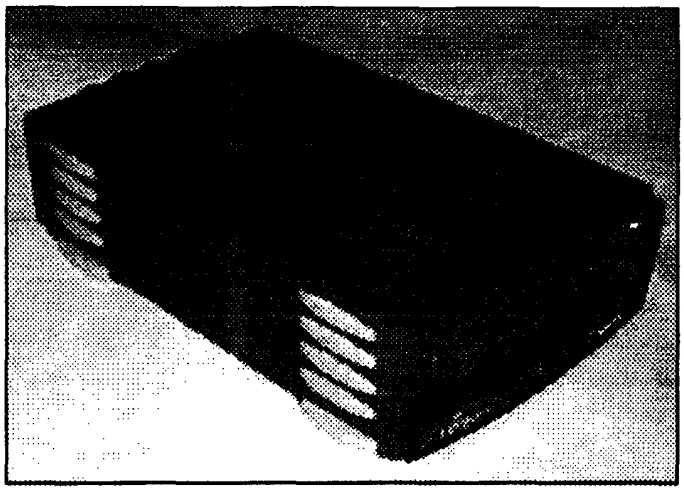
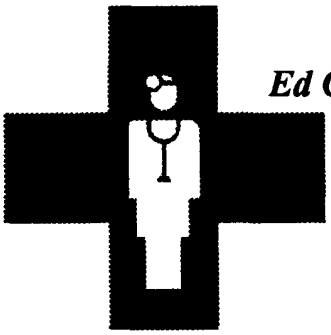
July/August 1996 Vol. 4, No. [illegible] $5.00 US/$[illegible] (North America)

the world of micros

Support for Motorola based computer systems and microcontrollers, [illegible]

Using Syquest's newest portable drive -- the EZ135 -- with the CoCo and OS-9/68K machines... and a comparison with Iomega's ZIP drive.





Ed Gresick, owner of Delmar Co. and stalwart OS-9/68K benefactor, is gravely ill.
See story on page 3.

CONTENTS

JOHN SUBSCRIBER 06/96
STREET ADDRESS
CITY, STATE
ZIP CODE

LAST ISSUE

Does your mailing label look like this?
If it does, the date after your name is the LAST ISSUE THAT YOU PAID FOR, and any other issue is basically FREE! You won't be getting any more unless you go ahead and RENEW NOW!!! Remember, ***WE NEED YOUR SUPPORT!!!***

POSTMASTER:
If undeliverable return to:
FARNA Systems PB
Box 321
WR, GA 31099

If your address is incorrect, send me a postcard!

the world of 68' micros

Publisher:
FARNA Systems PB
P.O. Box 321
Warner Robins, GA 31099-0321

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price. Canada add $0.50 per issue ($1.00 max) additional shipping, overseas add $2.00 per issue. Bulk/newsstand orders available.

microdisk:
Companion to magazine. Contains program listings, shareware, and freeware as mentioned in magazine text. Three issues per year.
US/Mexico: $20 per year ($8 single)
Canada: $25 per year ($10 single)
Overseas: $40 per year ($15 single)

Advertising Rates:
Contact publisher. We have scales to suit every type of business. Special rates for entrepreneurs and "cottage" businesses.

Contributions:
Any and all contributions welcome. Submission constitutes warranty on part of the author that the work is original unless otherwise specified. Publisher reserves the right to edit or reject material without explanation. Editing will be limited to corrections and fitting available space. Authors retain copyright. Submission gives publisher first publication rights and right to reprint in any form with credit given author.

General Information:
Current publication frequency is bi-monthly. Frequency and prices subject to change without notice. All opinions expressed herein are those of the individual authors, not necessarily of the publisher. No warranty as to the suitability or operation of any software or hardware modifications is given nor implied under any circumstances. Use of any information in this publication is entirely at the discretion and responsibility of the reader.

All trademarks/names property of their respective owners

The publisher is available via e-mail at: dsrtfox@delphi.com

A message from the editor...

I've just got to say it... this is one heck of an issue! There is a lot of stuff for everyone here... I don't think there has ever been a better one! PLEASE don't expect me to do this good with every issue, I just don't think I can (though I will certainly try to do even better)!

Well, that's enough self back patting. But I really do think I did a lot this time. It was tough squeezing everything in. As long as I can keep quality articles coming though, they will be printed and my job will be easier.

Please don't forget to renew your subscriptions and let me know if your address changes. And it wouldn't hurt to talk your friends into subscribing. I know some of you get your info from the Internet and/or Delphi, but there is a lot of info here that wouldn't be available through those sources. The magazine is more portable too... you can't connect all the time, and you can't share a terminal as easy as a magazine. All information sources are complementary... no one has everything.

On a personal note, we will sorely miss Ed Gresick. His wife, Betty, has indicated that the business will be shutting down, since it required a lot of input from Ed. I was especially sorry to hear about Ed's health situation. I have met and talked with Ed several times in Atlanta during CoCoFests there. He is a very dynamic individual. I, for one, am very sorry to see him leave us. Ed, you have been a great help to all OS-9 users in one way or another. thank you for all the support...

Speaking of CoCoFests, there won't be one in Atlanta this year. Instead, the Atlanta Computer Society will be hosting a picnic at Stone Mountain Park just east of Atlanta. This will be held October 5th. I will be there taking magazine subscriptions and renewals, and other vendors are invited to come show there wares also. The only thing that is missing is electricity, so we can't demonstrate any programs or hardware. Oh yes, the picnic is planned to start at 10a.m., and you'll need to let ACS know if you will be eating. Contact them at: ACS, P.O. Box 80694, Atlanta, GA 30366; BBS 404-636-2991. There is a $5 entrance fee to the park which will be reimbursed to ACS members. I'll have details as to the location of the pavilion, directions to the park, accomodations, etc., in the next issue. If you need it before then, contact ACS. All CoCo/OS-9/OSK enthusiasts are invited to attend! This year, come to Atlanta and have some fun as well as talk about your favorite system(s)! Stone Mountain Park is an excellent family park, with plenty of attractions for all. There is also a campground in the park. Hope to see you there...

Colin McKay at Northern Exposure is making some of the AT keyboard adapters. They are making only one short run, so if you want one, contact him at the address in the NX ad in the back, or contact Ken Scales via e-mial:
kscales@delphi.com

Well, that about sums everything up this time. I've really worked hard on this issue, and had to work fa to get it out in time. Hope I didn't miss anything! See you a[illegible] in a couple months!

Messages from our readers...

Dear Frank,

Enclosed is my renewal for "68' micros" and the disk. Any change you make to "68' micros" magazine I am sure will be for the best – it's always a pleasure to read it. I don't know how long CoCo's are still going to be still in use – it's such a nice machine -- although every once in a while it dos sort of act up! But a cleaning of the drives and new disks bring it back to good behavior.

Best wishes to you and your family! Till next issue...

Mrs. L. Boult
330 Metcalfe St., Apt 403
Ottawa, ON K2P 1S4 (Canada)

Mrs. Boult, thanks for the kind words! I do try to change things only for the best. As long as I have enough subscribers like you, the CoCo will always be a big part of this magazine, of that you can be sure!

I just received my first issue of "the world of 68 micors" and I must say that I am quite happy with it. Regarding your editorial, have you considered bringing in the Atari ST, Amiga, and even MacIntosh crowds. I'm not talking about general or advocacy either, I mean real techinal articles dealing with realwork applications. That's just my $0.02 though. Hope things continue to go well.

Jim Cox
jimc@amc.com

Well Jim, I have considered bringing in some of the other makes. Of those you listed, I have really only considered the Atari ST/TT series. That may be an idea.. what do you other readers think? There are two reasons I am thinking of maybe adding Atari support. The first is that it is an orphan just like the CoCo, but still very useful. The second reason is that there is a version of OS-9 that is affordable that runs on it. This was still available at least a couple years ago, though the company offered no support with the product (about $200 US). There is also a possibility that programs written for it could be ported the MGR system, and hardware could be developed for both systems also (the Atari has a built in SCSI port, I think).

I'd definately like to hear from readers on this. Would you mind sharing space with another 68K based computer?

Non-Blocking I/O in OS-9/68K

Russell Keller

What is the best way to do nonblocking IO in OS-9/68000?

I have a do loop that I want to execute continuously and accept keyboard charcter inputs should the user happen to hit a key. Functions like getc block until a user inputs a value. On other systems I have used a kbhit() function that lets the program know if there is anything in the buffer. Is there a similar OS9 function?

One way to handle keyboard input is to use _gs_rdy() to determine if there is any data available.

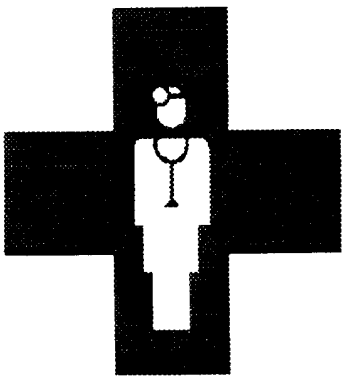
```
char input_buffer[1];
do {
   if (_gs_rdy(keyboard_path)>0) {
         read(keyboard_path,input_buffer,1);
         do_something_with_data();
   }
   else
         do_other_chores();
}
/* error checking ommited */
```

_gs_rdy() is not useful for getc() or other buffered calls and should be used with that in mind.

Another way is to make a signal based routine to send a signal when data is available. You may sleep() until data is available or use an intercept() routine to catch the signal and then fetch the data. This technique is useful in a complex environment such as multiple inputs or real time handleing of inputs. The signal system has a lot of flexibility and can simplify a complex set of inputs. One major benefit of the signal system is to allow the process to sleep() while no activity is occuring and allow other processes to use the cpu time. A programming technique of using a separate process to handle each input and sleep() between signals is very useful.

OS-9 can do it. It is like golf clubs. It takes practice. Have fun!

krussell@callruss.com



Many of you know Ed Gresick. He has supported OS-9/68K for a number of years through Delmar Co., which he owns. It is with sadness that we learned that he was hospitalized early in July with what appeared to be a stroke. This has turned out to be associated not with a stroke, but with cancer. The cancer is definately terminal, and he is not expected to live more than a few more months. I hope you all join with me in praying for him and his wife Betty. Cards may be sent to: **Delmar Co., P.O. Box 78, Middletown, DE 19709**. Please let his wife know that he will be sorely missed and is well respected by the CoCo/OS-9 community.

Removeable Hard Drives Made Easy

Allen Huffman

Use a Syquest EZ135 for OSK or CoCo... but not Iomega's ZIP...

One of the largest problems facing orphan computer hobbyists is reliability of old hardware. Many CoCo users are still relying on 1980s vintage XT based MFM hard drives, as well as 10+ year old expansion devices. Unfortunately, there isn't much one can do to "update" equipment that hasn't even been manufactured in ages. A partial solution does exist, however.

In 1995, two new removeable hard drive units were introduced to the market. Iomega brought out their ZIP drive, offering 100 megs of storage on compact floppy style disks, and then, later, SyQuest introduced their 135 meg removable EZ135 unit. Both were targeted at the PC and Macintosh market, with external SCSI models as well as parallel port units. Both are tiny units with their own power supplies and quiet operation (no noisy fan). To a Color Computer or OS-9/68000 machine owner the SCSI models hold particular interest.

Hard Drive Interfaces

There are three primary models in use: MFM/RLL (an older format, no longer produced, originally used for early PC/XT/AT machines), SCSI (old, but still in use and standard on most high-end or non-PC machines), and IDE (the current PC standard). Both SCSI and IDE also have more advanced versions such as SCSI-2, Wide SCSI, and Enhanced IDE. For the CoCo market, interfaces exist to support MFM/RLL and SCSI. For OS-9 machines, SCSI is the most common, though the AT306 system does come with a built in IDE interface. This article will focus specifically on SCSI hard drives.

An MM/1 owner will have no problems hooking up virtually any type of SCSI hard drive, but the Color Computer has a few restrictions. None of the CoCo SCSI interfaces seem to support hardware parity. Most SCSI drives have a jumper to enable or disable the use of this parity bit, but any drive that requires it will not be of use on the CoCo. The Iomega ZIP drive is one such example. The forced use of parity would require a hardware modification on a Disto, LR-Tech, or Ken-Ton style SCSI interface (a simple, one-chip modification if anyone will bother to design it). The MM/1 has no problems using a ZIP.

Another issue is sector size. Virtually all modern SCSI drives use 512-byte sectors. Again, OS-9/68K machines should have no problems with this since their drivers support different sector sizes. CoCo software typically expects 256-byte sectors. What this means is that under the CoCo, half the hard drive will not be useable. In past years, OS-9 Level 2 drivers were released that will support 512-byte sector drives. Matt Thompson's SCSISYS is the solution, allowing a CoCo OS-9 user full use of their 512-byte drive. Other drivers, and items such as RGB-DOS or Hyper I/O, are coded only for 256-byte sectors and won't make full use of provided storage - but they will work.

Lastly is SCSI ID support. The standard allows for IDs 0 through 7. Each SCSI device must have it's own ID. For example, two hard drives might be hooked up as SCSI ID #0 and SCSI ID #1. Not all driver software supports the full set of IDs. Once again, most OS-9/68K software will have no problems with this but the CoCo may. SCSISYS, for instance, only supports two bits for ID selection, giving you access to ID 0 through 3. The Iomega ZIP drive supports ONLY SCSI ID #5 and #6, making it useless even if we did have an interface that supported hardware parity. Older drivers, such as the ones designed for the original Ken-Ton, will support the full ID range but do not make use of 512-byte sectors. The delima is obvious. Even with the proper hardware, there would still be a software problem.

One final ZIP tidbit is that all internal media control is done through SCSI commands. A ZIP disk can be password or write protected through software. Unfortunately, the ZIP tools to do this haven't been made available for OS-9 yet which means if you were given a write protected ZIP disk you would not be able to use it without first low-level re-formatting it. Same goes for a password protected disk. The point is this: ZIP isn't for the CoCo. Proper use would require new hardware and software.

The Syquest EZ135

With all of this in mind, the focus is turned to the SyQuest unit. The EZ135 fixes everything that seems wrong with the ZIP. The EZ doesn't require hardware parity and, while it does use 512-byte sectors, it allows you to select any SCSI ID for the unit, meaning Matt Thompson's drivers will work great. Also, unlike the ZIP drive with DB25 Mac style SCSI ports, the EZ uses two full sized 50-pin SCSI connectors on the back and even comes with a terminator plug which must be put on the last unit in the chain of devices. There are also several major advantages to the EZ over ZIP. First is speed. The ZIP emulates a high-speed floppy. This makes the cartridges very rugged, but also very slow compared to a hard drive.

In speed tests, the ZIP transfered 90K a second on a non-accelerated MM/1 (68070 at 16mhz). The EZ135 transfered about 1.5 megabytes per second on the same machine. CoCo speed tests could not be made against the ZIP, but when comparing the EZ135 to a Quantum ProDrive 210S (250 meg SCSI hard drive), the public domain SCSISYS 1.0 drivers read 1 meg in 31 seconds on the EZ versus 26 seconds from the Quantum. This seems to indicate the EZ being slower than the Quantum, but running the same test again using the commercial SCSISYS 2.0 drivers resulted in 1 meg in 12 seconds from the Quantum or the EZ. That's quite a difference! (editor: SCSISYS 2.0 uses a cache to buffer the hard drive I/O— much like SmartDrive for you Windows users— which makes it extremely fast)

As you can see, the EZ is more like a true removeable drive, giving hard drive speeds with the downside of the media being a bit more fragile. For OS-9/68K users, if sending media through the mail was important and speed was not, the ZIP would seem to be better suited. Also, the ZIP is a bigger "standard" in the PC and Mac worlds making media easier to find. For everything else, the EZ seems to win hands down. (editor: If you do any desktop publishing, it is worth noting that the older, more delicate Syquest drives are preferred for sending large files.)

Storage Limitations?

One really nice thing is the unlimited storage potential for removeable units such as these. 135 megs on a CoCo is enough room for 256 RGB-DOS partitions as well as 52 megs for OS-9. If that gets full, but additional EZ disks at about $20 each.

Also, an EZ drive moves gracefully between the CoCo, MM/1, PC, Mac, Amiga, and just about anything else that supports SCSI hard drives. A multiple machine owner no longer has to buy a dedicated drive for each machine they own.

As you can see from the specs in the sidebar, the EZ135 really seems to be the better unit overall. A recent reduction in retail price to $120 makes it the cheaper of the two also. If storage size is important, another option might be the new EZ Flyer 230 meg removeable (which also reads and writes EZ135 disks) or the upcoming SyJet 1.6 gig removeable. Iomega also offers the 1 gig removeable JAZ drive for about $500 which has a different design and is reported to be much faster than ZIP and can use the full set of SCSI ids. The options are there, but it's difficult to ignore a $120 removeable media solution.

	SyQuest EZ135	Iomega ZIP
Average Seek Time	13.5	29ms
Max Transfer Rate	4mb/sec	1.4mb/sec
Typical Transfer	1.4 - 2.4mb/sec	1mb/sec
MTBF	200,000 hours	100,000 hours
Storage	135mb/disk	100mb/disk
Unit Warranty	2 years	1 year
Disk Warranty	5 years	Limited lifetime
Power Switch?	Yes	No - Unit is always on
Connectors	2 50-pin SCSI	2 DB25 SCSI
SCSI ID Range	0 through 7	5 or 6 only
Street Price	$120	$200

Summary:

If a drive for a CoCo is your only insterest, the EZ135 is the winner. If an MM/1 or other OS-9/68K system is your only interest, and widely used, rugged, mail-proof media is important, check into the ZIP. Remember, either of these can be used on any other machine with a SCSI port, so your drive can migrate with you should you change or can be used on all machines with different cartridges.

Using the EZ135 Drive on a CoCo

During experimentation with the EZ135 removeable hard drive on my CoCo, I found some tips that may be of interest to others.

How Big the EZ135 REALLY Is

Before getting started, let me mention that while the drive is promoted as a 135 megabyte unit, it only provides 262,144 sectors. Each sector is 512 bytes, giving a total of 134,217,728 bytes of total storage. Depending on your definition of "megabyte", your milage may vary. Assuming 1000 bytes per K, and 1000K per meg, this is just under 135 megs. This is apparently SyQuest's definition of a megabyte.

If you prefer the more "accurate" computer definition, with 1K being 1024 bytes, and 1024K per meg, then this unit gives exactly 128 megabytes of storage. For the sake of this discussion, we will use this definition since, mathematically, 135 megs will NOT work for the calculations I will be using.

About RGB-DOS

RGB-DOS is a patch that allows Disk BASIC to use a hard drive. It works by intercepting floppy disk i/o and converting that into hard drive calls. Typically, you access a floppy by reading or writing a particular sector of a particular track of a particular drive. RGB-DOS allows for drive numbers 0 through 255, and splits them up into "virtual drives" on the hard disk. One can switch between the "real" floppy drives 0-3 or the virtual drives through a special command. Virtual drives 4-255 mat be accessed along with the four physical floppys.

If your main interest is RGB-DOS, one EZ135 disk will give you room for 256 RS-DOS drives (and still have 50 megs left over for OS-9 use). The EZ135 has $40000 (262,144) sectors. Each RS-DOS disk takes up 630 sectors as follows: A drive is 35 tracks in size. A track is 18 sectors long. 35 tracks * 18 sectors per track = 630 sectors per disk (630 sectors * 256 bytes per sector = 161280 bytes.)

The EZ135 uses 512-byte sectors, but RGB-DOS doesn't know how to deal with them. RGB-DOS will simply use the first 256 bytes of each physical sector on the drive, wasting the remaining 256 bytes. RS-DOS "sees" it as: 630 sectors * 256 bytes per sector = 161280 bytes. However, the drive is actually storing it as: 630 sectors * 512 bytes per sector = 322560 bytes.

If you multiply this out for 256 disks, you will see that you are wasting 41,287,680 bytes of storage (about 40 megs). To remedy this would require a rewrite to RGB-DOS to use 512-byte sector drives (using 315 512-byte sectors per RS-DOS drive). Further rewrites might allow partitioning a hard drive into multiple banks, meaning a 135 meg drive might be able to allow three virtual banks of 256 RS-DOS disks each with 15 megabytes left over (possibly partially used by a fourth bank of 64 drives) for a total of 832 RS-DOS disks on one drive! It is, of course, very unlikely that any such rewrites will ever be made. Plus, with EZ135 disks being $20 each, why put all your eggs (software) in one basket (drive) anyway? Instead, I'll preset everything you need to get the most out of existing software and hardware, including some useful programs to get more out of RGB-DOS.

OS-9 and RS-DOS - A Skitzo Disk

If you wish to use the EZ135 disks for both OS-9 and RS-DOS you need to be aware of how both operating environments work. OS-9 typically expects the disk structure to start at the first sector of the drive. While changing this could be done, it would require modifications to the low level boot routines as well as higher level drivers. The drivers would have to be coded to add an "offset" to every disks access. Typically OS-9 would look for sectors 0 through MAX, where MAX is the total number of sectors on the device. Putting OS-9 further into the disk, say, starting at sector 1000, would require every disk i/o to have 1000 added to it. Reading sector 0 would end up reading sector 1000, and so on. This would make the driver slightly slower (Matt Thompson's SCSISYS 2.0 commercial drivers actually do support this, but mainly for partitioning large hard drives into smaller virtual sections.)

Instead of this, it's easier to let OS-9 have the first portion of the hard drive like it wants, and force RS-DOS to step over it. RGB-DOS allows you to specify an "OS-9 Offset" which will be added to every disk access made. By default, RGB-DOS expects RS-DOS drive 0 to exist as sectors 0 through 629 (630 sectors). Drive 1 would be at sectors 630 through 1259, and so on. By adding an offset, RGB-DOS can be made to access these virtual drives further into the drive in a similar manner as discused above with OS-9 drivers.

Before starting, you must decide how much room you want taken up for RS-DOS disks, with whatever is left over for OS-9. Decide how many drives you want and multiply that number by 630. This will tell you how far back from the end of the drive the RS-DOS section will begin. To simplify, think of it like this:

Suppose you had a drive with 100 sectors (numbered 0 through 99) and that each RS-DOS disk took only one sector. If you wanted it setup for 40 RS-DOS disks, they would start at sector 60 and go through 99, leaving sectors

0 through 59 for OS-9. The same thing applies here, except there are more total sectors and the RS-DOS partitions are 630 sectors instead of 1.

If you want the largest amount of space for RS-DOS (256 drives), you would take 256 drives times 630 sectors per drive which would give you 161,280. This is how many sectors they would take up. Moving back from the "end" of the hard disk by subtracting this value from the total number of sectors gives you 100,864 (262,144 total sectors minus 161,280 sectors). Since sectors start with 0, you'd need to subtract 1 from that value to get the actual starting sector: 100,863. Converting this into HEX is $189FF. This becomes the "OS-9 Offset" for RGB-DOS to use. Three memory locations inside RGB-DOS will contain $1, $89, and $FF. Then, everytime RGB-DOS goes to access a sector, it starts at this location rather than the start of the drive.

That takes care of the RGB-DOS side of things, but there is still the issue of letting OS-9 know how much drive space it actually has available. The drive must be formatted correctly so that it doesn't overwrite the RS-DOS area. To do this, you must "lie" to OS-9 and tell it the drive has fewer sectors than what it really has. In this example, we know that sectors 0 through 100,863 can be used for OS-9 so we have to tell OS-9 to logically format the start portion of the hard drive with just that many sectors. This is not as easy at is sounds, since the FORMAT utility bases what it does on entries in the device descriptor.

Format calculates the total number of sectors on the drive by multiplying the number of sides times the number of sectors per cylinder times the total number of cylinders on the device. Since I was using 100,863 512-byte sectors, I would normally have wanted values which produced that. HOWEVER, since I am using Matt Thompson's drivers which make full use of 512-byte sector drives I must do something different.

SCSISYS does a neat trick to keep OS-9 happy. It makes two "logical sectors" from each physical sector on the device. Thus, physical 512-byte sector 0 becomes logical sectors 0 and 1 to OS-9. Physical sector 1 is logical sectors 2 and 3, and so on. This must be realized since if you tell format how many physical sectors are there, it will only format half the space. (ie, 100,863 physical sectors remain, which is actually 201,726 logical sectors for OS-9.) So, I have my descriptor setup as follows:

sectors = 42
cylinders = 1601
42 * 1601 * 3 = 201,726 logical sectors
sides = 3

I changed my /H0 descriptor using the DMODE utility. The version I use accepts both hex or decimal entries, but I'll list both just in case you have an older version that does not:

Decimal values:

```
dmode /h0 sct=42 tos=42
cyl=1602 sid=3
```

Hex values: dmode /h0 sct=$2a tos=$2a cyl=$641 sid=$3

My /H0 descriptor for SCSISYS now looks like this:

```
adr=$FF74 drv=$00 stp=$00 typ=$80
dns=$80 cyl=$0641sid=$03 vfy=$00
sct=$002A tos=$002A ilv=$01 sas=$08
```

After this, you can format the OS-9 portion of the drive by doing a LOGICAL format:

```
OS9: format /h0
COLOR COMPUTER FORMATTER
Formatting drive /h1
WARNING: You are formatting
a HARD Disk..
y (yes) or n (no)
Ready?  [type `Y' here]
WARNING: You are formatting
a HARD Disk..
Are you sure? [type `Y' here]
Both PHYSICAL and LOGICAL
format? [type `N' here]
Disk name:
[enter disk name and proceed]
```

Then the logical format will complete creating LSN0, the file allocation bitmap, and the root directory. From this point on you should be in good shape with approximately 50 megabytes of storage for OS-9.

NOTE: If you are not using SCSISYS, and your OS-9 drivers do not support 512-byte sectors, you can still do this by using the actual amount of "left over" sectors (100,863). This gives you about 25 megs for OS-9. Changing the "sectors" value to 21 should suffice.

If you are wanting to experiment with other combinations of OS-9 and RS-DOS partitioning, I have created a basic program to help. In order to use it, you must know how many sectors your hard drive has, as well as if it is a 512-byte or 256-byte drive. To find the number of sectors, you can run the program HD-UTILS.BIN included on the RGB-DOS disk. LOADM and EXEC it, then select <F>ind Parameters from the main menu. It will show you a screen of information, including the actual number of physical sectors the drive supports. As an alternative, you can run SCSIFMT under OS-9 and have it query the drive size for you:

```
OS9: scsifmt
>>> SCSI SYSTEM FORMATTER 1.0 <<<
Copyright (C) 1993 by Matthew Thompson
      P) Physical Format/Verify
      L) Logical Format
      Q) Quit

      [type `L' here]

CAUTION: THE ROOT DIRECTORY
ON THE DRIVE WILL BE LOST!!!
YOU MUST MAKE A BACKUP IF
YOU WANT TO KEEP THE FILES!!!

Which device: /
[type your descriptor name here, such
as `h0']

      2) Drive Number:    0
      4) 512 Flag:        1
      5) SCSI ID:         0
      6) SCSI LUN:        0
      7) Err Type Flag:   0
      8) Cluster Size:    1
      C) Cylinders:       1601
      D) Heads:           3
      E) Sectors/Track:   42
      G) Segment Size:    8
      I) Device Name:     h0

Continue (Y/N)? [type `y' here]

Query drive size (Y/N)?
[type `y' here]

1) Disto, 2) Ken-Ton/LR-Tech?
[select your interface here]

Port address - 1) $FF74 (regular), 2)
$FF70 (alt) [and port address]
```

Watch which drive light flashes to make sure you've got the right one. Hit any key.

```
Total logical sectors: $07FFFE

Ready to begin.  Everything OK (Y/N)?
[type `n' here]
[then `q' to quit from the main screen]
```

Notice that on the EZ135 scsifmt actually returned the number of 256-byte sectors the unit will end up providing. On a 512-byte sector drive, you would really want to use half of this value.

Once this value is known, you can use the program OS9PART.BAS (listed below) to calculate the correct RGB-DOS offset and also modify the binary RGB-DOS image to support it. This is a replacement for the program OFFSET-9.BAS included with RGB-DOS.

```
0 REM *
1 REM *   RGB-DOS/OS-9 PartUtil V1.0
2 REM *       by Allen C. Huffman
3 REM * (formerly of Sub-Etha Software)
4 REM *
5 REM * This program will calculate the
6 REM * RGB-DOS offset needed to split
7 REM * the hard drive between OS-9 and
8 REM * RS-DOS. (Replaces OFFSET-9.BAS)
9 REM *
10  PCLEAR1:PALETTE0,0:PALETTE8,63
WIDTH40:CLS1
15 PRINT"RGB-DOS only uses 256-byte sec-
tors.  If your hard drive only supports
512-byte secotrs, RGB-DOS will only use
```

CoCo Disk Tester

Mark bad sectors on floppies to reduce read/write errors.

Robert E. Bruhl

In January 1985, the RAINBOW magazine printed a listing and text for a program named DISKTEST.UTL by Charles C. Zimmer. This program looks for bad granules by doing a thorough test of the disk by writing and reading all tracks with different data patterns. The test is created to effect the worst-case situations for data patterns and head positioning, so that this is not only a test of the diskette material itself but also a test of the head positioning accuracy of the disk drive. I modified the program to test 40 tracks and renamed it "DSKTST40.UTL".

If bad granules are found, the program creates a file called DEFLOG.###. The ### is the diskette number (000-999). All of the bad granules are assigned to this file. In this way these granules are effectively taken out of use and the disk can still be used for other things.

If you have a printer, a one line report can be created showing which granules are bad. The program was originally written for an OKIDATA 92 printer. I redid the printer routine for use with a DMP130 series printer. I use this program to check out all of my disks before using them. If the directory track is bad, I throw the disk away. If other granules are bad, I can always use the disk as a work diskette.

HOW THE PROGRAM WORKS

The program starts out by asking you to turn up the volume and then to press any key to continue. It then asks you to enter the drive number (0-3) and goes to the next message. The next message has you enter a "S" for a short test or anything else for a full test. The next message wants a "P" for a printout or anything else to continue. If a "P" is entered, the program then asks for a date (YYMMDD). At this point you are given the chance to "INSERT OF CHANGE DISKETTE". Put in a diskette and press "C" to commence the test or press "Q" to quit the program. If "C" is entered, the next message asks for the DISKETTE NUMBER (###). If no number is entered, 000 will be used. The program then starts the testing process by showing "OPERATION: FILE CHECK" and checking the directory track. If the directory track is bad, the message "DIR. TRACK ERROR-CAN'T CONTIN." "(ANY) TO RESTART OR QUIT". If the directory track is good, the program informs you whether or not the disk has files and if a defect log is present or not. If this disk has files, now is your chance to change disks in case this is a good disk.

At this point you are at the "FIRST DEFECT CHECK POINT" and the first 70 granules are shown on a defective granule map. You may either (E)nter changes or (C)ontinue. If you want to manually enter or delete defective granule numbers, you can do it at this time or you can continue. Entering a number less than 0 or greater than 77 will get you an "INVALID ENTRY" message. Pressing any key will get you back to the screen that asks if you want to continue or to enter changes. If you continue, the screen will be cleared and a new defective granule map will be drawn showing the last 8 granules. After all of the testing is complete, the program will return to the start of this paragraph, but then it will be called the "FINAL DEFECT CHECK POINT".

The program will now display 4 lines showing which cycle (1-6), track (0-39), granule (0-67,DIR) or operation (WRITE-READ) is taking place. If a bad granule is incountered, a fifth line (BAD GRANULE) will be displayed. After the "FINAL DEFECT CHECK POINT" maps have been displayed, the program will go back to the point where you can either insert a new disk or end the program.

FIGURE 1: READ, WRITE & POSITIONING SEQUENCES

CYCLE#	OPERATION	PATTERN & SEEK DIRECTION EVEN TRACKS	ODD TRACKS
INTERLEAVED ROTATING WORST-CASE PATTERN TEST			
1 (*)	WRITE	DB6	IN
	WRITE	6DB	OUT
	Read track 0 only; to set up for next		
	READ		IN
	READ	OUT	
2 (*)	WRITE	B6D	IN
	WRITE	6DB	OUT
	READ		IN
	READ	OUT	
3 (*)	WRITE	B6D	IN
	WRITE	DB6	OUT
	Read track 0 only; to set up for next		
	READ		IN
	READ	OUT	
INTERVEAVED 1,0 TEST			
4 (1)	WRITE	FF	IN
	WRITE	00	OUT
	Read track 0 only; to set up for next		
	READ		IN
	READ	OUT	
5 (2)	WRITE	FF	IN
	WRITE	00	OUT
	READ	IN	
	READ		OUT
6 (3)	WRITE	Directory track only, FF (a housekeeping cleanup pass)	

Patterns DB6, 6DB, B6D, FF & 00 are described in the article. (#) are cycle numbers shown in the short test, steps (*) are not done in the short test.

SOME TECHNICAL INFORMATION

The following information isn't needed to run the program, but it might be nice to know a little about the disk drive. When the drive records data on the disk, it magnetizes the iron oxide material under the head. As the disk rotates, the direction of the magnetization is switched back and forth according to an encoding algorithm that is known to the system. The effect is similar to small bar magnets laid end to end along the track. They vary in length and each successive one is turned around.

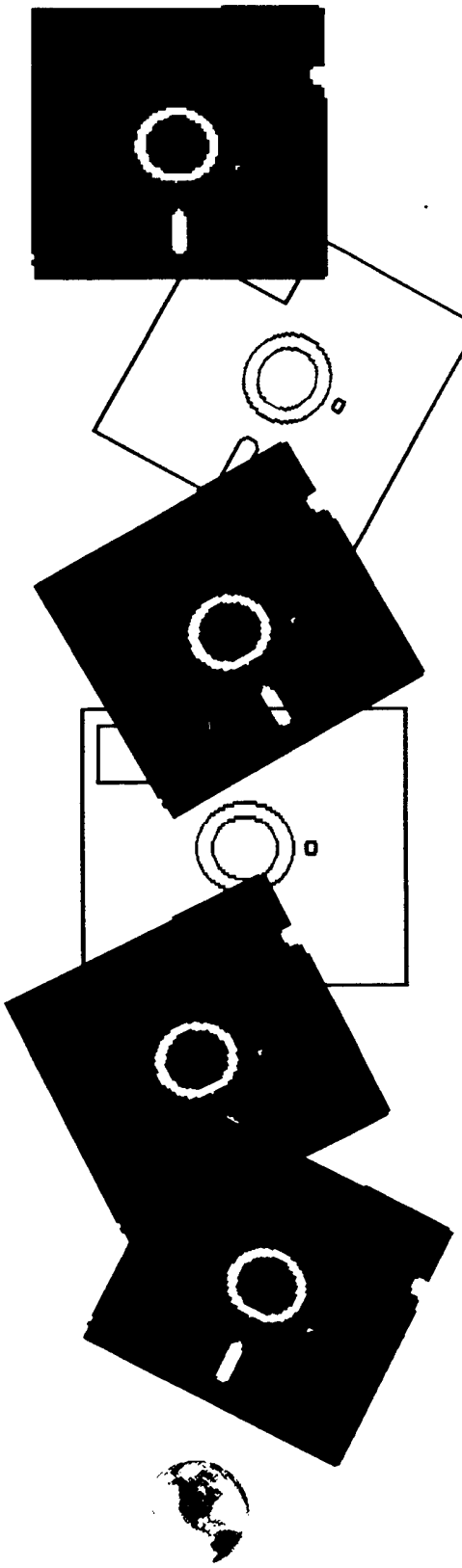
Later on, when the tracks are read, the bar magnets are passed under the head and every time a junction of two magnets passes under the head the magnetic flux through the head reverses and a small voltage is induced into the head. This signal is amplified and is sent back to the computer as the data which was recorded on the disk. The important thing is the time sequence of these flux reversals. They determine whether the data is a one or a zero.

If we want to test the disk, we want to put these flux reversals in as many places as we can in each track and then read the disk to see if they were properly recorded. Some flux reversals are harder to recover than others, so some good patterns have to be used. Mr. Zimmer found that the Hex patterns DB6, 6DB and B6D produced the worst-case flux reversals. Since each of these patterns is a 12-bit value, it is sent as DB 6D B6, etc. He also used the Hex patterns 00 and FF in testing the disks. Mr. Zimmer also set the program up to do a short test or a long test. The long test has 5 cycles that it runs through, while the short test uses only the last 2 cycles. Each cycle takes about 2 minutes so a complete test is only 4 or 10 minutes in lenght. The long test is recommended as it is the most complete test. Figure 1 shows what is done in each cycle.

If anyone would like a copy of the article from the RAINBOW magazine, just write to me at the address shown in the DSKTST40.UTL program and enclose a stamp. I will be glad to send them a copy.

```
10 'DISKTEST.UTL 1.1 (C) 1983 BY
CHARLES C. ZIMMER  101 AUSTIN RD
SUDBURY, MA 01776
11 'MODIFIED 1988 FOR 40 TRACK
OPERATION BY ROBERT E. BRUHL
12 '841 N. MAPLETON AVE.
13 'OAK PARK, IL. 60302
14 'NEW NAME DSKTST40.UTL
15 CLEAR200
20 BC=3:CLSBC
25 GOSUB180:GOTO280
30 'i/o sub----------
35
POKEPP,O1:POKEPP+1,D1:POKEPP+2,T1:
POKEPP+3,S1:POKEPP+4,4:POKEPP+5,0:
X=USR0(0):EC=PEEK(PP+6):RETURN
40 'gran to disp sub----------
45 IFX>33THENG=X+2ELSEG=X
50 IFG(X)=&HFF THENM$=" "ELSEM$=
RIGHT$(STR$(X),2)
55 PL=69+32*INT((G-24*INT(G/24))/2)
+10*INT(G/24)+(G-2*INT(G/2))*3
60 IFG(X)<>&HE9 HENPRINT@PL,
USING"%%";M$;
:RETURNELSEFORX2=0TO1:
POKE&H0400+PL+X2, ASC(MID$(M$,X2+
1,1)): NEXT RETURN
65 'inkey sub----------
70
K$=INKEY$:IFK$=""THEN70ELSERETURN
75 'pause sub----------
80
PRINT@I1,M1$;:GOSUB70:GOSUB170:
RETURN
85 'defect list form sub----------
90 CLSBC:PRINT@3,"MAP OF DEFEC-
TIVE
GRANULES:";:X3=0:FORX1=34TO54STEP10:
PRINT@X1,"TK/GR:GR";:FORX2=1TO12:
PRINT@X
1+32*X2,USING"##";X3;:PRINT"/:
";X3=X3+1:NEXTX2,X1:PRINT@239,"(DIR)";:RETURN
91 CLSBC:PRINT@3,"MAP OF DEFEC-
TIVE
GRANULES:";:X1=34:PRINT@X1,"TK/
GR:GR";
92 FORX2=1TO4:PRINT@X1+32*X2,
USING "##"; X3;: PRINT"/ :
";:X3=X3+1:NEXT: RETURN
95 'list defects sub----------
100
GOSUB90:FORX=0TO69:IFG(X)=&HB9
ORG(X)=&H99 ORG(X)=&HE9
THENGOSUB45
101 NEXT:RETURN
102
GOSUB91:FORX=70TO77:IFG(X)=&HB9
ORG(X)=&H99 ORG(X)=&HE9
THENGOSUB45
103 NEXT:RETURN
105 'declare defects sub----------
110 PRINT@I3,M3$;"DEFECT ENTRY
POINT     ";:PRINT@I1,"<C>ONTINUE
<E>NTER DEFECTS ";:PLAYA$
115 GOSUB70:IFK$="C"THEN RETURN
ELSE IFK$="E"THEN
GOSUB170:GOTO120 ELSE PLAY
E$:GOTO115
120 PRINT@I3,G$;:PRINT@I1,"<I#>
INSERT <D#>DELETE <C>ONTIN";:PLAY
A$:  PRINT@ I3, "";  : INPUTI$:PRINT@I3+
30,F$ ;: L$=LEFT$(I$,1):IFL$="C"THEN
110 ELSE IFL$="D"ORL$="I"THEN 125
ELSE GOSUB 155:GOTO120
125 X1=LEN(I$)-1:FORX2=1TOX1: IF
MID$
(I$,X2+1,1)<"0"ORMID$(I$,X2+1,1)>"9"THENX
2=X1:GOSUB155:NEXT:GOTO120 ELSE
NEXT: I=VAL(RIGHT$(I$,X1)):IFI<0ORI>77
THENGOSUB155: GOTO120
130 IFG(I)=&HB9 ORG(I)=&HE9 THEN
PRINT @I3,"CANNOT OVERRIDE
TESTED RESULTS";:
PRINT@I1,M1$;:PLAYE$:GOSUB70:GOTO120
140 IFL$="I"THENG(I)=&H99 ELSE G(I)=
&HFF
145 X=I:GOSUB45:GOTO120
150 'invalid resp sub----------
155 PRINT@I3,"INVALID ENTRY ";:PRINT
@I1,M1$;:PLAYE$
160 GOSUB70:RETURN
165 'msg clear sub----------
170 PRINT@I3,C$;F$;F$;C$;:RETURN
175 'logo sub----------
180 PRINT@134,"" D I S K  T E S T
"";:PRINT@224,"COPYRIGHT (C) 1983 BY
C C ZIMMER";:RETURN
185 'ml load sub----------
190 FORX=0TOLEN(P$)/2-1:POKEX1+X,
VAL("&H"+MID$(P$,1+2*X,2)):NEXT:RETURN
195 'print question sub----------
200 PRINT@I1,"<P>FOR PRINTOUT
<OTHER> CONTIN";:PLAYA$
205 GOSUB70:IFK$<>"P"THENP=0:
RETURN ELSEP=1:PRINT@I1,"ENTER
DATE  <YYMMDD>      ";:PRINT@I3,
G$;:PRINT@I3,"";:PLAYA$:INPUTD$:
PRINT@I3+30,F$;:RETURN
210 'diskette # sub----------
215 PRINT@I1,"ENTER DISKETTE
NUMBER <###>  ";:PRINT@I3,G$;:
PRINT@I3, " ";  :PLA YA$:INPUTDN: LN$=
LEFT$(LN$,8)+ RIGHT$("000"+ RIGHT$
(STR$(DN),LEN(STR$ (DN))-1),3):
GOSUB 170:RETURN
220 'print id sub----------
225 POKE150,18:PRINT#-2,CHR$(27);
CHR$(23)
230 PRINT#-2,USING"% %d#%% %    % "
RIGHT$(LN$,3),D1,T$,D$;:RETURN
234 'print log sub----------
235 PRINT#-2,TAB(15);:FORX=X1 TO X3:
IFX2=10THENX2=0:PRINT#-2,CHR$(27);
CHR$ (83);CHR$(0);CHR$(27); CHR$(14);
CHR$(27); CHR$(31);:PRINT#-2,USING
"#"; X10;: PRINT #-2,CHR$(27); CHR$
(32);CHR$(27); CHR$ (15); CHR$(27);
CHR$(88);
240 IFG(X)=&HB9 THENPRINT#-2,"-";
ELSE IFG(X)=&H99 THEN PRINT#-2,
CHR$(34); ELSE IFG(X)=&HE9 THEN
PRINT#-2,"."; ELSE PRINT#-2,USING"#";
X2;
245 X2=X2+1:NEXT:PRINT#-2,"":RETURN
250 'drive # sub----------
255 PRINT@I1,"ENTER DRIVE NUMBER
(0-3)     ";:PRINT@I3,G$;:PRINT@I3,"
";:PLAYA$: INPUTD1:GOSUB170:RETURN
260 'test type sub----------
265 PRINT@I1,"<S>HORT TEST
<OTHER> FULL TEST";:PLAYA$
270 GOSUB70:IFK$="S"THENT$="Sh":
RETURN ELSET$="Lg":RETURN
275 'initialization.....
280 I1=481:I3=449:FC=127+16*BC: B$=
CHR$(128):C$=STRING$(30,FC): F$=
CHR$ (FC ):G$=STRING$ (30,32): PP=
26*PEEK (&HC006)+PEEK(&HC007):DI
MG(79): DEFUSR0=&H0E00: DEFUSR1=
&H0E0A:  DEFUSR2=&H0E86: B=&H0400
285 LN$="DEFLOG ":A$="V31;O5; L35;
T50; EP3EP3E":E$="V31;O3; L35; T50;
EP3EP3E"
290 P$="AD9FC00439":X1=&H0E00:
GOSUB 190
295 P$="347FBDB3ED338D002D33CB10
A E8D0 0236F8D00211F31A68D001B815
5 27 0E EC81ED A1A684A7A06C8D000B
20E8 A684A 7A4357F3904 0000DB6DB6
DB6D000000FFFFFF":X1=&H0E0A:
GOSUB190
300 P$="343FBDB3EDBEC006A702860
93D4C5 CE7031F02AD9FC004E60627
10E602 C1112605C C00022 00ECC00
0120091F20810926DCCC0000 BDB4F
4353F39":X1=&H0E86:GOSUB190:P$=""
```

```
305 M1$="<ANY> TO CONTINUE
":M2$="<ANY> TO RESTART OR QUIT    "
310 PRINT@I1,M1$;
315 PRINT@I3,"PLEASE TURN UP
VOLUME ON TV  ";:FORX=1TO150:
NEXT:PLAYE$:K$=INKEY$: IFK$="" THEN
320ELSEGOSUB170: GOSUB255:
GOSUB 265: GOSUB200:GOTO350
320 PRINT@I3,"please";B$;"turn";B$;"up";
B$;"volume";B$;"on";B$;"tv";B$;B$;B$;:
FORX=1TO150:NEXT:PLAYA$:GOTO315
325 'dir error abort point.....
330 PRINT@I3,"DIR. TRACK ERROR-
CAN'T CONTIN.";:IFP=1 THEN GOSUB
225:PRINT#-2, "DIR. TRACK ERROR-
CANNOT CONTINUE-DISK NOT USABLE
OR UNFORMATTED": PLAYE$ ELSE
PLAY E$
335 'restart point.....
340 PRINT@I1,M2$;:GOSUB70:CLSBC
345 'start point.....
350 GOSUB180:PRINT@I1,"<C>
OMMENCE TEST  <Q>UIT
";:PRINT@I3,"INSERT OR CHANGE
DISKETTES  ";:PLAYA$
355 GOSUB70:IFK$="C"THENGOSUB
170: GOSUB 215: GOTO365ELSEIF
K$="Q"THEN GOSUB170: PRINT@I3-33
,"";:GOTO 356 ELSE PLAYE$: GOTO 355
356 IFP=1THENPRINT#2,CHR$(27);CHR$
(19); CHR$ (27); CHR$(54)
357 END
360 'file check.....
365 GOSUB170:PRINT@357, "OPERA-
TION: FILE CHECK ";:T1=17:S1=2:O1=2:
GOSUB 35:IFEC<>0THEN330
370 X1=1:FORX2=0TO77:G(X2)=PEEK
(B+X2): IFG(X2)<>&HFF THENX1=0
375 NEXT:IFX1=1THENPRINT@I3,"NO
FILES     ";:GOTO390
380 PRINT@I3,"files present ";
385 'directory check.....
390 T1=17:S1=3:O1=2:GOSUB35:
IFEC<>0 THEN 330
395 'log check.....
400 LF=1:FORX2=0TO7:IFASC(MID$
(LN$,X2+ 1,1)) <>PEEK(B+X2) THENLF=0
405 NEXT:IFLF=1THENPRINT@I3+15,
"LOG PRESENT  ";ELSEPRINT@I3+15,
"NO DEFECT LOG ";
410 PRINT@I1,"<C>ONTINUE  <OTHER>
ABORT TEST ";:PLAYA$
415 GOSUB70:IFK$="C"THEN GOSUB
170: GOTO 425ELSECLSBC:GOTO350
420 'log array prep.....
425 IFLF=0THEN435ELSEX1=PEEK
(B+13)
430 X2=G(X1):G(X1)=&HB9:IFX2>=&HC0
AND X2<=&HC9 THEN 435 ELSE IF
X2<=77 THEN X1=X2:GOTO 430 ELSE
PRINT@I3,"ERROR IN LOG - WILL TEST
ALL ";:PLAYE$: LF=0: GOSUB80
435 FORX2=0TO77:
IFLF=0THENX1=&HFF:GOTO440ELSEIFG(X2)=&HB9
THENX1=&HB9:GOTO440ELSEX1=&HFF
440 G(X2)=X1:NEXT
445 'first declare point.....
450 GOSUB100:M3$="first
":GOSUB110:GOSUB102:GOSUB110
455 'test sequence.....
460 DATA0,38,2,0,3,39,1,-2,1,3,0,0,1, 0,0,
1,39,2,1,2,38,0,-2,0,2,1,39,2,2, 3,38,0,-2,1,
3,0,38,2,1,2,39,1,-2,2,2, 0,38,2,2,3,39,1,-
2,0,3,0,0,1,0,0,1,39, 2,0,2,38,0,-2,2,2
465 DATA0,38,2,8,3,39,1,-2,5,3,0,0,1, 0,
0,1,39,2,5,2,38,0,-2,8,2,1,39,2,8, 3,38,0,-
2,5,3,0,38,2,5,2,39,1,-2,8,2
470 DATA17,17,1,8,3
475 'DATA0,3,1,0,2
480 DATA99,0,0,0,0
485 CLSBC:X1=.75:RESTORE:FORX= 77
TO34 STEP-1:G(X+2)=G(X): NEXT: DE=0:
G(34)= &HFF: G(35)=&HFF
490 IFT$="Sh"THENFORX=1TO70:
READOP: NEXT
495 READ TI,TE,ST,PA,OP:IFDE=1THEN
565 ELSEIFTI=99THEN FOR X=36TO79:
G(X-2) =G(X):NEXT:GOTO 555 ELSE IF
OP=3 THEN X= USR1(PA): GOTO 500
ELSE IF OP=0 THEN O1=2: T1=TI: S1=1:
GOSUB 35:GOTO 495
500 O1=OP:X1=X1+.25:PRINT@295,"
CYCLE:     ";:PRINT@327,"    TRACK:
";:PRINT@359," granule:    "; :PRINT
@391," OPERATION:     ";: PRINT@
423,STRING$(18,32);
505 POKEPP,O1
510 FORT1=TI TO TE STEP ST:PRINT@
307 ,USING"##";INT(X1);:PRINT@339,
USING" ##";T1;:IFOP=2THEN PRINT @
403,"READ "; ELSE PRINT@403,"WRITE";
515 FORX2=0TO1:PG=2*T1+X2
520 G1=2*T1+X2:IFT1>17 THEN G1=G1-2
525 IFT1=17 THEN PRINT@371, "DIR";
ELSE PRINT@370,USING"### ";G1;
530 IFG(PG)=&HB9 ORG(PG)=&H99
ORG(PG)= &HE9 THEN 545
535 PRINT@423,STRING$(18,32);:X3=2
56*T1+X2:EC=USR2(X3):IF EC=0 THEN
545 ELSE IF EC=2 THEN X2=1:T1=TE:
DE=1: GOTO 540 ELSE PRINT@423,"
BAD GRANULE  ";: PLAYE$
540 G(PG)=&HE9
545 NEXTX2,T1:GOTO495
550 'final declare point.....
555 GOSUB100:M3$="final ":GOSUB110:
GOSUB102:GOSUB110
560 'format allow table.....
565 CLSBC:IFDE=1THEN330ELSEX=
USR1(8): X3=0:X2=0:X1=0
570 FORLG=0TO77:IFG(LG)=&H99
ORG(LG)= &HB9 ORG(LG)=&HE9 THEN
575 ELSE 585
575 IFX3<>0 THEN POKEB+X1,LG ELSE
X2=LG
580 X1=LG:X3=X3+1
585 NEXT:IFX3=0 THEN 590 ELSE POKE
B+X1,&HC9
590 T1=17:S1=2:O1=3:GOSUB 35:FOR
X4=1 TO 200:NEXT:IF EC<>0 THEN 330
595 'format dir entry.....
600 X=USR1(8):FORX=0TO10:POKEB+X,
ASC (MID$(LN$,X+1,1)):NEXT:POKEB+
11,1: POKEB+12,0:POKEB+13,X2:POKE
B+14,1: POKEB+15,0: IFX3=0THEN
POKEB,0
605 T1=17:S1=3:O1=3:GOSUB 35:FOR
X4=1 TO 200:NEXT:IF EC<>0 THEN 330
610 GOSUB 100:GOSUB 630:GOSUB
102: GOSUB 630: CLSBC
615 IFP=0THEN625
620 GOSUB225:X1=0:X2=10: X3=39:
GOSUB 235: X1=40:X2=10:X3=77:GOSUB
235
625 PLAYA$:GOTO340
630 PRINT@I3,"test cplt - FINAL DEFECT
LIST ";:PRINT@I1,"<C>ONTINUE
"; : PLAY A$: GOSUB115:RETURN
```



FARNA Systems

Your most complete source for Color Computer and OS-9 information!

Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com

ADD $3 S&H, $4 CANADA, $10 OVERSEAS

BOOKS:

Mastering OS-9 - $30.00
Update of Paul Ward's "Start OS-9". Completely steps one through learning all aspects of OS-9 on the Color Computer. Easy to follow instructions and tutorials. Comes with a disk full of added utilities and software!

Tandy's Little Wonder - $25.00
History, technical information, hacks, schematics, repairs.... almost EVERYTHING available for the Color Computer all under one cover! A MUST HAVE for those wanting to keep their CoCo alive or for those new to the CoCo.

Quick Reference Guides
These handy little books contain the most referenced information in an easy to find format. Small size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II - $5.00
OS-9/68000 - $7.00

SOFTWARE:

CoCo Family Recorder: The very best in genealogy record keeping EVER for the CoCo! Requires CoCo3, two disk drives (40 track for OS-9) and an 80 column monitor.
DECB: $15.00 OS-9: $20.00

DigiTech Pro: $10.00
Easily add sounds to your BASIC and M/L programs! Very easy to use. Requires user to make a simple cable for sound input through a joystick port. Requires CoCo3, DECB, 512K.

ADOS: The premiere, most respected enhancement for DECB! Support for double sided drives, 40/80 tracks, faster formatting, many extra and enhanced commands!
Original (CoCo 1/2/3) - $10.00
ADOS 3 (CoCo 3 only) - $20.00
Extended ADOS 3 (CoCo 3 only, requires ADOS 3, support for 512K-2MB, RAM drives, 40/80 track drives mixed) - $30.00
ADOS 3/EADOS 3 Combo: $40.00

Pixel Blaster - $12.00
High speed graphics tools for CoCo 3 OS-9 Level II. Easily speed up performance of your graphics programs! Designed especially for game programmmers!

Patch OS-9 - $7.00
The best enhancement for OS-9 short of buying a 6309 and NitrOS9! Contains the latest version of all popular utilities and new commands with complete documentation. Has auto-installer (requires 2 40T DS drives, one may be larger). Ask for 35 track disks (manual install).

HARDWARE:

SPECIAL PRICE ON REMAINING VIDEO DIGITIZERS: $100 (was $125)
This may be your LAST CHANCE to get a DigiScan video digitizer! No MPI -- uses joystick ports. CoCo Max3 , Max 10, Color Max 3 compatible. Input from any VCR, camcorder, or TV camera (freeze-frame required).

Special Prices on Remaining DISTO stock! LIMITED QUANTITIES! BUY NOW!
Mini Controllers: Clones of the Tandy short disk controller, comes with DECB 2.1. NO CASE. $40 each
512K Upgrades: With RAM chips: $50, without: $25
Super Controller I : I have several boards that need repair. Buy two for $30 so you'll have extra parts, or one for $20. Comes with schematic. NO WARRANTY.
HD Controller: $30 - Connect embedded SCSI or MFM/RLL drives (with Adaptec controller) to your CoCo. These fit inside a Super Controller I or II.
Bare Board Set: $30 - Set of blank circuit boards for Disto products including SCI, SCII, HD, 4-n-1, MPROM, etc. Add $20 for complete schematic set.
Complete Disto Schematic set: $25.

FARNA Systems AT306 Based Computers

Complete computer systems based on the AT306 board from Kreider Electronics. Systems are completely setup and ready to go, just add a monitor (or we can supply that too)!

Both systems include:

16 bit PC/AT I/O bus with five slots
MC68306 CPU at 16.67MHz
4 30 pin SIMM sockets
IDE Hard Drive Interface
1.4MB Floppy Drive
Two 16 byte fast serial ports (up to 115K baud)
Bi-directional parallel printer port
Real-time clock
PC/AT keyboard
Desktop Case and Power Supply
BASIC (resembles Microsoft BASIC)
MGR Graphical Windowing Environment with full documentation
"Personal" OS-9/68000 Vr 3.0 (Industrial with RBF)
Drivers for Tseng W32i and Trident 8900 VGA cards
Drivers for Future Domain 1680 and Adaptec AAH15xx SCSI cards

Many other utilities and tools

FARNA-11122 Includes:
2MB RAM
200MB Hard Drive*
Trident 8900 1MB Video Card
$960.75

FARNA-11225 Includes:
2MB RAM
500MB Hard Drive*
Tseng W32i 1MB Video Card
$1114.47

****This is the SMALLEST amount of formatted space (most manufacturers specify UNFORMATTED CAPACITY). Prices fluctuate - we get you the largest drive possible for the money allotted!***

Call for a quote on different configurations and components. Warranty is 90 days for labor and setup, components are limited to manufacturers warranty.

Microware Programmers Package -
Licensed copies of Microware C compiler, Assembler, Debugger, Basic, and many other tools!
***With system purchase:* $65.00 *Without system:* $85.00**

More software will be listed later!

The Embedded Programmer

Paul K. McKneely

An introduction to embedded 683xx controller tools

It was just the other day when an engineer smiled and showed to me with pride a program he had just written to run on MSWindows-95. The program contained many screens worth of code and all it did was to print the traditional "Hello World!" message on the screen! If this is "progress" then it is time to strip the software down to the bare silicon and start over. And if we're going to do that, let's backup one more step and make a better choice of silicon to begin with.

Now-a-days this is not as bad as it seems. There are a wealth of tools available for those who resolve to master the binary world of system software. Those who shed the comfort of familiar desk-top operating systems to write programs that run right on top of the hardware are called Embedded Programmers. Their products are called Embedded Systems. This is a large field in today's market place and it only promises to get larger in the future. The Motorola 68K family has enjoyed more roles in microprocessor (mpu) based systems than any other, including embedded systems. That is because it is so flexible, easy to use, and so well supported by many vendors.

Software development involves two kinds of computer systems: The Host and the Target. Those who use MS-DOS and OS-9 are familiar with what is called Native Development. This is when you build a program on the same system that you will run it on. In may cases, embedded systems do not have all of the resources that are necessary for software development. Here you need a Host with a user interface, disk drives, printers, and a large amount of RAM. When the program is ready to run, it is "down-loaded" to the Target computer that it will run on. Because typical targets don't have user interfaces, the host often plays that role for testing and debugging purposes. This form of development is called Cross Development. It allows programmers to throw off the overhead of the host computer in their program executables without giving up the conveniences of their development environments.

For several years I have been doing 68K cross development using the IBM-PC as my host. I have used several different 68K designs as targets that run my programs. The IBM-PC is particularly good as a cross development host because it's OS is so primitive and because it is so well supported by the tool developing industry. Having a primitive host architecture gives you more freedom to write programs that simulate the target's behavior because it is easy to by-pass the host's native interfaces. But I chose the 68K as my target because it is so much easier to design excellent software for anything I want to create. 68K-based hardware is plentiful and diverse. This great diversity has been its own worst enemy because it makes it hard to find one platform that can be considered a "standard". But processors of this family have even more potential than the popular minicomputers (such as VAXes) of the '70s and '80s because they are so inexpensive. All they need is software that is worthy of the hardware it runs on.

In all cases, a target computer must have some way to communicate with the host. In the simplest (and cheapest) case, the target will have an Erasable Programmable Read-only Memory (or EPROM). The process of burning an EPROM and inserting it into the target's socket is rather time consuming. When you make changes to the software, you must remove the chip, erase it with an ultra-violet eraser (which takes around 15 to 40 minutes), and re-program it with the new version. This kind of communication tends to be cumbersome and begins to wear out your target's EPROM socket pins. A better (but more expensive) solution, is to use an EPROM Emulator. This is an embedded system in itself which uses RAM to mimic an EPROM. Your host communicates with the mpu in the emulator via a serial or parallel interface. It downloads the image data to the RAM that appears to your target where it expects its EPROM to be located.

There is another method (the most expensive) of communication between a host and a target that uses an In-Circuit Emulator (or ICE). This is a device that replaces the target's u-processor but can do a lot more than the mpu can. With an ICE, your target doesn't even have to have any ROM because it can download images through the micro-processor's bus interface. These devices are beyond the reach of your average hobbyist because they are so expensive. But they can even assist in hardware development because they can inform you of the states of individual pins as well as read and write memory locations without having to run any code.

Probably the best, and most cost effective, method of embedded development for the average hobbyist is to write some code that can eliminate the need for an EPROM Emulator. This is a simple program that you burn into an EPROM. It has to do little more that to accept a program downloaded via a serial port and copy it into RAM. If the target sees the proper format and detects no transmission errors, it jumps into the program which takes over control of the system. I have written such a program (I call it PREBOOT) and it runs on the Computer Design Services CD68X20. Since June of 1995, a friend of mine (who I met over the Internet) and I have developed a set of cross development tools that run on an IBM PC equiped with a standard VGA card.

These tools, together with PREBOOT, make it relatively easy, and inexpensive, to write embedded code for 68K targets that are available off-the-shelf. 68K-based computers range from truely embedded systems such as smart multi-serial I/O cards, smart ethernet protocol-processing cards, laser printers, and terminal servers to complete systems made by Hewlet-Packard, SUN, Apple, Atari, Next, and Commodore, not to mention Motorola. My favorite among these "high-performance" systems are the VME-based systems because they usually come with excellent hardware documentation; something you usually don't get with workstations. Now is a particularly good time to buy used 68K-based VME equipment via the Internet. A German student recently told me that companies are throwing away (or giving away) Motorola MVME147's. This is a high-integration board that uses surface-mount technology to achieve better performance than the average machine you will usually see running personal OS-9. It has all the features you would expect of a full system except a user interface. I bought one brand new for $6,000 back in 1990 and a used one a couple months ago for $500. I hate to mention how little I paid for one last week!

For the PC-compatible class, I like to use the boards made by Peripheral Technology/Computer Design Services because they come with schematics and excellent personal support from their designer, Fred Brown. These boards don't have the high performance of VME but they have the advantage of being able to use inexpensive PC hardware. The motherboard comes with more on-board devices than your typical Intel motherboard and is ready to be mounted in a PC cabinet. The CD68X20 is interesting in that the CPU comes on a daughtercard. Currently, the 68020 is the only CPU available but I am now talking with Fred Brown about engineering a 68030 CPU card for the CD68X main board. Here is a list of the CD68X20's built-in features:

* 25/33 MHz 68020 /w 68882 socket
* 2 28-pin EPROM Sockets
* 4 Serial Ports via two MC68681 DUARTs
* 2 Parallel Ports via MC68230 P/IT
* Socket for WD33C93 SCSI Interface
* IDE Drive Interface
* 6 16-bit ISA Slots
* MK48T02 2K BBRAM Clock/Calender
* 8 30-pin SIM Sockets (up to 128 MB)
* 8-bit A/D Speaker Interface
* IBM-PC/XT Keyboard Interface

Although I have been using this computer as an embedded programming target computer, it is capable of being the heart of a very powerful computer system. For those of you who desire to get down to the bare silicon, it is an excellent vehicle to learn on. I have three CD68X20's and I have been using them to develop a Kernel for embedded, real time, and general purpose use. The key to building a good operating system is in solving the problem of making a single processor to be able to handle many things well in parallel. There is a clever technique for doing this efficiently by making effective use of the 68K's interrupt structure but the way I did it requires some minor board modifications. These board modifications will not interfere with OS-9 for those of you who would like to keep their options open for that operating system which can be purchase from CDS.

I hope a few of you are guessing at where we are headed with this. In future articles I would like to demonstrate the use of the tools that David and I have written. I will discuss the 68K's interrupt structure in detail and explain its boot up process. This will prepare us to explore the fundamentals of how to write a high performance "micro-kernel" and how standalone embedded programs can be written using these tools along with the PREBOOT program. I will explain the board modifications that are necessary to run the PhiKernel and why they are necessary. Soon you will be off and running, writing your own standalone "embedded" programs. But don't let me fool you. Where I intend to eventually lead you is to a new operating system that I have been developing for several years, PhiOS (pronounced fi-oh-ess). It should be a fun experience. Those of you who really want to know how high performance computer systems work, come with me. We're in for a great adventure!

For those of you who would like to follow along in this series of articles titled "The Embedded Programmer", you can obtain a CD68X20 from its designer Fred Brown at:

Computer Design Services
1250 Piedmont Rd.
Marietta, Ga. 30062
Voice: (770) 973-2156
FAX: (770) 973-2170

Those of you who would like to respond to me directly about the contents of this article can reach me at:

techno Venture, Inc.
P. O. Box 5641
Pasadena, Texas 77508
E-Mail: gecko@onramp.net

Operating System Nine

Chris Burke

Install Multi-Vue on your hard drive!

Before we start, you need to know what my system looks like:

CoCo 3, 512K
Hi-Res Mouse
20 Meg Hard disk (set up as "/d0")
35 Track floppy (set up as "/d1")
Parallel printer (set up as "/p1")
Tandy ACIAPAK (set up as "/t2")
WIZ ACIA driver (set up as "/m2w")
Magnavox (RGB) monitor

The first thing you should do is read the "Getting Started" section of the manual. Then read pages 9-28 through 9-31, which give you the format of MV's "environment file". The environment file configures MV to your system, and you will have to modify it in order to successfully install Multi-Vue.

Now, make a backup of the release disk(s). Tandy has distributed Multi-Vue on 2 sides of a single floppy disk, so you need to back up each side to a seperate "normal" floppy disk. Label the backups DISK #1 and DISK #2. Put the master in a safe place, and use ONLY the 2 backups for the rest of this procedure.

Here we go:

1) Make sure that all of the files from your Level 2 OS-9 release and CONFIG disks are on the hard drive. If they are not, you can put the OS-9 release disk in /d1 and enter:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

Then put the CONFIG disk in /d1 and enter:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

This takes a while, but does the trick. Remember, /d0 is the hard drive in the above example! If your hard drive has a different name, don't forget to change when typing.

2) Put DISK #2 in /d1, and copy everything from it to your hard drive:

```
OS9:chd /d1
OS9:dsave -s20 /d1 /d0 ! shell
```

3) Unless you want Multi-Vue to boot automatically, you must delete the AUTOEX file from your hard drive:

```
OS9:del /d0/cmds/autoex
```

It seems that this is one Tandy didn't tell us about. Any file named /d0/cmds/autoex will boot automatically whenever OS-9 is started.

4) Put DISK #1 in /d1, and copy the contents of its CMDS directory to the hard disk:

```
OS9:chd /d1/cmds
OS9:dsave -s20 /d1 /d0/cmds ! shell
```

5) Copy the environment file from DISK #1 to the hard drive:

```
OS9:copy /d1/sys/env.file /d0/sys/env.file
```

6) Edit the environment file to tailor it to your system. I had to change the following entries:

OLD FIELD	NEW FIELD	Description
MONTYPE=0	MONTYPE=1	Select RGB monitor
RAM=128	*RAM=128	Comment out 128K code
*RAM=512	RAM=512	Use 512K code
PTRRES=0	PTRRES=1	Hi-Res mouse
RBFDEV= /d0,/d1	RBFDEV= /d0,/d1, /d2,/d3, /h1,/dd	Register all RBF devices
SCFDEV= /p,/t1	SCFDEV= /p1,/t2, /m2w	Register all SCF devices
PRPORT=/p	PRPORT=/p1	Use my printer driver

7) Add the following line to your startup file:

```
merge /d0/sys/stdfonts /d0/sys/stdpnts_4 /do/sys/stdptrs
```

8) Build a short command file to start Multi-Vue:

```
OS9:build /d0/xmv
tmode -echo
control -e
gshell &
tmode .1 echo
```

The CONTROL program uses the environment file to set up some of Multi-Vue's parameters, etc. GSHELL is the new graphic shell that is supplied with Multi-Vue; it will automatically create a graphics window for itself any time you run it.

9) Make a new OS9 boot disk that supports Multi-Vue. This will be identical to your existing boot file, with the following exceptions:

i) Use WINDINT.IO instead of GRFINT.IO

ii) Include W7.WD through W12.WD, to provide additional windows. NOTE: if you have 512K, you can also include W13.DD, W14.DD, and W15.DD

In this case, the easiest way to make a new boot disk is to use OS9GEN. If you have added anything to the standard Tandy boot, you probably already have a BOOTLIST file around somewhere — just add the new modules to it. For the standard Tandy boot, you can use the file "/d0/modules/bootlist.mv". In any case, you must put a new disk in /d1 and enter:

```
OS9:format /d1 r " "
OS9:os9gen /d1 #40K </d0/modules/bootlist.mv
```

10) Users of XT-ROM may wish to install the new boot file on the hard disk. To do this, enter:

```
OS9:bootport /d1 /d0
```

11) Take a break. After all this you deserve it. Of course, you've come this far, so . . .

12) Boot the new system, either from the hard disk or the floppy disk.

13) THIS IS IT! Run Multi-Vue by entering:

```
OS9:xmv
```

Now hit the CLEAR key until you end up in th Multi-Vue GSHELL window.

14) Each time you execute the XMV file, another Multi-Vue graphic shell will be spawned. You can rotate between all of your spawned SHELL and GSHELL windows by depressing the CLEAR key.

A Brief Review of Multi-View

Summary: I give it a 5 (out of a possible 10). Impressive, as long as you didn't expect a Macintosh (which I did . . .). The screens and icons look great. I wish that you could have MV come up in HI RES screen mode. As it is, you have to bring it up in LO RES and change over. Mouse operation is very smooth, and the default mouse speed is very comfortable (hi-res mouse).The documentation looks very good. There is an extensive section on accessing the windows from "C". You cannot drag icons around on the screen, and there is no TRASH icon (Apple computer has a copyright on the trash can anyway). All file manipulation is done by typing in file names from the keyboard. You cannot "double click" to execute an application.

There is no good way to open multiple directories on the desktop. You can only have one directory open at a time. Of course, since you can't drag icons around, much of the reasoning behind having multiple directories open does not apply.

GSHELL is written in "C", and it takes up about 16K (without any merged utilities). The standard OS-9 shell is written in assembler and uses about 1.5 K. This is a very big difference, and means you may not be able to run big applications (like the "C" compiler) from within GSHELL.

Several nifty "pop-up" utilities are provided with Multi-Vue. These include a clock, a calendar, a decimal and HEX calculator, and a method for escaping to the standard OS-9 shell. The clock utility draws an analog clock face on the screen and updates it in real time! The clock face is oblong, rather than round, and the proportions of the hands make the clock difficult to read — especially in HI RES mode.

The calculator is very handy, especaily the HEX feature (for programmers like me). The screen definitions of the key boundaries are off a little bit; for example, the "=" key often registers as "3". Also, you must press the keys with the mouse kind of slooooowly if you want them all to register.

Don't worry!
Rick Ulland will return with his usual hints, tips, and wit in the next issue!

Atlanta Computer Society's Picnic In The Park Stone Mountain Park, GA!

Rather than having a regular CoCoFest this year, ACS has decided to have a picnic at Stone Mounatian Park, just east of Atlanta. It will be held on October 5th, 1996. I will have more information in the next issue. For those requiring more info sooner, please contact ACS at: P.O. Box 80694, Atlanta, GA 30366; BBS 404-636-2991. You may also contact Frank Swygert via E-mail: dsrtfox@delphi.com

Color Computer Disk Drives

Mixing and matching disk drives for the Color Computer

Frank Swygert

I've had several inquiries on disk drives for the Color Computer... enough to warrant writing this article. The questions are a good sign, actually. They indicate that there are still a few beginners out there trying to learn more about their systems... and trying to keep them running a while longer. I'll cover enough to make this interesting reading even for the more seasoned CoCo veterans, so don't turn the page just yet!

Disk Basics

The very first storage media for "personal" computers was punched paper tape. This was just what it sounds like... a strip of approximately one inch wide paper with a series of holes punched across it. The holes translated into characters on the screen or in memory. The reason this was used was because it was cheap. There were disk drives available at the time, but they were "hard drives" and very expensive... try over $10,000 for a 5MB or so drive (these were used for mainframes). For those who have seen them, the paper tape was the same as used with Teletype machines. That's why they were so cheap at the time.

It didn't take long for computer hobbyists to adapt the audio cassette recoder for data storage use. Here was a common, inexpensive format. The problems were that it wasn't random access, there were no standards, and reliability wasn't the best with most systems.

Then IBM introduced the "mini-floppy" drive. This started out as an eight inch disk, but eventually got down to the current three-and-a-half inch variety. The "mini-floppy" became popular with business computers, and was soon introduced in prices that hobbyists could afford. Nowadays, a floppy drive is essential. Prices have come down to the point that there is no real reason for any computer, even the "classics" that are hanging on like the CoCo, to not have a floppy attached.

Let's take a look at what floppy disks have in common first. The least common denominator is the media itself. The media that is actually written to is a paper thin piece of plastic (usually mylar) with a magnetic coating. This is encased in either a flexible (5.25" disks) or hard (3.5" disks) plastic protective covering. This covering protects media from scratches and dust. 3.5" drives have a little metal shield that slides over the media access opening. 5.25" disks lack this cover and must be placed in "jackets".

The media coating is different depending on the density of the disk. "Density" refers not only to the number of tracks on a disk, but to the amount of data that can be stored on a track. There are basically two track densities. 360K and under disks usually have 48 tracks per inch and 256 bytes per sector. These are referred to as "double density" disks (the first disk drives only stored 128 bytes per sector). 720K and greater disks have 96 tracks per inch. 720K disks are double density, but the higher capacity 1.4M disks are called "high density" instead of "quad density", which at first glance would seem to make more sense (they store 512 bytes per sector, and the latest 2.8M disks 1024 bytes per sector). Only 40 (360K) and 80 (720K and up) tracks are actually used on the floppy disk. A more sensitive coating is used on the high density (1.2M 5.25" and 1.4M and 2.8M 3.5") media. The high density media is so sensitive that if it is formatted using a low density drive, it will no longer format at high density, and may not be reliable at low density either. In other words, the disk is ruined. While many people punch holes in the 3.5" double density disks and format them for high density use, this is not recommended. Information on such a disk will gradually "disappear", as the magnetic coating isn't sensitive enough to hold the data for long periods. Note that the 2.8M drives use the same high density disks as 1.4M drives.

Another drive identifier is the height of the drive unit face. A "full height" 5.25" drive is approximately three inches tall, measured at a right angle to the disk slot. "Half height" drives are therefore approximately 1.5" tall. 3.5" drives are designated a little differently, since there never was a "full height" model. When first introduced the 3.5" drive was approximately 1.25" tall, but the newer models are only 1".

When speaking of older drives, there is another item to consider — the number of "sides". A single sided drive reads only the "bottom" side of a disk. A double sided drive naturally has a read/write head on each side. In the early days, single sided drives were cheaper and easier to build. Even IBM used a 35 track single sided drive in their first PC. Many early computer users maximized their disk use by buying or making "flippy" disks. These had a second timing hole and side notch that allowed one to literally "flip" the disk over and record information on the "back" side. CoCo owners used this method often.

I realize that the above explanation is simple, but the reason for this article isn't to explain all about disk drives. The preceding was just to give a little background information.

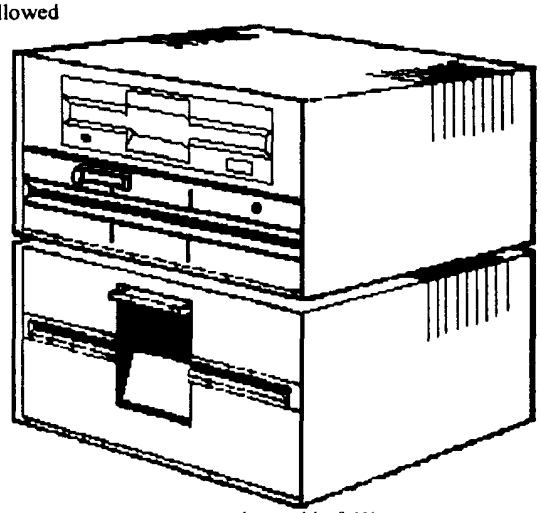
Tandy Color Computer Disk Drives.

Tandy first introduced disk drives for the CoCo in November of 1981. This was a double density single sided full-height 35 track 5.25" drive with a six inch long controller and a flat ribbon cable (catalog number 26-3022). Floppy drives were just starting to come into widespread use at this time, and some of the older systems used these 35 track drives. This included Tandy's TRS-80 line. Naturally, these drives and enclosures were the same as the TRS-80 Model 1 and Model 3 units. These systems retailed for $499.95. These early model controllers required 12 volts to operate. The only reason one would be desired now is that they can be modified to support high density drives. As they are, they only support double density (up to 720K).

In 1983 Tandy upgraded the disk system with a new five volt only controller. The new controller was necessary since the then new CoCo 2 didn't provide 12 volts on the cartridge port. To use the old controller with the new computer either a Multi-Pak Interface (MPI) was required or 12 volts had to be supplied. Many users simply borrowed 12 volts from their disk drive power supply and ran it back to the controller. This was called the FD-500 system, catalog number 26-3129.

Tandy upgraded the disk system again in 1986, this time with a more compact five volt controller and a 40 track single sided drive (FD-501, catalog number 26-3131). This was mostly due to the fact that 35 track drives had been phased out by the manufacturers. Tandy could actually buy the 40 track drives for less than 35 track models. Note that Tandy never upgraded the Disk BASIC ROM code to access the additional storage capacity. One modified the ROM code, bought a third party extension (such as ADOS), or used OS-9 to use the additional drive capacity. The standard for the CoCo was 35 tracks throughout its lifetime.

With the advent of the CoCo 3, Tandy again upgraded the drive system (FD-502, catalog number 26-3133). This time double sided 40 track drives were used. OS-9 was growing in popularity amongst CoCo users and really needed the extra disk capacity. In order to foster a growth in OS-9, Tandy announced that they would only be interested in marketing



CoCo 3 software if it were written for OS-9. Unfortunately, this was a little to late in the game ot have any impact on sales.

Note that Tandy used basically standard, readily available drives. Other manufacturers (especially Apple, Atari, and Commodore) used modified electronics in their systems. This means that upgrading the CoCo drive system is relatively inexpensive, since the IBM/PC clones use the same drives.

Upgrading CoCo Drives

If you are still using full height drives, please consider upgrading! The old full height units are slow and eat up a lot more power than newer half height drives. Double sided 40 track (360K) can be purchased new for as little as $10-$15. All one has to do is remove the old drive and plug in a new one. While you're at it, go ahead and plug in two. The second drive is really worth the money and backups will go much faster. Two drives are really a necessity when working with OS-9.

Note that all CoCo drive controllers can support up to 720K drives, whether they are 5.25" or 3.5". At one time the 5.25" 720K drives were very popular with OS-9 users. They were cheaper than the 3.5" models and could be "double stepped" (head moved two tracks at a time) to read (but not reliably write) 360K disks. While many of these are still in use, they are not recommended as upgrades now. The 3.5" 720K drives are getting hard to find also, but that isn't as much of a problem as it may seem. Most of the newer 3.5" 1.4M drives can be set to operate as a 720K drive. Check with dealers or manufacturers before buying to make sure. This is necessary since 1.4M drives spin faster than 720K drives. Some drives have a jumper to force them to always operate at the slower speed. With OS-9, you need all the storage capacity you can get, so the 3.5" 720K drives make a lot of sense! They run $25-$30 new. It is a good idea, however, to ALWAYS have at least one 5.25" 360K drive in the system. Single or double sided disks can be formatted, copied to, and read, making it easier to swap files and disks with other users.

The old full height drive cases have plenty power to operate two floppy drives easily, especially the cases with the transformer hanging outside the back of the case. The FD-502 power supply is a little weak, so be careful with it. Tandy even supplied a small fan with the Drive 1 kit, so a fan is definately recommended. A small 2" fan can be purchased from any electronics store and added inside the drive case. It may take some ingenuity to mount, but that double sided picture hanging (foam core) tape sure comes in handy...

Another problem one may run into is the power connector. 3.5" drives use a different power connector than the standard 5.25" drives. An adapter can be purchased from most computer stores (including Radio Shack) and plugged between the drives. Many users simply snip the old connector off and solder on a new one of the correct type or add another set of leads and a connector.

The drive cable itself needs to be looked at. Older drives use a flat ribbon cable. Take a close look at the connectors. Tandy's first drive systems had all the drive selects jumpered on the drives. A drive knew which number it was by the position on the cable — the "teeth" in the cable that selected drives were missing except the one for the appointed number. If your cable has connectors with missing teeth you can get new connectors from Radio Shack and crimp them an inch or so away from the original connectors. Then use the drive select jumpers to designate which drive is which. The jumpers are located near the connector on the drive circuit board. They will be labeled "DS0" through "DS3", designating drive 0, 1, 2, or 3. Place the jumper over the numbers you wish the drive to be. Some drives may use letters instead of numbers (A-D), and some newer ones will only have two selects. If there are other jumpers nearby, don't move them.

If you find that you have a cable with a few wires twisted, you have an IBM type cable. IBM ships their systems with both drives (if there are two floppies) jumpered as drive one (or A). The twist in the cable moves the drive zero select line to the drive one line. The drive on the end of a twisted cable is always drive zero (A), and the one on the center connector one (B). This was supposed to make it easier to install the second drive later... no jumpers to fool with! Most CoCo systems used a flat cable, but this is good information if you buy a new cable through a PC dealer or find a system set up this way.

The FD-502 used a round cable between the drives and controller. This is basically a flat ribbon cable rolled into a plastic jacket. This was done because the FCC regulation governing radio emissions from electronic equipment was made more stringent. There is some extra shielding inside that plastic jacket also. Tandy left just enough room between the two drive connectors to connect one drive directly under the other. Since there has never been a standard location for the connector on the drive (except at the back!), this can pose a problem. The solution is very simple. Cut the wire tye holding the cable to the case. Then carefully slit the cable jacket. There is a flat section of cable every few inches where a new connector can be crimped on. If this sounds like to much, get a flat cable made or get a cable from any PC dealer and install as previously mentioned.

Disk drive connectors are pretty standard. Cables are usually marked with one wire colored or striped. This wire should be connected to pin one of the connectors. The pin numbers are marked on the drive circuit board, so make sure pin one is on the correct side of the cable. If by chance you connect one (or both) of the drives wrong, don't panic. The lights and motors of both drives will come on, but no harm is done. Find the connector that is backwards and flip it over. Note that 5.25" drives use an edge card connector but 3.5" drives use a pin connector. The connectors are readily available at Radio Shack, so if you plan on using a 3.5" drive on your existing cable get one and crimp it on. Alternately, PC cables usually have both types of connectors, so you may want to get a PC cable. Some PC controllers use a pin type connector on the controller end, the CoCo requires an edge card connector. Make note of this when purchasing a cable from a PC dealer!

A mounting adapter can be readily purchased to put the 3.5" drive in a 5.25" opening. Make sure you get the faceplate with it, or be prepared to make a filler plate from cardboard. Most adapters available now are made for 3.5" hard drives and don't come with the filler/face plate. Some of the old full height drive cases won't be drilled to mount more than one drive, so some drilling may be required. If the sides of the case don't extend far enough to mount a second drive, drill a plate to mount one drive on top of the other, using the side mounting screw holes to secure the drives together.

OS-9/68000 Machines

Like the newer Windows based machines, OS-9/68K machines have adopted the 3.5" 1.4M drive as standard. Unless there is a need to read files from a CoCo with only a 5.25" drive, there is no real need for a 5.25" drive. The FHL KiX and other Hazelwood based machines have controllers that support 2.8M drives. Machines that use standard PC type controllers are capable of using 2.8M drives provided the controller and drivers support this format. The 2.8M drives never have really caught on. The reasoning is simple: almost all modern computers (and I can't think of a single exception) use hard drives, and most also have CD-ROM drives. The only real purpose for a high capacity floppy is to backup the hard drive or maybe distribute software. Tape backup systems are much more efficient than using floppies for backup, and large programs are more likely to be distributed on CD-ROMs than on floppies.

The days of the floppy only system is pretty much over. There are a few of the older systems, such as the CoCo, that will perform reasonably well with just floppies, but any up-to-date system requires over 4MB or RAM, 100MB of hard drive space, a CD-ROM, and a high resolution monitor. And that is just a very basic system! Progress... where is it taking us? We used to be able to do so much with less... at least the costs are still on par! (A basic CoCo floppy system costs nearly $1000 when first introduce... computer, drives, and all... a basic 486 or AT306 system costs about the same today).

Basic09 In Easy Steps

Chris Dekker

Windows... can't live with them, can't live without them!

...Of course this has been said about many entities (dead or alive) throughout history. My guess is that we could live without them if we had to, but our programs would look nowhere near as impressive. There are more legitimate reasons for using windows. Desktop packages, for instance, or GUI's (graphical user interfaces) would leave you with a cluttered (and mostly useless) screen in no time if they couldn't use windows.

Another very important reason shows up with multitasking operating systems like OS9. We want to have a series of windows into the computer so we can actually see what the various programs are doing. If OS9 had no windows you could always launch 3 or 4 background tasks and hope that none of them ever encountered an error. If they did, your means of communications would be limited to an occasional beep or a messed up screen.

So let's cherish our windows! But how do they work? On the surface it is easy enough: RUN gfx2("dwset",...) will get you a window to work with. However to make the most of them, you need to have some understanding of what's actually going on. In this article I will try to get you underway.

Under OS9 the windows system is a part of the operating system. It consists of the GrfInt module (WindInt for MultiVue, which includes all the code of GrfInt) and a subroutine package in the GrfDrv file. This package contains the actual code for functions like handling windows and graphics buffers or drawing letters or arcs on the screen.

The code in GrfInt checks and sets up variables for the various functions. It also contains a number of system calls like the code that returns screen size, type, colors, etc. The extra code in WindInt mostly deals with what's called the high level window handler. This code sets up and updates menubars, scrollbars and a window's borders and control regions.

What you won't find here is the code that handles your mouse. Although a mouse is usually associated with a windows environment, it is an input device and handled by the CC3IO module. Similarly, AIF's and icons (under MultiVue) are not part of the windows system but are handled by GShell. According to OS9, this is simply another application.

I am making these distinctions to show that, even if you have only "plain vanilla" OS9 and Basic09, you can write programs that make full use of the windows. It just takes a little ingenuity and some extra work. For instance: MultiVue's icons are 24x24 pixel GET/PUT buffers. You can put them on the screen just as easy from a Basic09 program as from within GShell.

Another avenue would be to create or use a subroutine package to draw borders on the windows. A public domain package called Gfx5, written by Floyd Resler, does an excellent job in this respect. It also handles menubars, scrollbars and pulldown menus with simple RUN statements.

The only drawback on the CoCo is that you have to use the graphics screens to make your setup visually appealing, but this slows things down quite a bit. Fortunately the 6309 processor can overcome these problems. You will need a set of screen drivers specifically written for the 6309, but it can handle windows as if you had a coprocessor installed.

OS9 recognizes 2 types of windows: device windows and overlay windows. A device window is the window we use to run programs in; most of the time anyway. If you boot up using the TERM_WIN descriptor, you will be looking at a device window when the green 32x16 screen disappears. This device window is called Term. Other device windows are called w, w1, w2, etc. OS9 can handle up to 16 of these device windows (assuming you have enough memory available) and up to 32 windows in total. This total counts both device and overlay windows.

So what does a window look like to OS9? To the core operating system it is just another device, like a disk drive, a printer or a modem. On the other hand you could call a window a virtual device in the sense that it exists only in cyberspace and vanishes the moment you turn off your CoCo. From this point of view a window is only a chunk of memory, connected with a couple of smaller chunks of memory that hold it's vital parameters.

The reason we can see a window is that some of these parameters get jammed into the GIME's registers. This causes the hardware to present us with with a certain interpretation of the data in a memory block. If we are lucky this data will make sense to us and we call it a window; if we can't make sense of it, we generally call it a crash!

The smaller chunks of code mentioned above are part of the internal data structures set up by OS9 to keep it's house in order. One of them is called the "window table". You may have come across this name in the error messages.

The window table is a 512 byte block of memory in (MMU) block 0 (at least on a CoCo). For the curious: it starts at offset $1A80. This block is divided into 16 32-byte blocks. Each block is associated with a different device window. It contains the window's type, memory block, starting address and fore/background and border colors, as well as a window's current palette. Every time you push the clear key (to switch windows) most of this information is written into the GIME's color and DAT registers, so you will see the window you expect in it's own colors.

Note that if you use the command RUN gfx2("palette",...); you change a byte value somewhere in this window table. Using the "defpalette" command changes a byte in a different memory block. This block is also situated in block 0 (offset $10C7), but resides outside the windowtable.

Each window has also a different block of data associated with it. I don't know what it's official name is, but analogous to OS9's path descriptors and process descriptors we'll call it a windowdescriptor. (Note that this is not the same as the device descriptor for a window which is loaded with the bootfile.)

Window descriptors are 64 bytes long. OS9 (during boot up) reserves space for a list of 32 descriptors; starting at offset $1280 in block 0. The parameters in this block really define the window and what's on it. Among them you'll find the window's size and working area, cursor position, screen attributes and flags. You will also find links to graphics, font, pattern and gfx cursor buffers, and to saved overlay windows so the original screen can be restored if the overlay window closes.

Apart from that the block holds more arcane data like register masks, scale factors, jump addresses, etc. As you can see, it takes a lot of parameters to define a virtual device. You see, the computer addresses screen memory as a long string: it really has no clue where one line ends and the next one starts unless you give it some pointers and starting addresses. Communicating with a printer, for instance, is much easier because all those tasks are taken over by the printer's own microprocessor and associated hardware.

Unlike the window table which contains only entries for DEVICE windows; window descriptors are also initiated if you open overlay windows. Every time you open an overlay window it gets linked to the underlying window by a pointer. If this pointer (at offset 2 in a descriptor) reads $FF it means you are looking at a device window: the bottom the stack.

For those of you who have gotten really curious what these things look like in real

life, I have included the following program. It gives you a hexdump of block 0. I admit it is somewhat crude, but.. nobody is going to stop you from experimenting your way to a more useful program. You can add some code to dump modules (a F$Link system call) or files.

```
  PROCEDURE dump
   TYPE registers=cc,a,b,dp:BYTE; x,y,u:
INTEGER
  DIM regs:registers
  DIM i,pointer,blockend:INTEGER
  DIM paus,options(32):BYTE
  regs.a=1 \regs.b=$26
RUN syscall($8D,regs) \(* get screen size
  IF regs.x<80 THEN
    RUN gfx2("dwend")
    RUN gfx2("dwset",2,0,0,80,24,1,0,0)
    RUN gfx2("select")
    RUN gfx2("palette",0,0)
\(* black background
    RUN gfx2("palette",9,63)
\(* white letters
  ENDIF
\(* now we have an 80 column window
  regs.b=0 \regs.x=ADDR(options)
  RUN syscall($8D,regs)
\(* get window parameters
  paus=options(8)
\(* preserve current status
  options(8)=1
  RUN syscall($8E,regs)
\(* force end of page pause
  regs.x=0 \regs.b=1
  RUN syscall($4F,regs)
\(* map block 0
   IF LAND(regs.cc,1)=1 THEN ERROR
regs.b \ENDIF
  pointer=regs.u \blockend=pointer+8192
  WHILE pointer<blockend DO
    PRINT USING "x2,h4";pointer-regs.u;
    FOR i=0 TO 15
                    PRINT USING
"x1,h2";PEEK(pointer+i);
    NEXT i \ PRINT
    pointer=pointer+16
  ENDWHILE
  RUN syscall($50,regs)
\(* clear block 0
  options(8)=paus
\(* restore window status
                regs.a=1      \regs.b=0
\regs.x=ADDR(options)
  RUN syscall($8E,regs)
  END
```

EZ135...

continued from page 6

```
15 PRINT■RGB-DOS only uses 256-byte sec-
tors.  If your hard drive only supports
512-byte secotrs, RGB-DOS will only use
half of  it.  To determine how many
sectors your drive has, use the HD-
UTIL.BIN program.■:PRINT
20 PRINT■Does your drive use (2)56 byte
or (5)12 byte sectors?■;
25 A$=INKEY$:IF A$=■■ THEN 25 ELSE IF
A$=■2" THEN SZ=256 ELSE IF A$=■5" THEN
SZ=512 ELSE 25
30 PRINT SZ:PRINT
35 LINEINPUT■How many total sectors?
■;A$:IF A$=■■ THEN 35
40 IF LEFT$(A$,1)=■$■ THEN TS=VAL
(■&H■+RIGHT$(A$,LEN(A$)-1)) ELSE
TS=VAL(A$)
45 TS=TS-1:IF TS<1 THEN 35
50 MX=INT(TS/630):PRINT:PRINT■Total sec-
tors entered as■TS:PRINT■which is enough
for■MX■RS-DOS drives.■
55 IF MX>256 THEN MX=256:PRINT■RGB-DOS
supports up to 256 RS-DOS drivesper hard
drive.  Any remaining space canbe used
for an OS-9 partition.■:PRINT
60 PRINT
65 PRINT■How many RS-DOS drives ( 1 -
■MX■)? ■;:LINEINPUT A$:IF A$=■■THEN 65
70   IF   LEFT$(A$,1)=■$■   THEN
DR=VAL(■&H■+RIGHT$(A$,LEN(A$)-1)) ELSE
DR=VAL(A$)
75 IF DR<1 OR DR>MX THEN 65
80 RS=DR*630:OS=TS-RS:PRINT:PRINT■You
have          requested■DR■RS-DOS
drives,■:PRINT■taking up■RS■sectors of
hard    drive■:PRINT■space    and
leaving■OS■for OS-9.■
85 PRINT:PRINT■Is this acceptable (Y/
N)? ■;
90 A$=INKEY$:IF A$=■■ THEN 90 ELSE IF
A$=■Y■   THEN   PRINT■Yes!■   ELSE
PRINT■No.■:GOTO 60
95  RK=(RS*SZ):PRINT:PRINT■RS-DOS
:■INT(RK/1024/1024)■mb /■RK■K■
100 OK=(OS*SZ):PRINT■OS-9     :■INT(OK/
1024/1024)■mb /■OK■K■
105 PRINT:IF SZ=512 THEN PRINT■NOTE:
Since RGB-DOS does not fully use 512-
byte sectors, half of the space willbe
wasted.■:PRINT:PRINT■ALSO:  If you are
not using OS-9 driversthat support 512-
byte sectors, storage will also be half
of what is listed.■
110 OF=TS-RS:PRINT:PRINT■The RS-DOS par-
tition will start at■: PRINT ■sector■
OF■which is an offset of:■
115      L1=INT(OF/65536):L2=OF-
(L1*65536):OF$=■000000":A$=HEX$(L1):A=LEN(A$):
MID$(OF$,3-A,A)=A$:A$=HEX$(L2):
A=LEN(A$):MID$(OF$,7-A,A)=A$:PRINT:PRINT
TAB(12)■■> $■OF$■ <■■
120 PRINT:PRINT■Configure RGB-DOS now?
■;
125 A$=INKEY$:IF A$=■■ THEN 125 ELSE IF
A$=■Y■ THEN PRINT■Yes!■:GOTO 140
130 PRINT■Nope.■:PRINT:PRINT■Good luck.
I hope this program is of   use. ■
Allen■:END
135 END
140 PRINT:PRINT■Loading `RGB-DOS.BIN'
image. ■;:LOADM■RGB-DOS■
145 PRINT■Patching.■
150 FOR A=0 TO 2:POKE &H3938+A,VAL
(■&H■+MID$(OF$,A*2+1,2)):NEXT
155 PRINT:PRINT■Saving LOADM compatible
`RGB-DOS.BIN'.■
160 SAVEM■RGB-DOS■,&H1FD0,&H3FFF,&H1FD0
165 PRINT:PRINT■Saving EPROM compatible
`EPROM.DOS'.■
170     SAVEM■EPROM.DOS■,&H2000,
&H3FFF,&H2000
175 PRINT:PRINT■Complete.  New RGB-DOS
image is ready.■
180 END
```

This information should give you enough to create an EZ135 disk with both RS-DOS and OS-9 sections. Of course, OS9PART.BAS can be used with any SCSI drive and RGB-DOS as well. Note that, for most values, both decimal and hexidecimal values can be used. Just enter a "$" before a hex value. (For example, at the "total sectors" prompt, typing "$40000" or "262144" will both work.)

To be continued...

In the next issue, I'll discuss using an EZ135 for DECB only and for OS-9 only on a CoCo. There will be a couple programs to make this a simpler task. Of course that means I have two more months to play around with the setup until then! If I discover anything new, I'll be sure and let you know!

OS-9/68K Programming Help

How do you set/get the time???

Ark Systems, USA

QUESTION:

(from Steve Wallace <wallace@titan.com> on the Internet):

I am having a bit of a difficult time with the _os_setime() documentation.

From p 2-450 of Ultra C Library reference manual: error_code _os_ setime(u_int32 time); _time_ specifies the time. The time is stored as the number of seconds since 1 January 1970 Greenwich Mean Time in the form 00hhmmss.

Then, from p 2-451: error_code _os9_setime(u_int32 time, u_int32 date); time is in the format 00hhmmss date is in the format yyyymmdd.

Also note: There are two functions for getting the time called _os_getime() and _os9_getime() where _os9_getime() retrieves a Gregorian time as 00hhmmss and date as yyyymmdd or Julian time as seconds since midnight and date as the Julian day number, whereby _os_getime() returns the number of seconds since midnight of some date (which, after non-trivial digging turns out to be 1 January 1970).

My question, then, is that if I have the time expressed as some number of seconds since some date (for the sake of argument we may assume that it is corrected to 1 Jan 1970), how do I set the time/date? The arguments for _os_setime() don't have enough hours in one byte to support this...00hhmmss only allows for 255 hours, which is only 10 days, 15 hours. So this can't be right...Is this really a typo, and the argument for _os_setime() is really just the number of seconds since some date (1 Jan 1970), and the business about 00hhmmss is a cut and paste error by the tech writer that didn't get caught? Even if it's two bytes, it's still only about 7 yrs 5 mos, so even that isn't right.

Or do I really need to convert the number of seconds that I have to the yyyymmmddd, 00hhmmss format and use _os9_setime()? If that's the case, are there routines that can convert between the two, taking into account things like daylight savings time and leap years and such?

ANSWER:

_os9_getime() and _os9_setime() take two arguments (other than the format specifier and ticks in _os9_getime()), *time of a day* in "00hhmmss" and date in "yyyymmdd." So, hh in time is always between 0 and 23 inclusive. They are different from _os_getime() and _os_setime() which use Unix-style numbers of seconds since a reference time.

Be careful: OS-9000 V1.3 or earlier uses midnight of Jan. 1, 1980 UTCas the reference though OS-9 uses Jan. 1, 1970 UTC. Another glitche is that the manual uses a "byte 3..1" notation to fool you about endian.

Don't be fooled. The actual bye positions agree to natural feelings: say, 00 is always the MSB and ss is always the LSB in "00hhmmss," and they are neither the "first byte" nor "last byte."

Iku
ARK Systems USA
P.O. Box 23, Santa Clara, CA
95052-0023
http://www.arkusa.com
Phone:(408)244-5358
mailto:arkusa@arkusa.com

EDTASM6309 v2.02 - $35.00
Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

CC3FAX - $35.00
Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

HRSDOS - $25.00
Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

DECB SmartWatch Drivers - $20.00
Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

RGBOOST - $15.00
Make the most of your Hitachi 6309 by using it with DECB! Uses 6309 functions for up to 15% gain in speed. Compatible with all programs tested to date! *Only $10 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Please add $4 S&H per order

Northern Xposure

CoCo OS-9:
NitrOS9 - Upgrade OS-9 Level II for the CoCo to a new level of performance! Requires Hitachi 6309. Call or write for upgrade info. $29.95
Shanghai:OS-9 or **Thexder:OS-9**
Send manual or ROM Pak to prove ownership. Transfers and modifies code to run under OS-9! $29.95
Rusty - Launch DECB programs from OS-9! $20.00
SCSISYS 2.2 - SCSI hard drive drivers... fast! Supports 256 or 512 byte sector drives! $25.00
Hitachi 6309 CPU $15.00
SIMM 512K Memory Upgrade - runs cooler!
W/512K - $44.95 w/o memory - $39.95

OS-9/68000:
OSTerm 68K - v2.2terminal program. TTY/ANSI/VT100/KWindows support $50.00

7 Greensboro Cres
Ottawa, ON K1T 1W6
CANADA

All prices in US funds. Check or Money Order only.
Prices include S&H.

for all your CoCo hardware needs, connect with

CoNECT

NEW ADDRESS
1629 South 61st Street
Milwaukee, WI 53214-5055
(pulland@omnifest.uwm.edu)

That thing that Tandy calls a serial port on the CoCo has always been a problem. It was designed with minimal cost in mind, and never upgraded. Even Tandy tried to fix it with their RS-232 Pak, but even it was only half done! Our Fast 232 port uses a 16 byte buffer at alleviate missed characters at any speed and also has ALL RS-232 lines implemented. It is easy to set up with jumpers for different addresses. A daughterboard can be purchased to easily add a second fast serial port! And all this in a cartridge the size of a ROM Pak! 6809 and 6309 OS-9 drivers included. Completely supports up to 57,600 bps, limited support for 115,000 bps.

Fast 232 - $79.95
Daughter Board - $45.00

Check with us for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software!

ADVERTISER'S INDEX

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!
The more people who want the support, the longer it will be here!

EDTASM6309 v2.02 - $35.00
Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

CC3FAX - $35.00
Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

HRSDOS - $25.00
Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

DECB SmartWatch Drivers - $20.00
Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

RGBOOST - $15.00
Make the most of your Hitachi 6309 by using it with DECB! Uses 6309 functions for up to 15% gain in speed. Compatible with all programs tested to date! *Only $10 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Please add $4 S&H per order

Northern Xposure

CoCo OS-9:
NitrOS9 - Upgrade OS-9 Level II for the CoCo to a new level of performance! Requires Hitachi 6309. Call or write for upgrade info. $29.95
Shanghai:OS-9 or **Thexder:OS-9**
Send manual or ROM Pak to prove ownership. Transfers and modifies code to run under OS-9! $29.95
Rusty - Launch DECB programs from OS-9! $20.00
SCSISYS 2.2 - SCSI hard drive drivers... fast! Supports 256 or 512 byte sector drives! $25.00
Hitachi 6309 CPU $15.00
SIMM 512K Memory Upgrade - runs cooler!
W/512K - $44.95 w/o memory - $39.95

OS-9/68000:
OSTerm 68K - v2.2terminal program TTY/ANSI/VT100/KWindows support $50.00

7 Greensboro Cres
Ottawa, ON K1T 1W6
CANADA

All prices in US funds. Check or Money Order only.
Prices include S&H.

for all your CoCo hardware needs, connect with

NEW ADDRESS

CoNECT

1629 South 61st Street
Milwaukee, WI 53214-5055
(pulland@omnifest.uwm.edu)

That thing that Tandy calls a serial port on the CoCo has always been a problem. It was designed with minimal cost in mind, and never upgraded. Even Tandy tried to fix it with their RS-232 Pak, but even it was only half done! Our Fast 232 port uses a 16 byte buffer at alleviate missed characters at any speed and also has ALL RS-232 lines implemented. It is easy to set up with jumpers for different addresses. A daughterboard can be purchased to easily add a second fast serial port! And all this in a cartridge the size of a ROM Pak! 6809 and 6309 OS-9 drivers included. Completely supports up to 57,600 bps, limited support for 115,000 bps.

Fast 232 - $79.95
Daughter Board - $45.00

Check with us for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software!

ADVERTISER'S INDEX

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... "*the world of 68' micros*"!
The more people who want the support, the longer it will be here!

Wittman Computer Products

HARDWARE * SOFTWARE * CONSULTING

We Offer Gift Certificates

Software

Game Pack - Sea Battle, Minefield, KnightsBridge, Othello, & Yahtzee	$30 MM/1 $25 COCO
Variations of Solitaire - Pyramid, Klondike, Spider, Poker, & Canfield	$30 MM/1 $25 COCO
Gold Runner 2000 - KWindows game	$35 MM/1
KChess - GNU Chess with KWindows GUI	$25 MM/1
GNU Chess Source Code Disk	$5 OSK
CirCad - Circuitry CAD program with PostScript output	$80 MM/1
DeskTamer v2.0 - Personal Information Manager	$65 MM/1
LaTerm / LaDial - GUI terminal package with script auto-dialing	$65 MM/1
LaPhone / LaFax - GUI voice, fax & address database manager	$65 MM/1
InfoXPress - auto-dial and download messages	$75 OSK $55COCO/OS9
X-10 Home Controller - Program for monitoring your X-10 system	$40 MM/1
M6809 - CoCo OS-9 emulator	$65 OSK

TVP Point of Sale - Fully integrated multi-user accounting, inventory, ordering, and cash drawer. $CALL

Hardware

WCP306 single board computer with 16 bit PC/AT I/O bus, MC68306 CPU running at 16.67MHz (code compatible with 68000), OS-9 v3.0, MGR GUI, many utilities.
3 Slot board - **$400** 5 Slot board - **$425**

We also carry Point-of-Sale equipment, hard and floppy drives, cases and power supplies, data/fax modems, keyboards, memory, mice, and more!

For more details, write or call for a catalog.

Wittman Computer Products
39 South Lake Avenue
Bergen, NY 14416
716-494-1506

e-mail: ww2150@acspr1.acs.brockport.edu
(acspr "number 1" - lowercase l won't go through!)