better suited to the media type, EVEN when one is only writing lower density tracks.

I'm not sure of the answer to this question, and it bears some experimenting. I do know that the difference between 600 and 720 Oerstead magnetic coercivity is not that great, and if one does use a 1.4 meg diskette in a 720K drive, it likely will work reasonably well. Folks have been punching holes in 720K diskette and using them as 1.4 meg diskettes on PC compatibles for years, and MOSTLY this results in REASONABLY reliable function (even tho the media IS somewhat different, and not perfectly suited to the 1.4 meg drives).

The third current variety of 3.5" diskettes and drives are the 2.8 megabyte variety. The 2.8 megabyte system is also an 80 track system, which manages to put FOUR times the amount of data PER TRACK on the diskette. The media for these drives is a special barium ferrite material (as opposed to the cobalt ferrite material used for making 720 and 1.4 meg diskettes). This media has a magnetic coercivity of 750 Oersteads. But it differs more significantly from 720 and 1.4 meg diskettes in that the media is VERTICALLY POLARIZED, which is what facilitates the denser recording. These drives yield a data transfer rate of 1 meg bps. 2.8 meg floppy drives are supported by modern PC Bios ROMs, and the drives are actually fabricated with two separate heads so they can read and write 1.4 and 720K diskettes too, but they have not caught on much, due to their considerably greater expense for the media and for the drive. It's likely that they will never catch on that much, for modern technology seems to be heading for removeable media systems that hold 20, 25, or up to 100 megabytes, where the drives and media are physically similar in size to 3.5" diskettes.

Note that 2.8 meg drives use the same general sort of media (vertically polarized barium ferrite) as do the 20 meg "floptical" drives. But floptical drive media has an optical positioning track to allow much greater track density (and their media may be more fine, as well).

20 meg floptical drives can also read and write 1.4 meg and 720K diskettes, because they have a second physical head inside them. But 20 meg floptical drives cannot read or write 2.8 meg floppies. They'd need a THIRD head to be able to do that! Also, note that floptical drives can read and write 720K and 1.4 meg diskettes about four

times as fast as can an ordinary 1.4 meg or 720K disk drive. Floptical drives may, however, not become a standard either, for they are being challenged by other high capacity, removeable media systems that offer more data capacity for less money. Most floptical drives I've seen use a SCSI interface to hook to a give computer system.

Neither 1.4 nor 2.8 megabyte floppy drives can be used with ordinary Color Computer disk controllers as 1.4 or 2.8 meg drives. 1.4 megabyte drives CAN be use with a CoCo under OS-9 as 1.4 megabyte drives IF you are running a system that has a SCSI host adaptor talking to a stand alone SCSI hard and floppy controller that supports 500K bps data transfer rate on the floppies, and have appropriate drivers for the system. There was only one system like this ever made, by Hemphil Electronics. This was some years ago and they weren't very popular due to high cost.

What about 5.25" drives?

The 5.25" 360K and the 5.25" 1.2 meg diskettes are very different beasts. The 360K drives are 40 track, and use media that has a magnetic coercivity of 300 Oersteads. 360K drives send and receive data at 250K bps. The 1.2 meg diskettes are 80 track, and have a magnetic coercivity of 600 Oersteads. 1.2 meg drives send and receive data at 500K bps. 1.2 meg drives, therefore, CANNOT be used with a normal CoCo disk controller. Diskettes for 1.2 meg drives CANNOT be reliably formatted or written to by a 360K drive.

Now, there did briefly exist a very odd sort of 5.25" drive, that had 80 tracks and a 720K formatted data capacity. This drive was electrically identical to the 3.5" 720K drive. It COULD be used directly with Color Computer disk controllers. Very few of these drives were actually made and used. They did find their way into a number of late vintage CPM machines, and even were used in the very vaguely PC compatible Tandy model 2000. These drives use the same disk media as 360K drives. For a while, OS-9 users were buying up these drives surplus, and using them with normal CoCo controllers as 720K OS-9 floppy drives. Some OS-9 users still use this type of drive. Heck... I just sold a few that I salvaged from a Tandy 2000 to some OS-9 users on the Delphi CoCo SIG.

To further confuse the issue, 1.2 meg diskettes can be told to decrease their head write current and be able to write to 360K 5.25" diskettes. In doing this, you MUST NOT use diskettes that were formatted in a real 360K drive, because the formatted track will be too wide and you can get data

corruption! You must in this situation start with a totally BLANK (wiped clean with a bulk tape eraser) diskette, THEN format the diskette in a 720K or 1.2 meg drive as a 720K or 360K diskette. When trying to create a 360K-like diskette with a 720K or 1.2 meg drive, one "double steps" the head, moving the head two steps for each "track". Double stepping is used to allow a 1.2 meg or 720K 5.25" drive to READ 360K diskettes. Problems arise here because some 1.2 meg drives rotate their media at DIFFERENT SPEEDS when being used with high and low density media! With 1.2 meg diskettes, the drive rotates at 360 RPM. MOST 1.2 meg drives also read and write 360K type diskettes at 360 RPM, which is a DIFFERENT SPEED (and hence different data transfer rate) from a normal 360K drive, which rotates at 300 RPM. Many 1.2 meg drives, however, have JUMPERS on them that can be set (or resoldered) to tell the drive to slow down to 300 RPM when working with 360K or 720K diskettes. One needs elaborate and detailed manufacturer's data sheets for the particular make and model drive one is using to be able to know whether this is an option, and how to set it.

Note that 720K and 1.2 meg drives both have 80 tracks, and have the SAME TRACK WIDTH, which is HALF THAT of a 360K 40 track drive. But 1.2 meg drives have twice the data density per track, and rotate 1.2 times faster than 720K and 360K drives when used as 1.2 meg drives. The difference in number of tracks on the media, and consequently of physical track WIDTH is the source of all sorts of compatibility problems between 5.25" drives that do not afflict 3.5" drives, which all have 80 tracks per diskette... tracks that are all of the same width. The differences in media rotation rate among 5.25" drive types is also a source of compatibility problems until one fully understands what is going on and choose appropriate drives and jumpers them properly. 3.5" floppy drives all rotate at the SAME rotational speed.

Note that, unlike the later 3.5" diskettes, 5.25" diskettes did NOT have any unique identifying characteristic by which the drive could tell the difference between 1.2 meg style and 360K style media. Unless the label on the diskette clearly identifies the media type (usually by saying "double density" for 360K diskettes and "High Density" for 1.2 meg type diskettes) there's no clear way to tell the difference. It has been brought to my attention that NO ONE I know has ever seen a 1.2 meg 5.25" diskette that has a hub re-enforcement ring on it, while SOME 360K diskettes have this ring and others do not. Thus, if you have a

5.25" floppy diskette that HAS a hub re-enforcement ring, it's HIGHLY LIKELY (I can't yet say "certain") that it is a 360K type diskette. The 1.2 drives use one of their datapins to select high or lower density media (higher or lower write current, and 300 vs 360 rpm rotational speed if they are properly jumpered for that).

Some bits of history

The first floppy disk in common use was the 8" floppy. This existed in increasingly high density versions, up thru a 1.2 megabyte variety that had 80 tracks and two sides. These 8" floppy disks were given highly standardized formats under CPM, the dominant operating system for personal computers until 1981.

When the 5.25" diskette came out, makers of 5.25" drives went to Digital Research (makers of CPM), and requested that that Digital Research assign a standard disk drive format for CPM on 5.25" diskettes. The story goes that the late Garry Kildale (founder of Digital Research and creator of CPM) sneered at these folks, saying that 5.25" drives will always be an obscure toy... the only real drives were 8 inchers. So the makers of CPM computers went to their respective copmany sites, and each created their own unique "CPM disk format" for 5.25" drive - based CPM computers. Before the death of CPM at the hands of the IBM PC and MS-DOS there were created literally THOUSANDS of different 5.25" CPM diskette formats (of which about 400 can be read on a PC compatible using the program Xenocopy, distributed by Xenosoft of 2210 Sixth St. Berkeley, CA 94710 (510) 644-9366).

The 1.2 meg "PC/AT style" 5.25" floppy drive was created to be electrically identical to the 1.2 meg 8" floppy drive, which was the last and most advanced 8" floppy drive system distributed. It put 15 512 byte sectors per track onto each of its 80 tracks. By the time the 3.5" high data transfer rate drives were introduced, they decided they could happily and reliably fit 18 512 byte sectors per track, and so those diskettes were standardized to support 1.4 megabytes of data instead of the 1.2 supported by the 5.25" drives.

When "shirt-pocket sized" diskettes were first being developed, there were four sizes produced: 3.0 inch, 3.25 inch, 3.5 inch, and 3.9 inch. The 3.5 inch size rapidly became the industry standard, and the others became footnotes to computer history, and soon became virtually forgotten. SOME Color Computer users, tho, may remember the Amdek 3.0 inch 360K drives and diskettes, which for a while were sold and used in the

Color Computer market. These were also used by Brother and some other typewriters and dedicated word processors. The disks don't have a sliding cover over the slot. Instead, they use a sleeve as do the 5.25" disks. This type disk can still be found in larger office supply stores (or special ordered), though most typewriters today use the more common 3.5" disks.

George Morrow (creator of some of the finest of the CPM machines, under the companies "Thinker Toys" and later "Morrow Designs"), when first told about the swing to "shirt pocket sized" diskettes, is allegeded to have snorted "Why not just get all the tailors to make shirt pockets a little bigger, and continue to use 5.25" floppy diskettes?"

While overall the newer 3.5" media style is more desirable, due to its smaller dimensions and more rugged plastic shell and re-settable write protect system and media ID hole system, I kind of like the old 5.25" diskettes, for they could be stacked so much more closely together. I also like the old 5.25" diskettes because you could conveniently STAPLE one to a letter you were sending. If you were careful to staple it in the extreme corner, such stapling would not damage the media, and the diskette would be perfectly readable, tho the receiver might get so freaked out about the staple as to not take the time to realize this.

Along the way, some obtuse and obstinate

"I did it my way" companies elected to take a "more advanced" approach to floppy drives and incorporate into their products some (now to me bizarre) newer alternate size "micro mini" diskettes. Thus, there were made and used 2.5 inch and 2.9 inch type floppy diskettes and drives. This included machines like the Zenith SuperSport lap computers, and the Tandy WP2 dedicated word processor.

Other oddities among floppy drive footnotes include one drive that wrote a single track on each side of the diskette in a SPIRAL fashion (like the groove on a vinyl 12 inch LP audio record, or like the data recording track on a modern audio CD or CD ROM), and a 3.5 inch 67.5 TPI floppy drive used in an EPSON CPM-based laptop computer.

If any readers have at home any of these odd size sub-5.25 inch diskettes or drives that they no longer wish to keep, I would greatly appreciate their sending them to me (I'll re-imburse you for nominal shipping costs). My interest here is in helping my friend, Fred Cisen of Xenosoft (who is my teacher about disk drive formats) increase his collection of odd-ball diskettes and drives, to assist him in the classes he teaches on this subject.

Happy Hacking!



Summary: 3.5" diskettes			
Formatted Capacity:	720K	1.4 meg	2.8 meg
Magnetic Coercivity:	600 Oerstead	720 Oerstead	750 Oerstead
Media Coating:	Cobalt Ferrite	Cobalt Ferrite	Barium Ferrite
Polarization:	Horizontal	Horizontal	Vertical
Data Xfer Rate:	250K bps	500K bps	1 meg bps
Number of Tracks:	80	80	80
Tracks Per Inch "TPI":	135	135	135
CoCo Compatible:	YES	YES	Probably can be used as a 720K drive but what a waste!
		can be used as a 720K drive at 250K bps data xfer rate.	

Summary: 5.25" diskettes			
Formatted Capacity:	135K	360K	1.2 meg
Magnetic Coercivity:	300 Oerstead	300 Oerstead	600 Oerstead
Media Coating:	Cobalt Ferrite	Cobalt Ferrite	Cobalt Ferrite
Polarization:	Horizontal	Horizontal	Horizontal
Data Xfer Rate:	250K bps	250K bps	500K bps
Number of Tracks:	35	40	80
Tracks Per Inch "TPI":	48	48	96
CoCo Compatible:	YES	YES	Some models can be used as a 360K drive at 250K bps data xfer rate.

The Chicago CoCoFest!

James Jones & Brother Jeremy

Another weekend in the Windy City for the CoCo, OS-9, and OSK!

First, a disclaimer—I didn't take notes, and what I noticed is a function of my personal interests and erratic-at-best observational abilities. The following is going to reflect that, and if I've left anything or anyone out, it's due to my lapses, not by intent. OK?

That said, here's my Fourth Annual "Last" Chicago CoCoFest saga. It started long ago in a state far, far away, with two young men amazed that they'd rented such an atrocious video: *Luther the Geek*.

Some years later, one of the two, Boisy Pitre, came to work for Microware and found (1) a fan of bad movies among his new coworkers and (2) mirabile dictu, a copy of *Luther the Geek* for rent at a Hy-Vee store. He and the coworker—me—watched the movie and somehow managed to see through tears of laughter as the debits rolled by that (1) the movie is an example of Your Tax Dollars at Work, being a result of the University of Iowa Film Production Project, and (2) the movie was filmed in Stirling and Rock Falls, Illinois. Boisy realized that Stirling was on the scenic route that we took to the Third "Last" CoCoFest in 1994...and it was clear that this year would be a Pilgrimage of sorts.

The weather started out gray, but cleared as we entered Illinois, and not long afterward we stopped at the Casey's near the edge of Stirling. I'm constitutionally incapable of keeping a straight face, so I let Boisy ask the lady at the counter. She seemed reluctant to admit that Stirling was the home of *Luther the Geek*—but admitted it, and explained to her younger coworker. For a second she was incredulous that we'd come to Stirling because of the movie, but we assured her this was the path we'd taken for other purposes, and that *Luther* was just icing on the cake, or perhaps I should say the batter on the chicken.

We asked her what store a certain scene was filmed in, and she told us Kroger's. That was our next stop, and with a little additional help on the way, we found it. An employee there knew about the movie, but didn't seem terribly enthusiastic. We visited the dairy case, and then went on our way.

The best thing about the Fest—meeting friends—started upon our arrival. (I had my first view of Frank Swygert's 1963 Rambler, a beautifully-maintained car.) The Hawkses kindly helped us get into our room, and after dinner, I gave an impromptu demonstration of recursive descent parsing.

I was anxious to buy a CD-ROM drive, but Boisy was more so, and he had his installed Friday night! More familiar faces appeared as the evening went on.

Mostly technical things that stick out in my mind:

1. The 68306-based computer, running and on display.

2. The HAWKSoft CD-ROM file manager—they sold out of drives in short order. There are LOTS of CD-ROMs containing source code that is begging to be ported out there...

3. Planet Engine running on the MM/1.

4. LA-Term. (*Very* nice looking terminal program.)

5. Ongoing optimizations in NitrOS-9.

6. Brother Jeremy's ongoing work to revive abandoned software...and his musical plea to Kevin Darling (not to be confused with a certain Conway Twitty song, despite having the same first two words). This and the Planet Engine port clearly display that talking to the author and/or vendor of software no longer sold is not only preferable to rationalizing the piracy of so-called "orphanware," it has better results.

7. Chris Dekker's very neat graphics displays (and his setup that uses a serial link to a PClone to let the CoCo use the PClone's hard drive).

8. The continued efforts of vendors and the OS-9 Users Group—they deserve your support.

9. The Iomega Zip drive that Dave Pellerito had hooked up to his MM/1. \$200 for an outboard SCSI drive that takes removable media about the size of a 3.5" floppy but holding 100 Mbytes. About 30 ms access time, and the medium is about \$20 a shot. (Maybe it's time to sell my floptical...)

10. Frank Swygert's edited and updated printing of Paul Ward's *Start OS-9* under the title *Mastering OS-9 on the Tandy Color Computer 3*. It's a book well worth having, and the only thing I wish were different is the binding—like the Tandy OS-9 manuals, it's a bit too small for the amount of paper therein.

This is far from a complete list of all the cool stuff that was there to be had—heck, it doesn't even list all the things I bought (though I didn't buy everything listed; if nothing else, the Zip drive wasn't for sale). I'll have to leave it to others to do a more complete report. In any case, it seemed to me that there were as many people and vendors as last year, and I hope to be in Elgin once again next year. (I know I'll be in Atlanta this October.)

Also omitted above is a lot of pleasant conversation and dining with old friends—all of whom I know better than I know the people who live in the condos next to mine. Thanks, and I hope to see you regularly.

James Jones

jjones@delphi.com

My dear friends:

Once again another Fest has come to a close. It was a wonderful time, and while I am certain that Allen Huffman will once again post his Fest review, I would just like to comment that, from my humble perspective, it was a success. There was new hardware and software.

Rick Ulland of Conect debuted the Fast232 serial port, and Northern Exposure had an AT Keyboard Adapter. Nitros9 1.21 was available. And Chris Dekker, an extremely talented programmer, who does a lot of work in Basic09 had several excellent pieces of software, including BasicBoost, which speeds up Basic09/RUNB and ScreenBoost. ScreenBoost, is a "monk-must". It greatly speeds up GRFDRV and does such neat things as horizontal scrolling on type 2 text screens. There were lots of other new things for both OS-9 and OSK. Please forgive me for not mentioning all of you by name, but it is somewhat late for me as I am writing this.

On a more personal note, it was certainly a great joy for me to be able to attend. It is always so wonderful to see so many old friends again, and get to put names and faces together of new ones.

I would be remiss if I did not publically offer my heartfelt thanks to all who put in so much time and effort to make this fest possible. Glenside, and in particular, Tony Podraza, did an outstanding job.

Finally, thank you to all of you for the many acts of kindness which you showed me once again. The warmth of your greetings, the conversations, and the special times we had together are a blessing to me. Your friendships are very precious to me. Thank you for allowing me to be with you and for sharing a part of your life with me.

With all best wishes,

Brother Jeremy, CSJW

OS9 Users Group Treasurer

revwcp@delphi.com

Programming in "C"

Henry Harwell

Some notes on Microware's C compiler

Editor: The last column was the end of P.J. Ponzio's little C tutorial. It should get one good started in the world of C programming. To complete our C programming series, here are some tips on using Microware's C compiler on the CoCo from Henry Harwell.

Just to make things interesting, I have printed Henry's entire letter to me. I did this to let everyone know that any of you can write a decent article! Many thanks to Henry for "taking the bait" and coming up with an excellent article!

Dagnabbit all, Frank:

You shamed me into writing Craig Wheeler about getting Cup and running (letter enclosed), based on your nudging comment to his Letter to the Editor in the most recent 68' micros. All is fair in love and squeezing copy out of turnips?

I don't know if this is what you want (or even what you said I promised to do). My style tends toward chatty, and your space is at a premium. This letter contains both hints and history, so perhaps these should be separated. Perhaps any <opinion> needs to be excised altogether...

Henry O. (Hank) Harwell

Editor: Well Henry, perhaps all IS fair in squeezing GOOD copy out of turnips! Your article does prove you are FAR from a turnip though!

Mr. Wheeler's Letter to the Editor in the March issue reflects frustration -- even a sense of abandonment -- in getting the Microware C Compiler up and running! Hence this article (noting the prod from Frank Swygert). While you may have gotten some help already, I'd like to offer the following tips and thoughts out of comradeship, since I've been there too!

The Tandy manual, or users guide, is paltry (the pages are falling out of mine) -- but note the author's statement on pages 1-1 & 1-2 referring the user to "The C Programming Language" by Brian Kernighan and Dennis Ritchie

(published by Prentice Hall). K&R, as it is called, is the Bible of the C programming world.

Tandy was always tepid in marketing the CoCo, and our compiler began life as a Raggedy-Ann port over to Tandy's "robust" environment for OS-9 -- the 64K Color Computer (Andy?). If the marriage was a bit hasty, maybe the matchmakers were in a hurry. I believe they were, fearing how soon this window of opportunity might shut.

If you want the MW compiler to hum, buy K&R. It has come out in reprint several times, selling for around \$31. You cannot go wrong in buying it; just make sure it is the original version. The ANSI version can be used with our stock package, but refers to a later implementation of the C compiler (the Microware version we have seems to be 'late pre-ANSI', a significant distinction in the world of C). Let me know if you cannot find a copy back there -- check used technical book stores and college bookstores. Big chains like "BookStar" or "Waldens" can probably get it for you new.

Actually, Microware didn't really intend for the users guide or compiler to be especially user friendly: the compiler's main purpose was as to develop other packages to run under the company's (then) relatively new operating system. The users guide (UG) is basically a technical reference manual to the compiler for seasoned programmers. I would recommend that you at least familiarize yourself with chapters 1 and 4 (standard library) of the UG; they're not bad reading if you've got some other books to teach you C. Look through the appendices: you'll find connections in Appendix C to Basic09 and to Assembler in Appendix D. No matter which implementation of C you settle with, you're going to need to know most of this stuff, especially the standard C library.

Getting Into C Programming

I'm going to assume that you have two disk drives and at least 512K RAM.

If you are using floppies (with or without a hard drive), make sure you have them formatted appropriately for your system. Of course you will never, ever use your factory diskettes to do actual work. Instead, make a lot of backups, using backup or dsave. Get to know dsave, it can be a real friend.

The C library disk is needed during program compilation: put it in /d1 and leave it there throughout your programming session. The compiler is hard coded to look for the various library files it needs in /d1.

As far as the C compiler modules are concerned, you will need them all. My suggestion is to dsave then into a CMDS directory on a RAM drive, and make this your execution directory during compilation (chx /r0/cmds). You can still operate totally from floppies (as I did when first starting), but it is MUCH slower. In this case, your compiler's CMDS directory would be in /d0 and your execution directory would be /d0/cmds.

Which modules do you really need? There is an ANSI front-end out there for the compiler which I believe drops the need for one of the modules. I haven't used this much, but it is worth getting, because it will allow you to do "just about all" of the things that you could do using compilers under MS-DOS, including declaration of function prototypes. Perhaps this is old news by now though.

At this point I have failed to mention one thing: how utterly important it is to get a good text editor. I use Bob van der Poel's Ved. I know that there are others, but none better in my view. Tandy's edit zapped my interest in OS-9 for nearly three years! Whatever editor you choose, get one that you can use, because it's going to be a constant ally or foe as you learn to write and rewrite C code. There are even beautifiers to properly indent your code if you are interested. At \$30, you can't go wrong buying Ved.

Writing the Program

You should be about ready to put

some code together. When we say that C code is compiled code, recall how radically this differs from BASIC (if you are familiar with assembly, you are way ahead of me on this). Namely, the compiler uses several steps to process, optimize, assemble, and link your code to external functions and utilities for a unitary program.

What this means is -- put your library disk in /d1 and keep it there (I speak from experience). Also, at compile time, your compiler is going to be looking for a filename you provide with a ".c" extension. If you forgot the extension you'll error out (although rather benignly -- just rename filename to filename.c). The compiler will be looking for the file in your working directory, so make sure it's there (this can be /r0/src if you are using the RAMdisk). You invoke the compiler as follows: "cc1 filename.c" <ENTER>.

Assuming the compiler modules are all in your current execution directory, the compiler should step through several stages with linking the last and the cursor returning to you. If you have given the compiler no silly syntactical reasons to reject your program (you code -- not you!), you should get an executable module deposited in your execution directory.

If the compiler doesn't work the way you think it is supposed to, it's probably because *that's what you think*. For instance, in the code below for "hello", I got about three syntax errors for not putting the newline escape character where it belongs, i.e., inside the quotes with the text string to be printed:

```
printf ("Hello, World," \n);  
printf ("Hello, World, \n");
```

The first line failed, but the second went through the compiler just fine. Make sure to watch for the semi-colons and matching ellipses, right after main{} and at the end of the program.

During execution, your compiler creates and apparently deletes temporary files during execution. If it errors out, you will be left with files that can interfere with any subsequent compiles. With this version you may be stuck with deleting the temp files manually, as I do. You'll see them as ctmp.3.m and ctmp.3.i. To these add c.com, which

may not be absolutely necessary to shed. One could probably write a stripping routine to get these guys out automatically (the ANSI version seems to do just that).

If you're like me, you'll create source code files with minor variations in them to test what the compiler is doing or will accept -- or what you are doing. That's good, and I would heartily suggest you plan out beforehand how you'll keep the filenames straight: i.e., file1a, file1b, file2a, etc. for a major change. You may become so exasperated during a session that you'll be glad to have a file designation scheme to fall back on.

Eventually, your data/execution directories are going to fill up. Before that comes close to happening (use free for this), I suggest you do some house cleaning and save what you know you will need. I prefer to copy successful code and modules en masse to floppies. Just remember that mistakes happen and redundancy is advisable!

Here is an example of some source code that should work, taken right from K&R:

```
#include <stdio.h>  
main ()  
{  
    printf ("Hello, World. \n");  
}
```

Supposing you save this in your data directory as "hello.c", you will invoke your compiler with "cc1 hello.c". Generally, if you get through the compiler's second pass (c.pass2) you're going to have a successful compile -- generally. By the way, "cc2" had been planned but was scuttled.

Debugging

As you may know, when you write more complex code, it can prove helpful to step line-by-line through your code to make sure what's happening. Depending on your programming style and love of C, you'll probably want to have access to several good books on C. You can use the UG as a technical reference to see what commands like getc are up to. You will make mistakes, but the more you make, you'll develop a better intuitive grasp about what the compiler will and will not allow. Remember, this is a learning situation.

The power that C bestows is worth the extra effort.

Summary

In terms of getting the compiler up and running, and getting some code out of it, that's about it. At this point, let me summarize:

Library disk in /d1;
C CMDS directory is current execution directory;
Save programs to current data directory;
"cc1 filename.c" invokes compiler; six step process;
Compiled executable deposited in execution directory;
You're done!

When Debugging:
Where is library disk?
Be very picky about syntax errors;
Did you clean out temporary files from previous compile?

No obvious coding errors, but program won't print longs, floating point integers, or doubles -- MUST call pflinit() or pffinit() -- UG page 1-7;
Won't link, but otherwise OK -- check available space

Conclusion

I hope this helps. If you know assembler, you'll be way ahead of the game when it comes to pointers. You'll have to suffer with Typedef and Struct: something fishy there. Just keep trying to develop phrasing that works for you. C allows brevity, if you can master it. I don't know what I would have done without the help of Zack Sessions (Color Systems). At the start, it's hard to know just what to expect.

You're going to need some decent textbooks and references about the C language, rather than so much about the C compiler. The Mix series is good (Power C, Workout C, etc.). The company sells an inexpensive compiler along with other tools that are super values. When I enrolled at a community college for C programming, I also got the knowledge and savvy of a real professional. Something to consider.



244 S. Randall Road • Suite #172 Elgin, IL 60123
 (708)742-3084 eves & ends • MO, Check, COD; US funds
 Shipping included for US, Canada, & Mexico

MM/1 Products (OS9-68000)

COMING SOON!!! CDF - CD-Rom file Manager! Unlock a wealth of files on CD with the MM/1!!

VCDP \$50.00 - New Virtual CD Player allows you to play audio CDs on your MM/1!! Graphical interface emulates a physical CD player. Requires SCSI interface and NEC CD-Rom reader.

KLOCK \$20.00 - Optional CUCKOO on the hour and half hour!! Continuously displays the digital time and date on the /term screen or on all open screens. Requires I/O board, audio cable, and speakers.

WAVES vr 1.5 \$30.00 - Now supports 8SVX and .WAV files!! Allows you to save and play all or any part of a sound file. Merge files together or split into pieces. Record, edit and save files with ease. Change playback/record speed. Convert Mono to Stereo and vice-versa!! Record and Play requires I/O board, cable, and audio equipment.

SOUND CABLE \$10.00 - Connects MM/1 sound port to stereo equipment for recording and playback.

GNOP \$5.00 - GNOP is the AWARD-WINNING version of PONG(tm) exclusively for the MM/1!! You'll go crazy trying to beat the clock and keep that @#\$%& litt... ball in line! Professional Pong-ists everywhere swear by (at) it!!! Requires MM/1, mouse, and lots of patience.

Coco Products (DECB)

HOME CONTROL \$20.00 - Put your old TRS-80 Color Computer Plug 'n' Power controller back on the job with your Coco 3!! Control up to 256 modules, 99 events!!

HI & LO-RES JOYSTICK ADAPTER \$27.00 - Tandy Hi-Res adapter or NO adapter at the flip of a switch!!

KEYBOARD CABLE \$25.00 - Five foot extender cable for Coco 2 and 3. Custom lengths available.

MYDOS \$15.00 - CUSTOMIZABLE! EPROM-ABLE! The commands Tandy left out. Optional 6 ms. disk drive speed. Supports double-sided and forty track drives. Set CMP or RGB palettes on power-up. Power-up in any screen. Speech and Sound Cartridge supported. Point and click mouse directory and MORE. For all Coco 3 with Disk Basic 2.1. More options than you can shake a joystick at!!

DOMINATION \$18.00 - MULTI-PLAYER STRATEGY GAME! Battle other players armies to take control of the planet. Play on a hi-res map. Become a Planet-Lord today! Requires CoCo 3, one disk, and joystick or mouse.

SMALL GRAFX ETC.

"Y" & "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR -----	\$6.50
Rainbow 40 wire ribbon cable, per foot-----	\$1.00
Hitachi 63C09E CPU and Socket -----	\$13.00
512K Upgrades, with RAM chips -----	\$72.00
MPI Upgrades	
For all large MPIs (PAL chip) -----	\$10.00
For small #26-3124 MPI (satellite board) -----	\$10.00
Serial to Parallel Convertor with 64K buffer, cables, and external power supply -----	\$50.00
2400 baud Hayes compatible external modems -----	\$40.00

ADD \$2.00 S&H TO EACH ORDER

SERVICE, PARTS, & HARD TO FIND SOFTWARE WITH COMPLETE DOCUMENTATION AVAILABLE. INKS & REFILL KITS FOR CGP-220, CANON, & HP INK-JET PRINTERS, RIBBONS & Ver. 6 EPROM FOR CGP-220 PRINTER (BOLD MODE), CUSTOM COLOR PRINTING.

**TERRY LARAWAY, 41 N.W. DONCEE DRIVE
 BREMERTON, WA 98310 206-692-5374**

Want to move your CoCo keyboard? Get a

Puppo PC/XT Adapter

Allows use of any PC/XT compatible keyboard with your CoCo. Easy access to OS-9, DECB, and other options. Hold space bar on power-up to skip menu. Switchable and most auto-sensing keyboards supported. **ONLY 14 LEFT!!!**

Only \$72.50 post paid!

\$100.00 post paid with keyboard!

FARNA Systems

Box 321

Warner Robins, GA 31099-0321

NOTE: Keyboards will be 101 key. FARNA reserves the right to substitute 84 key models if necessary without notice.

ROMY-16

EPROM EMULATOR

- Emulates ROMS (2716-27010) or RAMs in 8- and 16- bit systems.
- Window/menu driven interface.
- Provides 8 hardware breakpoints for 8-bit systems.
- \$195 (2716-27256) or \$245 (2716-27010), 90 day warranty.
- 15 day money-back guarantee.
- Optional assembler, disassembler, and ROM debugger add \$100.

Universal Microprocessor Simulator/Debugger V2.1

- Simulates Z8, Z80, 64180, 8048, 8051, 8085, 6800, 6801, 6805, 6809, 68HC11, 6303, 6502 & 65C02.
- Assembler, Disassembler, & Windowed Symbolic Simulator. Supports on-board debug through RS232.
- \$100 each CPU (S&H \$8)

6809 Single Board Computer

- Support for 8K RAM, 8K ROM.
- Two 6821 PIAs connect 32 bits of I/O to outside world.
- No jumpers for 2732, 2764, & 6116.
- Two interrupt signals on CPU bus.
- Size is 2.75"x5". \$60 each board.
- For an integrated development system with assembler, disassembler, and on-board debugger please add \$70.

68HC11 Microcontroller Development System

- Eight channel 8-bit A/D converter. 32K ROM and 32K RAM.
- \$120 each SBC. Add \$70 for assembler, disassembler, BASIC interpreter and on-board debugger.

J&M Microtek, Inc.

83 Saman Road, W Orange, NJ 07052
 Tel: 201-325-1892 • Fax: 201-736-4567

Basic09 In Easy Steps

Chris Dekker

Advanced Programming: Keyboard and Screen Interfacing

bas0911.txt

This time around we will look at what we can use to enhance Basic09 in the areas of keyboard and screen interface. The first thing that comes to mind in this area is the following example. Suppose you want to write a program in which you want to use the arrow keys. Either to move the cursor on screen, to view records or whatever. This type of interface is easy to learn, easy to use and generally works very well.

There is, however, a big hurdle to be taken in implementing it. The arrow keys generate codes that, when send to the device driver, will affect the screen formatting. Either by moving the cursor or, in the case of the up arrow, clearing the screen altogether. This is, for obvious reasons, less than desirable. So what can we do about it?

Well,, for starters you could remap the keys, causing them to generate different codes. This is quite easy under DECB, but much harder to implement under OS-9. Another way would be to use the SHIFT or CTRL keys in conjunction with the arrow keys. This is asking for trouble because the first time you press only an arrow key - you end up with a mess again.

By the way, if you do need specific key codes for your program; they are listed on page C-1 (appendix C) of the OS-9 commands section in your manual. The only codes not listed are the ones for the F1 and F2 keys. They generate the following codes:

```
F1: B1 B3 B5 (normal, shift, ctrl)
F2: B2 B4 B6 (normal, shift, ctrl)
```

Now back to our original problem. As you might guess we can solve our problem easily with a system call. The reason is quite simple. What we consider "normal" operation for a computer (you press a key and something appears on the screen) is basically an emulation of the machine from which keyboards evolved: the typewriter. Unlike a typewriter, however, we can tell the computer to sever this link: if you hit a key it doesn't have to be echoed to the screen. Using this technique your program can pick up the keypress, decide whether to print the character or not and take it from there. The following code will implement this mode of operation for you:

```
DIM options(32):BYTE
regs.a=1 \regs.b=0 \regs.x=ADDR
(options)
RUN syscall($8D,regs)
options(5)=0
RUN syscall($8E,regs)
```

What happened? Basically we executed the command Tmode -echo. Doing it with this

code saves you memory and loading the Tmode utility. As you can see we used our "old buddies" I\$Getstt and I\$Setstt to pull it off.

By using functions 0 (regs.b=0; see pages 8-112 and 8-130) we can easily access and/or change a lot of parameters that affect the way our CoCo operates. These system calls read or write a 32 byte block from/to a path descriptor. A path descriptor is a 512 byte memory block that OS-9 has set aside for it's own use. This block contains addresses, pointers and parameters describing what OS-9 can and cannot do with data sent along that path. What we can do with these system calls is read or alter the parameter area of this path descriptor. You will find the exact definitions of this parameter area on pages 6-4 and 6-5 of your technical reference.

As you can see the fifth byte in this area controls the screen echo, hence the use of: options(5)=0. If your program has a BASE 0 statement in it this would become: options(4)=0. The only other noteworthy point in this code is: regs.a=1. As you should know register a always hold the path number for I\$Getstt and I\$Setstt. In this case we are dealing with the screen which is always addressed through stdout and OS-9 defines stdout as path number 1.

Now that you know how to turn screen echoing OFF, it would be handy if you could turn it back ON. This is done by writing a value larger than zero to this particular parameter:

```
options(5)=1
RUN syscall($8E,regs)
```

If you execute these two lines make sure that registers a, b and x still have the correct values. If they don't it is very easy to crash your program (incorrect a or b) or lock yourself out (problems with x) resulting in a trip to the reset button.

Two other parameters I change from time to time are PD.UPC (options(2)) and PD.PAU (options(8)) which disable lower case characters and end of page pauses, respectively.

To round things out: you can also use this technique to change the baud rate for printer and modem. And you can use it to change settings for RBF type devices: diskdrives. This may, however, have dire consequences if you don't know exactly what you are doing so I will leave that alone.

Another function of I\$Getstt I use from time to time is number 1. This function tests for the availability of data. I mostly use it to clear the keyboard buffer. You are probably all familiar with OS-9's type-ahead feature. This

is implemented by capturing keypresses in a buffer. When needed a program takes it's input from that buffer.

The problem there is that this data is accessed sequentially, which may cause your program to skip to the wrong subroutine, start looking for a non-existing file, etc. In cases were this input is critical to a program I use the following code to clear the buffer up front:

```
DIM key:BYTE
10 regs.a=0 \regs.b=1
RUN syscall($8D,regs)
IF LAND(regs.cc,1)=0 THEN
GET #0,key \GOTO 10
ENDIF
```

As you can see this code loops until it encounters an error. This means that the buffer is empty. As long as there are characters in the buffer, we read them one at a time. Since we can not be sure whether they were meant as input for our program or not, we just discard them without further processing.

I will skip the ss.EOF call here. This does the same job as Basic09's EOF function, which is easier to implement. If you insist on using it, you should check the cc register in the same way as in the code for clearing the input buffer.

Another function dealing with the keyboard is number \$27: ss.KySns. This implements a separate mode in which the computer only responds if you press an arrow key, control key or spacebar. You will find a complete list of keys and codes on page 8-120 of the technical reference. For switching this mode ON and OFF we use function \$27 of I\$Setstt.

The following piece of code shows you how to get into and out of the key-sense mode plus a way of processing the bit pattern it returns.

```
DIM i:INTEGER
regs.a=1 \regs.b=$27
regs.x=1 \(* turn key-sense on
RUN syscall($8E,regs)
90 regs.a=1 \(* ge: bit pattern
RUN syscall($8D,regs)
FOR i=0 to 7
IF LAND(regs.a,2^i)>0 THEN 100
NEXT i
GOTO 90 \(* no keys pressed
100 regs.x=0 \(* turn key-sense off
RUN syscall($8E,regs)
ON i GOTO ....
```

This code is fairly straight forward. The only thing that might need explanation is the LAND statement. What we do there is check

the contents of register a, which holds the bit pattern returned by the system call, on a bit for bit basis to see which key was pressed. If a key was pressed the program jumps to line 100 with i pointing to the key that was pressed. Your program can now jump to whatever subroutine is associated with that key.

The only drawback of this approach is that it is fairly slow. Remember what we discussed in part 10 of this series? It would be better to replace the term 2^i by a little table and use i as a pointer into that table. Your code would look like this:

```
DIM powers(8):BYTE
IF LAND(regs.a,powers(i))>0 THEN
100
```

Your table (powers) should hold the following values: 1, 2, 4, 8, 16, 32, 64 and 128. When initiating the loop for i; keep an eye on what the base value of your array is (BASE 1 or BASE 0) or the code won't work.

A very interesting call is function \$89, which sets and reads the mouse parameters. To do a thorough job explaining it probably takes a complete article. If enough people express an interest I might write that too, but for now I'll just skip this subject. This leaves us with one more group to deal with. These system calls deal with the screen and windowing system.

A number of these calls specifically deal with the VDG screens. This is the green 32x16 screen and since I don't think there are many people writing software for that anymore I will skip those calls too. Then you will find references to several calls only available through the WindInt module shipped with Multivue. These calls are also outside the scope of this article.

What's left is a number of calls dealing with screentype, size and colors. Most of them are very easy to implement. All they require is a path number and function code. Some of them are so simple they don't even return error codes. Examples are ss.ScSiz (\$26) for a window's size and ss.ScTyp (\$93) for screentype.

There are still a few things to say about those calls though. First of all ss.ScSiz. This call always returns the size of the current window. This may be, but not necessarily is (as I explained in part 4), the same as the size of your screen. The path number on this call can be set to 1 in which case it returns data about the window in which your program runs.

You can also use it to get information about a window that you can not see. In that case you must open a path to that window first. Like this:

```
DIM path:BYTE; window:STRING[5]
window="/w1"
```

```
OPEN #path,window:READ
regs.a=path \regs.b=$26
RUN syscall($8D,regs)
CLOSE #path
```

Regs.x and regs.y now hold the size of device/w1. If/w1 doesn't exist yet, the values from it's device descriptor are returned.

The ss.ScTyp call works in the same way but with a twist. There seems to be a bug in that code. It works fine for getting the type of a graphics window, but all text windows are labeled as type 2 (80 columns) even if they are type 1 (40 column) screens.

I usually work my way around that by executing ass.ScSiz call. If this returns a value in regs.x that is smaller than 80, the program assumes the worst and forces OS-9 to set up an 80 col. screen.

ss.FBRgs returns the register numbers that display the foreground, background and border colors. Please note that this is not the same as telling you which colors are displayed.

If you want to know that you must use ss.Palet (\$91). To use this call you must first set up a 16-byte array to hold the colors. After executing this call you can pick up the numbers of the colors that are held in the various registers in that array.

It would be nice if we could alter the colors available in a window by altering the values in that array and executing a I\$Setst call. Unfortunately this is not the case. You can use this technique to alter your CoCo's default palette (through function \$97) but this will not affect already existing windows.

If you want to change the colors that are available on an existing window the only thing you can do is use the Palette function as it is described in your manual's windows section on page 3-26. Of course we can make life a little easier for ourselves by using gfx2. The code would then look like this:

```
DIM register,color:INTEGER
FOR register=0 TO 15 \READ color
RUN gfx2("palette",register,color)
NEXT register
DATA .... (16 numbers in the range of
0-63)
```

There are a few more Setst calls like ss.Tone, ss.Montr and ss.GIP, but they do pretty much what they are supposed to do according to the manual. If you are interested in using those: read pages 8-146, 8-147 and 8-150.

In the last part of this article I want to show you how to set up windows dynamically (i.e. device names are not hardcoded into your program) and how to put more than one device window on one screen. If you want to dynamically allocate windows you must do the initial window access through the /w descriptor. This allows OS9 to pick out the first

available window for you.

```
DIM path,codes(20),device(32):BYTE;
i:INTEGER
DIM window:STRING[5]
OPEN #path,"/w":READ
regs.a=path \regs.b=14
\regs.x=ADDR(device)
RUN syscall($8D,regs) \CLOSE #path
i=1 \window="/"
WHILE (device(i)<128) DO
window=window+CHRS(device(i))
i=i+1 \ENDWHILE
window=window+CHRS(device(i)-128)
```

Now that we know the name of the window that is available to us (in window) we can initialize the device. This is the same function as OS-9's iniz utility.

```
regs.x=ADDR(window)+1 \regs.a=3 \
(* update mode
RUN syscall($80,regs) \
(* also check for errors
```

Now we can put the window on the screen without worrying about losing it:

```
FOR i=1 TO 12 \READ codes(i) \NEXT i
DATA 27,32,1,0,12,40,12,0,2,5,27,33
OPEN #path,window:WRITE
PUT #path,codes
regs.x=2 \RUN syscall($0A,regs)
CLOSE #path
RUN gfx2("select")
```

To add the other windows on the same screen: get the name of a window in the same way as we did for this one or simply add 1 to the number of this window. I.e. if "window" holds /w4 your next window would be called /w5; so load window with "/w5". You do not have to explicitly initialize these two windows for the program to work. So we can skip that code and open a path to "/w5" and establish it. Then you repeat this for "/w6". In order to avoid conflicts (device windows can not overlap) you must reload the "codes" array with the following values.

```
For /w5: 27,32,255,23,0,17,11,2,0,5,0,0
For /w6: 27,32,255,0,0,22,11,2,0,5,0,0
```

And finally to start a shell in the original window:

```
SHELL "shell i="+window+"&"
```

< 268'm >

Chris can be reached in care of
this magazine or directly at:
Chris Dekker
RR #4
Centreville, NB E05 1H0
CANADA

micro notes

Microware has released two new books for OS-9/68000. The first is "The OS-9 Primer" (\$45). This is a basic guide for the beginning OS-9 programmer. Contains step-by-step instructions, helpful hints, and many coding examples. It is written by Mark Heilpern, a trainer at Microware for the past two years.

The second book is a major revision of an old favorite, "OS-9 Insights" (revision 3.0, \$75; \$45 w/ title page of previous revision). Peter C. Dibble has added over 100 pages to the new edition.

For those unfamiliar with this tome, it includes details on the internal structure of OS-9, unique tips for customizing, ways to tap into additional resources from the kernel, and many illustrative program and script samples.

Both books can be ordered together for only \$100. Add \$6 per book S&H (\$13 to Canada). IA, CA, CN, NJ, and NY customers add sales tax. Order Dept., Microware, 1900NW 114th St., Des Moines, IA 50325-7077. Credit card orders call 800-475-9000.

OS-9 actually got a good mention in a mainstream computer magazine... the February 13, 1995 edition of PC Week. In the Application Development section, columnist Peter Coffee (Soft Talk) talks about

operating systems that are "... mature alternatives that are as different from Windows and OS/2 and Unix as a motorcycle is to a Taurus or an Accord."

Of course, OS-9 was mentioned in this article. I quote:

"If you're tired of being tied to the dated architecture of the X86, you can take a look at Microware Systems' OS-9 on Motorola 68xxx series processors and embedded controllers. You may be running it already. It runs more than 4,000 products, made by more than 700 companies, and will soon be running tens of millions of set-top interactive TV controllers thanks to a major deal with Bell Atlantic. OS-9 is already the standard platform for California's Department of Transportation."

Good work Mr. Coffee! Thanks for noticing a great operating system! The only thing missing was a mention of OS-9000 for X86 platforms.

RECENTLY DISCOVERED:

A cache of 180 CoCo 2s... new in the box! Call the TRSure Trove BBS at 213-664-5056 for more info (modem number!). This BBS supports the CoCo and other TRS-80 systems.

There is a CoCouser's group in Arlington, Texas! Write Mid Cities TRS UG, Box 170717, Arlington, TX 76003. They support the Model 1/3/4, CoCo, and Model 100/102/200 notebook computers. Dues are \$12 per year, and a newsletter is sent out regularly.

FOR SALE: 512K CoCo 3, CM-8 RGB Monitor, FD-501 double sided drives (two!), DMP-105 printer with stand, 300 baud modem, two joysticks, hi-re interface, and mouse - all in excellent condition with dust covers. Rainbow Magazine from 2/87 through 1/90 with some Rainbow on Disk. Many ROM Paks and approximately 50 disks full of programs. All for \$275! Please call 9-5 EST 1-800-283-5412, after 6pm EST 508-833-0765. Ask for Ed. or write:

Ed Eszlari
28 Cromwell Road
Sandwich, MA 02563

"micro notes" is for news on anything that may be of interest to readers, personal classified ads, and new product releases. If you see anything that may be of interest to other readers, have something for sale, or need an item, please write in and let us know! Vendors: be sure and let us know when you have a new product even if you don't already advertise!!

Did you miss the Chicago CoCoFest? Then get ready to attend

The 6th Annual Atlanta CoCoFest

September 30 & October 1, 1995
Northlake Holiday Inn, Atlanta

Show Hours:

Sat., Sept 30: 9:00AM-5:00PM

Sun., Oct 1: 9:00AM-3:00PM

Vendor Setup:

Fri., Sept 29: 6:00PM-9:00PM

Sat., Sept 30: 8:00AM-8:45AM

Admission:

\$10 both days

Reservations:

Northlake Holiday Inn

1-800-465-4329 or 404-938-1026

Sponsored by:

Atlanta Computer Society

PO Box 80694

Atlanta, GA 30366

BBS: 404-636-2991

SPECIAL FIND!

Joe Scinta of Ken-Ton Electronics has uncovered FOUR of Ken-Ton's DUAL COMM PORTS. These are RS-232 paks that sport two ports. They are completely compatible with OS-9 and DECB applications. They are housed in a short ROM-pak type case with gold plated cables extending to DB-25 connectors. These are the last four of these ports that will be available! Come complete with 90 day warranty and documentation. Use with standard OS-9 drivers. Get yours NOW while they are still available! Ports are \$115 plus \$3 S&H each. All docs and cables included. Send orders to:

FARNA Systems
Box 321
Warner Robins, GA 31099-0321

NEW PRODUCTS

from FARNA Systems!

New Book: "Mastering OS-9"

This is a completely revised edition of Paul Ward's popular classic, "Start OS-9". The new format is easier to read, tutorials have been revised to be easier to follow, and all the information articles have been completely updated. A vendor list appears in the appendix as well! If you are relatively new to OS-9 on the Color Computer, this book is a must have! Only \$30.00, and that includes the utility disk!

A version is in the works for OS-9/68000. Should be released in time for the Atlanta CoCoFest in October.

FARNA Systems

Box 321

Warner Robins, GA 31099

\$2.50 shipping and handling per order.
Canada S&H \$4.00; Overseas \$7.00

NOTICE: WE CAN NOW ACCEPT CREDIT CARDS!!! Visa and MasterCard (only!) can now be accepted for payment. There is a 6% service charge and a minimum charge of \$25. Cards are accepted through a third party and will be billed through "FS Printing".

Puppo PC/XT Keyboard Adapters Now Available!!

We now have a source for NEW Puppo PC/XT keyboard adapters. Use the 84-101 key keyboards with your CoCo! Has a menu burned into the on-board EPROM for easy access to OS-9 or DECB and other options. Requires a PC/XT compatible keyboard; original, switchable, or auto-sensing types. Cost is \$70 + \$2.50 S&H for interface alone, \$100.00 post paid with keyboard (101 keyboards are sent if available, but 84 key models may be substituted if 101's not available at this price). **NOW SHIPPING!**



FARNA Systems

Software, Books, and Hardware for all OS-9/IOSK Systems!

Box 321
Warner Robins, GA 31099
Phone 912-328-7859
Internet: dsrtfox@delphi.com

CoCo DECB Software:

CoCo Family Recorder - \$17.50

Genealogy program for CoCo 3. Requires 2 drives, 80 col. monitor.

NEW! OS-9 Version - \$32.50

DigiTech Pro - \$12.50

Sound recorder for CoCo3. Record any sound for easy play-back in your BASIC or M/L programs.

ADOS: Support for double sided drives, 40/80 tracks, faster formatting, much more!

Original (CoCo 1/2) - \$15.00

ADOS 3 (CoCo 3) - \$25.00

Extended ADOS 3 - \$30.00 (ADOS 3 req., RAM drives, support for 512K-2MB)

ADOS 3/Ext. Combo - \$50.00

Mind Games - \$5.00

Collection of 9 classic games. Run from included RAM disk w/ 512K.

Cross Road II - \$5.00

Simple Tic-Tac-Toe, but with amazing sound and graphics! Sound recorded with Digi-Tech.

Space Intruders - \$10.00

Looks just like Atari's classic "Space Invaders"! CoCo 1/2 and 3.

Donut Dilemma - \$10.00

Climb, jump, and ride elevators to top of Donut factory to shut it down! 10 levels. CC 1,2,3.

Rupert Rythym - \$10.00

Collect Rupert's stolen notes, then work our correct sequence. Great action adventure!

***** Only \$25 for all three!!! Save \$5!!!**

CoCo Hardware:

DigiScan Video Digitizer - \$150: Capture images from VCR, camcorder, or TV camera. No MPI required- uses joystick ports. CoCo Max3, Max 10, Color Max 3 compatible. Special order- allow 90 days for delivery. Send \$75 deposit, remainder due before delivery.

Puppo Keyboard Adapters - \$70: Use IBM PC/XT keyboards with your CoCo! Mounts in your CoCo case with no soldering. 101 key keyboards available for \$30 with order.

Ken-Ton SCSI Hard Drive System Components

No-Drive Kit: controller, OS-9 drivers, 2 pos.

"Y" cable, and drive cable (specify direct to drive or SCSI case connector). ----- \$175.00

Controller w/drive cable ----- \$135.00

OS-9 Drivers ----- \$25.00

RGB-DOS (for DECB access) ----- \$35.00

Seagate N series drive with ROM rev. 104 or greater needed for DECB compatibility.

**** \$50 for RGB-DOS and OS-9 drivers when purchased together with a controller ****

Add \$2.50 S&H per order. Can/Mex \$4.00; Overseas \$7.00

FARNA Systems Publishing Services

Type Setting and Printing: We can prepare professional typeset manuals, books, booklets, catalogs, and sales flyers for you - we can print or you reproduce as needed from a master set! Very reasonable prices - inquire!

Mailing Service: If you send catalogs or letter correspondence to 200 or more persons at once, we can do all work for you for about the same cost of your materials alone! How much is your time worth???

Contact Frank Swygert at above address/phone for quotes

for all your CoCo hardware needs, connect with

CoNect 449 South 90th Street
Milwaukee, WI 53214
E-mail: rickuland@delphi.com

The main problem with OS9 under a CoCo is the serial port. With a one character buffer, it's hard to do much before the serial port needs service. Windows and OS2 have the same problem. Or did, until National released the 16550 uart- 16 bytes of internal fifo buffering gives multitasking systems time to do some.

Announcing Fast232

Tandy with Sacia	Fast232
bps load ___ thrupt	load ___ thrupt
2400 28.3 sec 237 cps	25.3 sec 235 cps
9600 73.6 sec 938 cps	31.4 sec 950 cps
57600 not available	32.6 sec 5373cps

Local machines with faster modems can now be connected to properly (or improperly at 115K!). More down to earth, Fast232 is a ROMPak sized case, which will accept a daughterboard (once I get the case to close) to give two ports in a very 'concise' package. OS9 drivers by Randy Wilson. Free bonus software! The pd release of 'SuperComm' (Dave Phillipsen and Randy Wilson). All software includes 6809 and 6309 versions.

Fast232	\$79.95
Second port	\$45.00

NEW FOR 1995 FROM DISTO!

1. "Inside 2-Meg": A technical booklet that fully describes how the DISTO 2-Meg Upgrade kit works. Includes schematic, PAL listing, theory and chip by chip circuit explanations. \$20 + \$2.50 S/H.

2. "Blank Board Kit": Includes blank virgin boards (no components) of the SCII, SCI, 4IN1, MEBII, MPROM and Mini Controller. Collect all the components and make your own! \$29.90 + \$4.50 S/H.

3. Call for other DISTO products in stock
(limited quantities available)

DISTO

1710 DePatie
St. Laurent, QC H4L 4A8
CANADA

Phone 517-747-4851

Color Computer OS-9 Software

Invasors09 by Allen Huffman
High speed machine language arcade game for OS-9. Multi shots and increasing levels.

Req: CoCo 3, Joystick Optional\$14.95

Chicago '94: The Game by Allen Huffman
Experience the 3rd Annual "Last" Chicago CoCoFest with this graphics adventure. Over 30 digitized locations from the 'Fest to explore. Req: CoCo 3.....\$ 9.95

MultiBoot by Terry Todd & Allen Huffman
Have up to 16 OS9BOOT files on one disk. Select with scrolling menu. Booster/Nitros compatible.
Req: CoCo 3\$19.95

Towel! by Allen Huffman
Point 'n click disk utility. Use mouse or hot keys for all common commands. Configure all colors/commands. Includes EthaWin interface.
Req: CoCo 3, Mouse Optional.....\$24.95

1992 CoCoFest Simulator by Allen Huffman
Dozens of digitized images from the '92 Atlanta 'Fest. Explore the vendor area and hotel locations.
Req: CoCo 3, 500K+ Disk Space.....\$ 9.95

MAC to DMP Print Utility by Carl England
Print classic Macintosh images to Tandy printers.
Req: Level 1 or 2, DMP Printer.....\$19.95

CheckBook+09 by Joel Mathew Hegberg
Point 'n click account manager with graphing. Save pie, bar, or line graphs out in VEF format!
Req: CoCo 3, Mouse Optional.....\$24.95

MiniBanners09 by Allen Huffman
Single or multi-line banners on ANY printer. Dozens of fonts included. A classic!
Req: CoCo 3, ANY Printer.....\$19.95



Now Offering StrongWare Products!

Soviet Bloc - The best CoCo3 Tetris(tm)-like game ever. Supports stereo! RS-DOS
Req: CoCo 3, Orch-90/Joystick Optional.....\$19.95

GEMS - A falling "columns" puzzler. Supports stereo! RS-DOS
Req: CoCo3, Orch-90/Joystick Optional.....\$24.95

Copy Cat - Simon says match the flashing colors and sounds (Also available for MM/1, \$14.95)
RS-DOS Req: CoCo3.....\$9.95

HFE HPrint Font Editor - Powerful machine language hprint/MiniBanners font editor. (Includes fonts!)
RS-DOS Req: CoCo3, Joystick Optional.....\$19.95

....with more to come!

Sub-Etha Software
P.O. Box 152442
Lufkin, TX 75915

Add \$2.50 S&H.

TX residents add 8.25% tax.

Write us for more info!

Color Computer Disk Basic Software

RS-DOS OS-9 Terminal Emulator by Terry Todd
Use a CoCo3 as a "smart" terminal. Emulates screen/color codes, overlay windows, attributes and more via the bitbanger port at 2400 baud. Run many full-screen apps over the modem.

Req: CoCo 3, Serial Cable.....\$24.95

InfoPatch by Terry Todd
Patch classic Infocom(tm) text games to run on 80 columns, upper/lowercase, 6ms disk on CoCo 3.
Req: CoCo 3, Infocom Disk.....\$ 9.95

Super Boot by Carl England
The BEST! Type DOS to set tracks, sides, step rate, printer baud, colors, speed and more. Auto runs program or gives menu, too.
Req: 64K+ CoCo, DOS 1.1/2.1.....\$14.95

512K Copy Utilities by Carl England
Copy, rename, kill and format using full memory.
Req: 512K CoCo 3.....\$14.95

Super Backup by Carl England
Use from 64K to 512K for easy backups.
Req: 64K+ CoCo.....\$14.95

4-D Checkers by Nick Johnson
Classic twist. Even talks with speech sound pak!
Req: CoCo 3, Speech Pak Opt.....\$14.95

CheckBook+ by Joel Mathew Hegberg
The "CoCoMax" or checking programs! Fully graphics and mouse driven.
Req: CoCo 3, Hi-Res Int/Mouse.....\$24.95

MiniBanners by Allen Huffman
The original multi-line banner printer. A classic!
Req: CoCo 3, ANY Printer.....\$19.95

Are you a CoCo,
OS-9,
OS-9/68000, or
OS-9000 vendor?
Your products
should be
advertised here!
See inside front
cover for
basic rates!

CoCoTop version 1.0	\$24.95
CoCoTop version 1.1	\$19.95
CoCoTop 1.1 + Tools 3	\$34.95
OScopy/RScopy	\$10.00
TOOLS 3 version 1.1	\$29.95
Quickletter version 2.0	\$19.95
Accounting level 2	\$34.95
Investing level 2	\$24.95
Level II graphics 1.2	\$34.95

upgrades only \$5.00 (return original disk)

Shipping+handling: US/Canada \$3.00 all others \$5. Prices in US dollars. Send cheque or money order NO COD'S. Call or write for Canadian dollar prices. *Mention the name of this magazine in your order and you will receive a free bonus disk!*

C. Dekker ...
RR #4 Centreville, NB
E0J 1H0, CANADA
Phone 506-276-4841

*User-friendly Level II
Programs!*



EDTASM6309 Version 2.02 ----- \$35.00
This is a major patch to Tandy's Disk EDTASM to support Hitachi 6309 codes. Supports all CoCo models. CoCo 3 version uses 80 column screen, 2MHz. YOU MUST ALREADY OWN TANDY'S DISK EDTASM TO MAKE USE OF THIS PRODUCT. It WILL NOT work with a disk patched cartridge EDTASM.

CC3FAX ----- \$35.00
Extensive modification to WEFAX (Rainbow, 1985) for 512K CoCo 3. Uses hi-res graphics, holds full 15 min. weather fax image in memory. Large selection of printer drivers. Requires shortwave receiver and cassette cable (described in documentation)

HRSDOS ----- \$25.00
Move programs and data between DECB and OS-9 disks. Supports RGB-DOS for split DECB/OS-9 hard drives. No modifications to system modules (CC3Disk or HDisk) required.

DECB SmartWatch Drivers ----- \$20.00
Access your SmartWatch from DECB! New function added to access date/time from BASIC (DATES). Only \$15.00 with any other purchase!

RGBOOST ----- \$15.00
Make the most of your HD6309 under DECB! Uses new 6309 functions for a small gain in speed. Compatible with all programs tested to date! Only \$10.00 with any other purchase!

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Add \$4 shipping & handling per order

Mid Iowa and Country CoCo Club

(non-profit)

If you want support, we're here for you! MI&CCC publishes the UPGRADE disk magazine, now in its tenth year, five as a national publication. We've grown to be one of the largest CoCo outreaches in the world! We have subscribers in over 40 states and five provinces of Canada, as well as in Australia and England.

Your MI&CCC membership brings you:

1. A year's subscription to the UPGRADE disk magazine (requires CoCo 3 and one disk drive), 8-10 issues per year. This is a news magazine, not a software disk.
2. Access to our shareware/public domain/orphanware library - we keep only the best!
3. Optional Christian sub-chapter that gathers Christian oriented software for those interested.
4. ROM burning and other support.

Say you saw this ad in "68 micros" and receive a bonus disk along with your new membership!

Mid Iowa & Country CoCo Club

Terry Simons, Treas./Editor
1328 48th, Des Moines, IA 50311

Please include your phone number and system information

The OS-9 User's Group, Inc.

Working to support OS-9 Users

Membership includes the Users Group newsletter, MOTD, with regular columns from the President, News and Rumors, and "Straight from the Horse's Mouth", about the use of OS-9 in Industrial, Scientific and Educational institutions.

Annual Membership Dues:

United States and Canada	Other Countries
25.00 US	30.00 US

The OS-9 Users Group, Inc.
6158 W. 63d St. Suite 109
Chicago, IL 60638
USA

Northern Xposure 'Quality Products from North of the Border'

OS-9 Level II Color Computer 3 Software

NitrOS-9 v1.20 Call or write for upgrade info or new purchase procedure. Requires Hitachi 6309 CPU	\$29.95
Shanghai:OS-9 Introductory price	\$25.00
Send manual or RomPak to prove ownership	
TheXder:OS-9 Send manual or RomPak to prove ownership	\$29.95
Smash! Breakout-style arcade game	\$29.95
Rusty Launch DECB/ECB programs from OS-9!	\$20.00
Matt Thompsons SCSI System v2.2 'It flies!'	\$25.00
256/512 byte sectors, multipak support	

Disk Basic Software

Color Schematic Designer v3.0 New lower price	\$30.00
Oblique Triad Software	<i>Write for catalogue</i>

Color Computer 3 Hardware

Hitachi 6309 CPU (normally 'C' model, may be 'B')	\$15.00
SIMM Memory Upgrade Runs Cooler! 512k	\$44.95
0K	\$39.95
Sound Digitizing cable	\$15.00

OS-9/68000 Software

OSTerm 68K v2.2 External transfer protocol support	\$50.00
TTY/ANSI/VT100/K-Windows/Binary Emulation	
Upgrade from TasCOM (Send TasCOM manual please)	\$30.00

7 Greenboro Cres
Ottawa, ON K1T 1W6
CANADA
(613)736-0329

*All prices in U.S. funds.
Check or MO only.
Prices include S&H*

Internet mail: cmckay@northx.isis.org

Don't have a subscription to "microdisk"? Don't want to pay the high price for back issues? You can now get the complete Volume 1 of microdisk for \$30 (plus \$2.50 S&H)! That's an \$18 savings over back issues and a \$10 savings over the subscription price! Just write and tell us you want the entire volume 1.

All files will be on as few disks as possible, not separate disks for each issue. "microdisk" is not a stand-alone product, but a companion to this magazine. Subscriptions are \$40 per year. Single issues are \$6 each in US, \$45/\$6.50 in Canada. Overseas add \$10 per year, \$1 each for airmail.

ADVERTISER'S INDEX:

Atlanta CoCoFest	21
BlackHawk Enterprises	11
C. Dekker	24
CoNect	23
Delmar Company	BC
DISTO	23
FARNA Systems	8,18, 22
HawkSoft	18
J&M Microtek	18
Mid Iowa CoCo Club	24
Northern Xposure	25
OS-9 User's Group	25
Robert Gault	24
Small Grafx Etc.	18
Sub-Etha Software	23

*Don't have a subscription yet?
WHAT ARE YOU WAITING FOR?!*
Subscribe today!
(Details inside front cover)

For superior OS-9 performance, the

SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait-state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The **SYSTEM V** builds on the design concepts proven in the **SYSTEM IV** providing maximum flexibility and inexpensive expandability. Optionally available at 33 MHz.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit inexpensive upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index of a **SYSTEM V** running at 25 MHz with a standard VGA board is 0.15 seconds faster than a 68030 running at 30 MHz with an ACRTC video board (85.90 seconds vs 86.05 seconds).

For less demanding requirements, the

SYSTEM IV

The perfect, low cost, high-quality and high performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform or just plain fun machine. Powerful, flexible and expandable inexpensively. Uses a 68000 microprocessor running at 15 MHz.

G-WINDOWS - a PROVEN WINNER

FOR OS-9

G-WINDOWS is now
AVAILABLE for OS-9000 from
delmar co

Available for the **SYSTEM IV** and **SYSTEM V** computers, the PT68K4 board from Peripheral Technology and the CD68X20 board from Computer Design Services. Resolutions from 640 x 480 x 256 to 1024 x 768 x 256. Support for multiple VGA cards running different processes, different portions of the same process or both.

Support for generic VGA boards and SUPER VGA boards including ET4000 (Tseng Labs), OTI067 (Oak), CT452 and CT453 (Chips and Technology), GENOA, WD90C11 (Paradise) and S3 (S3 Inc.) for modes from 640 x 480 to 1280 x 1024 depending on board.

Distributor of MICROWARE SYSTEMS CORPORATION Software

This ad was prepared and printed using QuickEd under OS-9.

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709
302-378-2555 - FAX 302-378-2556