OS-9 Newsletter®

Volume V Issue 10 | Bellingham OS-9 Users Forum | October 31, 1994

Special "BLOB-FIX" Issue

THE BLOB IS DEAD *PT2*

I will never forget my first encounter with the BLOB. I had purchased a second hand Maxtor 71MB hard drive which I added to my CoCo XT system as /h1, my second hard drive. The installation itself was quite uneventful. Everything seemed textbook perfect. However, several days later, when I needed to use the floppies, I was quite surprised to find that they no longer functioned. Every attempt at formatting aborted with an error #244. I have a 1987 GIME machine with lots of OS9 patches and I keep it in excellent condition. This problem really had me stumped. It took awhile to trace the problem to the installation of the /h1 descriptor. When it was removed, everything worked fine. Some more time with *ezgen* and *mdir* revealed that the moving of *cc3disk* within os9boot (and consequentally, in memory) determined whether the problem would occur. Hmmmmm.

Soon, I had an entire collection of OS9boots. Some would work fine, some would not format, and others would result in the first letter of each filename to be "lost". One thing seemed to be common among bad OS9boots - the *cc3disk* module was loaded into an ODD module start address offset. Not all odd addresses caused BLOB, however, all BLOB boots seemed to have *cc3disk* at an odd offset.

The next step was to disassemble the *cc3disk* module in order to find out where the exact instruction responsible for BLOB resided. I put together an OS9boot which suffered from BLOB, but worked fine when *cc3disk* was moved forward by 1 byte to an even address. Thus, I could move a NOP instruction through *cc3disk* until I found the offending instruction. The problem was traced to this harmless looking instruction: bita $FF48, which is a check of the FDC status register.

Before I explain how things were going wrong, I ought to explain how floppy I/O is supposed to work on a CoCo. I'll go through a write command. Read commands work similarly, data just flows the opposite direction, of course.

(Continued on page 3)

Blob Symptoms

1. Floppy formats abort with an error #244 (Read error) during the first phase of formating (physical format - before it asks you for the disk name). Note that the system MAY cease to function after this point and all commands yield an error #232. The error #232 in not directly caused by BLOB, but rather is the result of the domino efect.

2. Reads or Writes to floppies may yield an error #244 or more commonly, no error, yet the first byte of each sector is lost. PCDOS will corrupt file xfers under these conditions. Also, dirs may yield filenames with the first letter missing.

3. Floppy boots fail. Yet, when cc3disk is moved around in OS9boot or when another module, especially an od
across cc3disk's position, the boot succeeds.

*** IN THIS ISSUE ***

BLOB ARTICLES by Michael Shell

Q&A

by

Michael Shell

Q: *I use a Disto SCII controller in the buffered mode. So, I can't have BLOB, right?*

A: Wrong. The Disto drivers use a format routine that is susceptible to BLOB

Q: *What does memory surge have to do with CoCo floppy drivers?*

A: Both the Disto and conventional *cc3disk* drivers consume 6.5K of SYSTEM memory space for VERY brief intervals during the first stage of formatting. You'll have trouble catching it in the act with utilities like *smap*, but believe me, it's there. Just try a format with less than 6k of free system space. This could be fixed by blocking interrupts and swapping in a new 8k block into the system space as is done in the SCSI cache of Matt Thompson's SCSI system. Also, it would be nice to have an add on 8-16k buffer for CoCo floppy controllers. This would be especially practical for the SCII which already has all the support logic needed. All it needs is a bigger counter and static RAM chip. For the time being, just curtail multitasking during formats and watch out for low system space conditions.

Q: *Are the Western Digital 2793 controller chips susceptible to the BLOB bug?*

A: unknown. The 27xx series offers many improvements over the 17xx series. Hopefully, they fixed the BLOB problem in the 27xx series.....hopefully.

Q: *What about 17xx compatible clone chips made by companies other than Western Digital?*

A This is also unknown. When in doubt, patch.

EDITOR'S NOTE: My 17xx compatible clone chips are not susceptible to the BLOB. When I have a BLOB problem I just switch to my J&M controller with the 17xx clone and boot up without a hitch.

Q: *I have modified my controller and driver to support High Density floppies and/or give me other custom features. How do I get the needed patches for my setup?*

A: The technical information I have included in these articles should provide you with enough information for you to modify your custom driver using a disassembler, a text editor, and *asm* (the level I assembler). In other words, I give you the source code changes needed.

Q: *What about the GIME S0-S2 decoding BLOB hardware fix? How does this fit into everything?*

(Continued on page 7)

OS-9 Newsletter

Editor: Rodger Alexander

OS-9 Newsletter **is published monthly by the Bellingham OS-9 Users Forum and is protected under United States Copyright Laws. No material may be reproduced or copied in whole or in part without the expressed written permission of the Bellingham OS-9 Users Forum, 3404 Illinois Lane, Bellingham WA 98226**

Submissions **are welcomed in any format and can be mailed to the above address or sent via electronic mail to the editor: Rodger Alexander, on Delphi (UserID: SALZARD) or Internet (ralexander@nikita.bham.wednet.edu). Unfortunately, we do not have funds to reimburse authors of selected articles. However, a complimentary copy of the *OS-9 Newsletter* containing your article will be mailed to you, PLUS the satisfaction that you will have the admiration and appreciation of all of our readers.**

The *Bellingham OS-9 Users Forum* is a hobbyist club, organized for the purpose of providing information, services, products and events that support the OS-9 operating system for 6809/68xxx based computers. Our efforts are not intended to earn or generate any profit for the club or any of it's members.

SPECIAL NOTICE

The December '94 issue of the *OS-9 Newsletter* will be the last publication. Due to declining subscriptions and an increased demand of my time on other computer systems I find it necessary to bring the *OS-9 Newsletter* to an end.

Renewals and new subscriptions will not be honored. Rebates will be paid to subscribers who's subscriptions go beyond the December '94 issue.

(Continued from page 1)

How CC3disk is supposed to write to a floppy

1. The drive motors are allowed to come up to speed, if needed. CC3disk has prepared a buffer in the system space which contains ALL of the data bytes which must be sent to the FloppyDiskController. This buffer is 256 bytes long in the case of a write sector command and 6.5k in the case of a write track command (format). The R/W head of the floppy is already positioned over the appropriate track.
2. *CC3disk* masks all CPU interrupts so that OS9 cannnot disturb all the critical timing loops. This is why we lose characters from the keyboard if we type during floppy I/O.
3. *CC3disk* issues the write commanad to the FDC and then waits the required 64 usec to give the FDC time to update its status register.

POP QUIZ! Here's the actual delay code used by *cc3disk*. How long does it take to return to the caller (i.e bsr delay)?

```
delay  lbsrnext1
next1  lbsrnext2
next2  lbsrnext3
next3  rts
```

Answer: About 58us. This should be adequate for the FDC and is not the cause of BLOB. However, you guys who are accelerating the CoCo with crystal hacks could cause a problem here. Note that this routine is not patcher friendly with regard to changing the delay time. Future patches could rewrite this routine to make the delay "programable". This code really flies with a 6309 in native mode. *My native mode patches slow things here to keep the needed delay.*

4. The FDC chip has two pins with signals of interest:
   a: DRQ -> this is the data request line. When DRQ is high, the FDC is requesting that a data byte be read or written to it. When the CoCo controller is in the halt mode, the CPU is halted when DRQ is low (no data xfer needed) and the CPU is running when DRQ is high (data xfer needed). The DRQ line can be read from bit #1 of the FDC status register ($FF48).
   b: INTRQ -> When the FDC has finished a command, with or without an error, this line goes high. If the driver has enabled the NMI circuits of the controller, a high INTRQ will trigger a NMI (Non Maskable Interrupt). Thus, when the FDC completes a command, the CPU will receive a NMI. Note that NMIs cannot be masked by the cc register of the 6809. *CC3disk* installs a NMI routine whose sole purpose is to check to see if the FDC encountered an error during the last instruction and if so, to report this to OS9. *CC3disk* now enables NMIs.
5. *CC3disk* loads a value into the b register which, when sent to $FF40, will enable halts without disturbing the other settings.
6. *CC3disk* now waits in a "time out" loop and uses the instruction bita $FF48 to check the FDC status register to see when DRQ becomes set and the transfer is ready to begin. If a few seconds pass with no DRQ because someone forgot to close the drive door (the FDC uses index pulse counts to determine when a transfer is to begin. no disk = no pulses = FDC will wait forever), *cc3disk* will timeout and abort with a device not ready error. Note that this behavior is contrary to the 1773 docs. Western Digital claims that the 1773 will assert DRQ on the FIRST byte of the xfer regardless if index pulses are present or not (provided a head load delay is not required). Microware found out differently and implemented this timeout code.
7. If the hardware is working correctly, the FDC will set the DRQ line and *cc3disk* will put the controller into the halt mode. *CC3disk* will then put the CPU into a very tight infinite loop whose sole purpose is to read data from the buffer and write it to the FDC. The halt line is used to let the CPU write data only when it is needed. The CPU must be able to move about 250k bits/sec for double density operation and 500k bits/sec for high density (or 8" drives). The loop easily achieves this requirement on a 1.78 MHZ CoCo.
8. When the FDC command is complete, a NMI interrupt is sent to the CPU.
   The CPU is yanked out of the infinite loop and enters the NMI routine. The occurance of the NMI automatically takes the controller out of the halt mode. This is done in hardware.
9. The NMI routine pulls the junk that was put on the stack during the NMI. This way, a RTS instruction will return the CPU to OS9 or to another place in cc3disk depending on how many bsr levels deep the code is. In any event, the important thing here is that a RTS will NOT return the CPU to the infinte loop.
10. If the data lost bit was set in the FDC (because the CPU did not respond in time to a DRQ), the NMI routine will return a #244 error to OS9 (even if a write operation generated the error).
11. The NMI routine then checks the FDC status register for any other errors and if present, returns the appropriate error code to OS9.

Inserting test code into *cc3disk* revealed that the BLOB problem was occurring in step 6 with the bita $FF48 instruction. When this instruction was located on certain odd addresses (odd *cc3disk* start = odd bita $FF48), the CPU would never see the DRQ for the first byte. The CPU would keep checking and never see the DRQ until 150 us into the write command when the FDC would error out with a NMI and the lost data bit set. The NMI would then see the error and report a #244 error to OS9. If this problem occurred in the boot module, the user would see "BOOT FAILED".

In effect, the 1773 was saying, "What happened!? I asked for the first byte and you never sent it to me in time!". The CPU replied, "WHAT!!!??? I sat there checking you and you never asked for a data byte!". It's like two people transferring a vase.

(Continued on page 4)

(Continued from page 3)

One tries to hand it to the other, but lets it go before the other has a chance to get a grip on it. The result is a dropped vase.

Now, the DRQ line can be cleared by an access to the DATA register ($FF4A), but it should NOT be affected by reads to the FDC STATUS register ($FF48). So, it appears as though the 1773 has an internal hardware bug. When the CoCo checks the FDC status register, the FDC sometimes clears the DRQ immediately even though data transfer has not occurred. The fact that the status register only seems to be misread immediately after a data transfer command has been issued futher implicates the 1773. If the CoCo had an address decoding problem, we would have problems with status register reads at other times.

The $10,000 question becomes: Why does this not bother the CoCo when the bita $FF48 is located on an even address? My theory is that the 6809 (and 6309) exhibit slight internal timing differences depending on where in memory the code is located. If everybody honors the CPU timing specs, then there is no problem. However, if these specs are violated, as the 1773 appears to do by altering a data bit in the middle of a read operation, then weird and erratic results can occur. In other words, when bita $FF48 is located on an even address the CPU can "see" the DRQ before the 1773 has time to rip it away.

It is interesting to note that RSDOS uses code that can cause the BLOB problem. RSDOS always seems to work correctly because:
a. The .89Mhz speed is less likely to cause the 1773 to malfunction (I have stopped some BLOBs in OS9 by bringing the CPU out of the 1.78Mhz mode.).
b. RSDOS always loads into the same memory area, so the offending code is always at a "safe" location.

Now that we understand the problem (and hope that it is right!), how do we fix it? The answer is simple: Once any command is issued to the 1773 that will result in a data transfer, DO NOT poll the status register until after the command is completed (in the NMI routine). Instead, let the 1773 control the CPU with the hardware DRQ<-->HALT link thoughout the ENTIRE data transfer, ESPECIALLY the first byte.

Before and After Source Code for the affected sections.

You guys with custom drivers may need this stuff. The read sections apply to the boot module. The write code applies to the format routines of the Disto drivers. Both apply to the stock *cc3disk*, from which this code is taken (Microware won't mind. I hope! Educational use!). The comments are added by me.

Original read section:

```
* L010c    bita     $ff48     is FDC ready for first byte? Cause BLOB here.
*          bne      readloop  if DRQ, start reading
*          leay     -$01,y    dec the timeout value
*          bne      L010c     cont checking until timeout
*          lda      $00a9,u   get drive select data for this drive
*          ora      #$08      make sure the motor stays on till it times out
*          sta      $ff40     turn off NMIs, DDEN, etc
*          puls     cc,y      restore interrupts and y reg
*          lbra     L031a     back to OS9 with read error
* readloop lda      $ff4b     get a data byte
*          sta      ,x+       store it in the buffer, point to next cell
*          stb      $ff40     enable halt mode
*          bra      readloop  continue till NMI
```

The new read code:

```
L010c     stb      $ff40     enable HALT mode
          nop                allow two op code fetchs for HALT
          nop                to take effect
          bra      readloop  enter infinite loop
readloop  lda      $ff4b     get data from fdc
          sta      ,x+       store byte in buffer
          nop                one more op code fetch for HALT
          bra      readloop  repeat till NMI occurs
```

Original write section:

```
* L017d    bita     $ff48     is FDC ready for first byte? Cause BLOB here.
*          bne      wrtloop   if DRQ, start writing
*          leay     -$01,y    dec the timeout value
*          bne      L017d     cont checking until timeout
```

(Continued on page 5)

(Continued from page 4)

```
*           lda     $00a9,u   get drive select data for this drive
*           ora     #$08      make sure the motor stays on till it times out

*           sta     $ff40     turn off NMIs, DDEN, etc
*           puls    cc,y      restore interrupts and y reg
*           lbra    L02e9     find out what happened, then back to OS9
* wrtloop   lda     ,x+       get  byte, advance buffer counter
*           sta     $ff4b     write byte to FDC
*           stb     $ff40     enable halt mode
*           bra     wrtloop   cont writing till NMI
```

New write code:

```
L017d       stb     $ff40     enable HALT!
            bra     wrtloop   enter infinte loop
wrtloop     nop               two opcodes before fdc write
            lda     ,x+       get data byte
            sta     $ff4b     write byte to fdc
            bra     wrtloop   repeat till NMI
```

Original NMI routine:

```
* NMIRQ     leas    12,s      do not return to infinite loop
*           puls    y,cc      pull y off stack, restore interrupts
*           ldb     $ff48     get status
*           bitb    #$04      Do have lost byte error?
*           lbne    L031a     if so, return read error to OS9
*           lbra    L02ec     check for more errors, back to OS9
```

New NMI routine code:

```
NMIRQ       leas    12,s      do not return to infinite loop
            puls    y,cc      pull y off stack, restore interrupts
            ldb     $ff48     get status
            bitb    #$04      Do have lost byte error?
            lbeq    L02ec     branch if no lost byte error, check more
            comb              set error flag cc
            ldb     #$fa      get device busy error
            rts               go to os9 with error
```

Note that in the read and write sections, the first branch is not needed at all. I could have just let the CPU "fall" into the loop without a branch. Having a branch serves two purposes: **1.** it gives HALT time to have an effect in case the 1773 is "sluggish" on the first byte. **2.** It allows for a place to easily insert patch and/or test code.

I did not implement the change in error reporting in the NMI code of the Disto drivers. Also, the 6309 native mode patches change the leas 12,s into a leas 14,s to correct for the two extra bytes pushed onto the stack. This is one reason why unmodified Disto drivers crash during formats under native 6309 operation.

Notice that the NOP instruction that I used in the loops should also not be needed. I put it in there just in case the 1773 violates its DRQ reset timing specs. Since the CPU will not halt until after the end of its current instruction, it's a good idea to have two opcode fetches between the action the causes a halt and the point where the CPU must actually be stopped. The new loops as given deliver 890K bit/sec performance. This should be enough for even you high density folks. It is interesting to note that without the NOP, a 1.78 MHZ CoCo will max out at over 1 Mbit/sec transfer speed. We could get almost twice that if we used a special controller with the data register mapped into two consecutive addresses and used a ldd and std like the Disto buffered mode. It would have to allow two byte transfers between halts. Can you say, "2.88 MB floppies on a CoCo."? However, if somebody is planning this, I suggest at least a 32k buffer to hold an entire track of a 2.88 MB disk and eliminating the need for this halt stuff altogether.

Well, I certainly hope that all of this effort finally puts a stake through the heart of a problem that has plagued many CoCo owners for so long. Lastly, I would like to thank all of you who helped to bring forth all the hardware and software for the CoCo. Without the tools you folks provided, I could not have made this patch set.

==Michael Shell==

BLOBSTOP.AR (Blob fix patches) is available on Internet from *FTP.Chestnut.cs.wisc.edu* or from us (*OS-9 Users Foru; 3404 Illinois Lane; Bellingham, WA 98226*) Send a disk and return postage with your name and address or $1.

Miscellaneous Tips

Here's a bunch of odd ball items that you might find helpful:

1. In case anybody is wondering who turns up the speed, one of the last things the boot module does is to put the CoCo into the 2 MHz mode.
2. If you have test code within a device driver and you want it to send you a "signal", try poking values into the border register at $FF9A. OS9 won't mind a bit.
3. If your disassembler is the type that misses IRQ routines and prints them as a bunch of *fcbs*, try this trick:
   a. Disassemble the code.
   b. Put in long branch subroutines (lbsr) to the beginning of each IRQ routine that the disassembler missed. The best place to do this is in the last instructions that the disassembler picked up:

```
decb
bne L004b
clrb
rts
```

end of code here

Becomes:

```
    decb
    bne L004b       note that the lbsrs are in a "path
    of
    clrb            execution"
    lbsr D0100      branch to IRQ#1 that disa missed
    lbsr D02f0      branch to NMI that disa missed
    lbsr D01a0      branch to IRQ#2 that disa missed
    rts
```

end of code

   c. Assemble the code with *asm*.
   d. Disassemble the resulting object code.
   e. Remove the statements that you added. Remember that the offset labels to code after your inserted code will no longer be valid as offsets from the start of the module. If you have no labels after the place where your lbsrs were, then this will not be a problem.
   f. Presto, fully disassembled code!

3. The magnetic zones on a disk tend to repel or attract each other depending on their respective polarities, just like little magnets. Thus, the ones and zeros on a disk tend to "move" a bit depending on the adjacent bits. If pronounced enough, this effect can lead to read errors. This is particularly true on the inner (high #) tracks where the bits are closer together. Write precompensation is a process in which the position of the bits is shifted during writes to compensate for the natural "wandering" of the bits. As Kevin Darling mentioned in his book "Inside OS9 Level II" (p 3-5-3), *cc3disk* never activates the 1773's built in write precomp circuits. However, I found unused, leftover code in cc3disk that is designed to activate write precomp. Talk about skeletons in the closet! Perhaps future patches could re-enable precomp on the inner tracks. A small improvement in read reliability could be realized especially by those using older drives.

4. If you are experiencing I/O errors with a MPI, especially with the buffered mode of the SCII, try changing IC1 on the older MPI or IC2 on the newer MPI. These chips handle the E clock and some other signals. I have seen a case in which a bad chip caused some of the address lines to bounce with transitions of the E clock -> bad news!

5. The owner's manual of my Disto SCII has a mistake in table 2. The register addresses are listed backwards. Here's the corrections:

Table 2 - SCII registers

Location		Description
Hex	Dec	
FF77	65399	FF76 mirror
FF76	65398	write: D0 = 0 FDC write operation *1 = 1 FDC read operation *1 D1 = 0 normal mode = 1 buffered mode D2 = 0 normal NMI = 1 masked NMI D3 = 0 no FIRQ (masked) = 1 FIRQ enabled read: D7 = FDC INTRQ status (inverted)
FF75	65397	FF74 mirror
FF74	65396	Read/Write buffer *2

*1: In the buffered mode.
*2: Any write to $FF76 or $FF77 clears the buffer counter.

Note that the data buffer mirror allows std and ldd for pseudo 16 bit transfers.

Also, there are a few corrections needed to the SCII schematic set:

Version 1.3:

   a. There should be an inverter in series with the line that leads to the counter (between pin 18 of U6 and pin 1 of U7). This is the small "hack" that you see on the "wrong" side of the PC board near the CoCo connector.
   b. The address line pins of U6 should read: 1,3,5,7,2,4,6,8 not 8,7,6,5,4,3,2,1 as given.

Version 1.4:

   c. Pin 1 of U8 should be shown connected to GND not VCC.

On both versions:

   d. Pin 10 of U8 connects to pin 12 of U13, NOT to pin 13 of U13.

(Continued on page 7)

(Continued from page 6)

6. Concerning GIME S0-S2 hacks: Four signals are produced by IC9 on the CoCo3 motherboard. They are SCS (which helps decode I/O devices), CTS (which turns on the cartridge ROM), ROM (which turns on the motherboard ROM), and the signal which decodes the PIAs which I'll call PIA. These signals belong to two different groups.

a. Address Decoding Signals (ADS). SCS and PIA belong to this group. These signals are used in conjunction with the address lines to decode an address. The E clock is then used to actually turn on the device.
b. Chip Select Signals (CSS). CTS and ROM belong to this group. When these signals are active, the devices turn on regardless of the E clock.

The problem many hackers see with all this is that CTS and ROM do not APPEAR to be gated with E (unless the GIMEs do this internally). So, a ROM device could start outputting junk onto the data bus without checking E. Since ROMs tend to be a bit slow, this could cause a collision on the data bus. The result is a BLOB causer.

As a solution, many hackers rigged U9 so that ALL FOUR signals were gated with the E clock. This is fine and well for CTS and ROM, but I don't think this is a good idea for SCS and PIA. The ADS signals should be stable BEFORE E becomes high in order to give the address decoding logic enough time to stabilize prior to E becoming high. If SCS changes with E, a logic race condition could occur in some CoCo accessories. This could cause I/O glitches.

So, I suggest that any E gating hacks should be confined to CTS and ROM. A 74LS157 (or 74ACT157 if you want the best) should do the job nicely (E clk to A/B select; A inputs go to VCC; B inputs go to CTS and ROM; outputs are new, gated CTS and ROM). Since CTS is hardly ever used under OS9, but ROM is used for every interrupt, you can try a simpler hack that affects the ROM on the motherboard only. Using an inverter, make an inverted E clock and then route this signal to pin 22 (OE, active low) of the ROM. Be sure and remember to cut the trace that currently grounds pin 22. As a final note, all of this address decoding appears to have been done correctly with the SAM chip in the CoCo2. However, without knowing how the GIME generates S0-S2, we can't be sure that it is done correctly in the CoCo3. Note that the GIME probably does SOME additional processing of these signals. For example, R/W is probably taken into account to prevent the CPU from trying to write to a ROM and thereby creating the mother of all bus collisions. If the GIME does indeed consider E and R/W properly in developing S0-S2, then no hack should be needed at all. Gating SCS with E, as most "improper" hacks do, delays the enable of the 1773 chip by a few nanoseconds. This could affect the 1773 BLOB problem. Thus, many S0-S2 hacks may appear to fix GIME problems, when in fact, the 1773 was the problem all along. Time spent with a logic analyzer could settle this issue once and for all.

==Michael Shell==

(Continued from page 2)

A: I am not convinced that this was ever a problem. Instead, I suspect that the 1773 problem is responsible for the vast majority of BLOB related phenomena. You can try jumping pin 6 of IC9 to the E clock (the pin of R9 facing away from the GIME is a good source of E) and see if your problem goes away.

Check out the "Correct All FIX" in the July'93 and October'93 issues of the *OS-9 Newsletter* for step by step instructions and digrams. *Ed.*

Q: *BLOB has something to do with keeping modules in the "proper" order and/or trying to keep cc3disk in the same 8K block as RBF, right?*

A: Wrong. As is have explained elsewhere in the *OS-9 Newsletter*, BLOB has to do with the absolute position of cc3disk in memory not it's relative position to other modules.

NEW OS-9 COMPUTER

Wittman Computer Products is proud to announce our upcoming release of a new OS-9 based computer. Running OS-9 V3.0, MGR a graphical environment interface, and utilize PC cards in its built in slots. The product will be known as a WCP306 computer. It is based on a Motorola 68306 chip

Our target date to accept and fulfill orders will be the middle to end of December 1994. We are accepting orders now, with no money down. The pricing of the machine will be:

$400.00 for the base package below.

Single Board Computer with 16 bit PCAT I/O Bus
WCP306

- MC68306 CPU at 16.67 MHz
  - code compatible with 68000
  - 2.4 MIPS
  - 68681 DUART - 2 serial ports
  - 16 Bit Timer/Counter
- 8 to 16 Bits Parallel I/O
- DRAM controller
- 0.5MB, 2MB, or 16MB DRAM (4 SIMM sockets)
- IDE Hard Disk Interface (2 drives max)
- 1.44MB Floppy Interface (2 drives max)
- 2 16 byte FIFO serial ports (up to 115K baud)
- Bi-directional parallel port
- On board RS232 buffers
- Battery Backed Real Time Clock
- AT Keyboard Interface & Standard AT power connector
- 5 AT (16 bit) I/O slots
- Baby AT Size Footprint
- Basic (resembles Microsoft Basic)
- MGR - graphical windowing environment, full documentation!
- "Personal" OSK V3.0 (Industrial with RBF)
  - MW Managers: SCF, RBF, PipeMan, RamDisk
  - MW Utilities: date, deiniz, devs, dump, echo, and etc..
  - PD (at least): dir/ls, cp, mv, arc, ar, tar, cpio, ...
  - Development: At least an absolute (non-linking) assembler, and perhaps an r68/l68 clone.
- Display drivers: Tseng 4K, generic inexpensive VGA
- SCSI card support: Future Domain 1680 & Adaptec AAH 15XX
- UUCP package from Bob Billson

WCP306 board only $400.00

- includes OS-9 v3.0 & MGR
- Desktop case w/ 200w ps $110.00
- Midsize Tower case w/ 200w ps $110.00
- AT keyboard $ 50.00
- VGA card $ TBA
- SVGA card $ TBA
- Monochrome card $ TBA
- Teac floppy drives 3.5" $ 50.00
- or 5.25" $ 60.00
- Generic serial mouse $ 20.00
- SCSI and IDE hard drives are around $1.00 per Megabyte, call for current pricing and availability.

=William L. Wittman, Jr. - Owner=

Atlanta CoCoFEST

Some recollections from the recent Atlanta CoCoFEST

Vendors:

Adventure Survivors had a few new adventure games for the CoCo. John Strong had a cool late version of his MM/1 paint program. John's paint program also used a hardware drawing pad (a little smaller than Tandy's old x-pad).

Digital Frontier Productions (formally Hyper Tech but without Mike Haaland) had a cool scrolling platform action game for the MM/1 called Gold Runner 2000 - complete with digitized sound. I think Gold Runner 2000 is the first MM/1 software product which shipped on 3 high density disks. The program has a sound-track that runs while the game plays, as well as special effects. As everyone else has said it's worth the money. (Note, anyone who bought gold runner on Saturday may wish to check with Eric, since he made a fix to something Sunday afternoon).

Northern Xposure was showing off a Beta test version of OSTerm 2.3, which implements a much better vt100 emulation (which I think includes support for the IBM-PC character set), as well as a better ANSI emulation which supports color, etc. So now you can call up your favorite MS-DOS BBS, and see all the color & graphics.

Bill Wittman was showing off DeskTamer for the MM/1. A calendar/to-do list/phone-message program. A pretty sharp looking program which would show a calendar, and a graphic representation of your day (like if you had a meeting from 8am-noon there was a little bar graph from 8am-noon.)

Farna Systems table had a sign saying they are now selling complete CoCo w/hard drive systems packaged in PC cases for round $500. I'm not sure on all the details...but certainly a good idea for someone looking

for that sort of thing.

There was also a demo of the **CoCo 3** emulator running on a color notebook...it's pretty spooky to walk by a table, and see a little notebook computer with a GREEN 32 column display with the "Disk Basic 1.2" message on it.

= Erik Seielstad=
erik@acs.brockport.edu

The Lectures

Saturday:
12:00-12:55 "The Future of OS-9" by Ed Gresick
He's contributed a lot to OS-9's having a future, that's for sure.

3:00- 3:55 "Open OS-9 Forum" by Kevin Darling
Those who attended were updated on Kevin's current career in writing software for video gaming systems. Look for an article

Sunday:
12:00-12:55 "Inside CD-i" by Boisy Pitre
A fascinating demo of the capabilities of the CD-i machines. Boisy also demonstrated his bootable CD-i disk which gives shell access on a CD-i machine via a terminal. Boy would I have liked to see this, though I suppose an even better version will come to Chicago. Wonder if Frank Hogg showed up at all, doesn't look like it. I remember Frank giving the first lecture (at 1st annual Chicago Fest?) about the notion of using a CD-i machine as an OSK home computer.

=Mike Knudsen=

A CoCo 3 Emulator??????

YES, It really was a CoCo 3 emulator! Actually, it was the CoCo 3 emulator prototype (some work is still needed).

=Jeff Vavasour=

Miscellaneous Tidbits

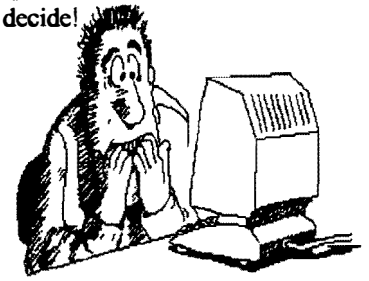
The real name of the Bill Gates is William Henry Gates III. Nowadays he is known as Bill Gates (III), where "III" means the order>of third (3rd.)

By converting the letters of his current name to the ASCII-values and adding his (III), you get the following:

```
B       66
I       73
L       76
L       76
G       71
A       65
T       84
E       69
S       83
+       3
--------------------
        666!
```

Some might ask, "How did Bill Gates get so powerful?" Coincidence? Or just the beginning of mankind's ultimate and total enslavement??? YOU decide!

Signed: Fearful



Club Activities Report

Bellingham OS9 Users Group - Longview/Kelso CoCo Club
Mt. Rainier CoCo Club - Port O'CoCo Club - Seattle 68xxx Mug

Port O'CoCo

At the morning general meeting of the Kitsap Computing Seniors Club, there was a record turn out. Although there were still chairs available, 94 people signed in. And this was for a meeting that had no announced speaker. The group didn't just eat and visit though, there was a lively Q&A session with questions from the simplest to the most complex. In a group this large the magic was that someone had the answer or the name and phone of someone who did.

I was lucky enough to win one of the drawing prices at the end of the meeting. I won the New Riders' Official Internet Yellow Pages. It contains over 10,000 entries from around the world—and beyond with topics I didn't even know existed. It's fun to read even though I haven't figured out how to get on the Internet yet.

The evening meeting was the other extreme in attendance. Terry Laraway is away, away in San Diego on a work assignment. Tom Brooks is on the night shift. Les Bulyar is under the weather with the latest bug. Buzz Jones is consumed by serious personal matters. And Gene Elliott is putting in 12 hours most days of the week. What chance did our CoCo meeting have this month? Almost none.

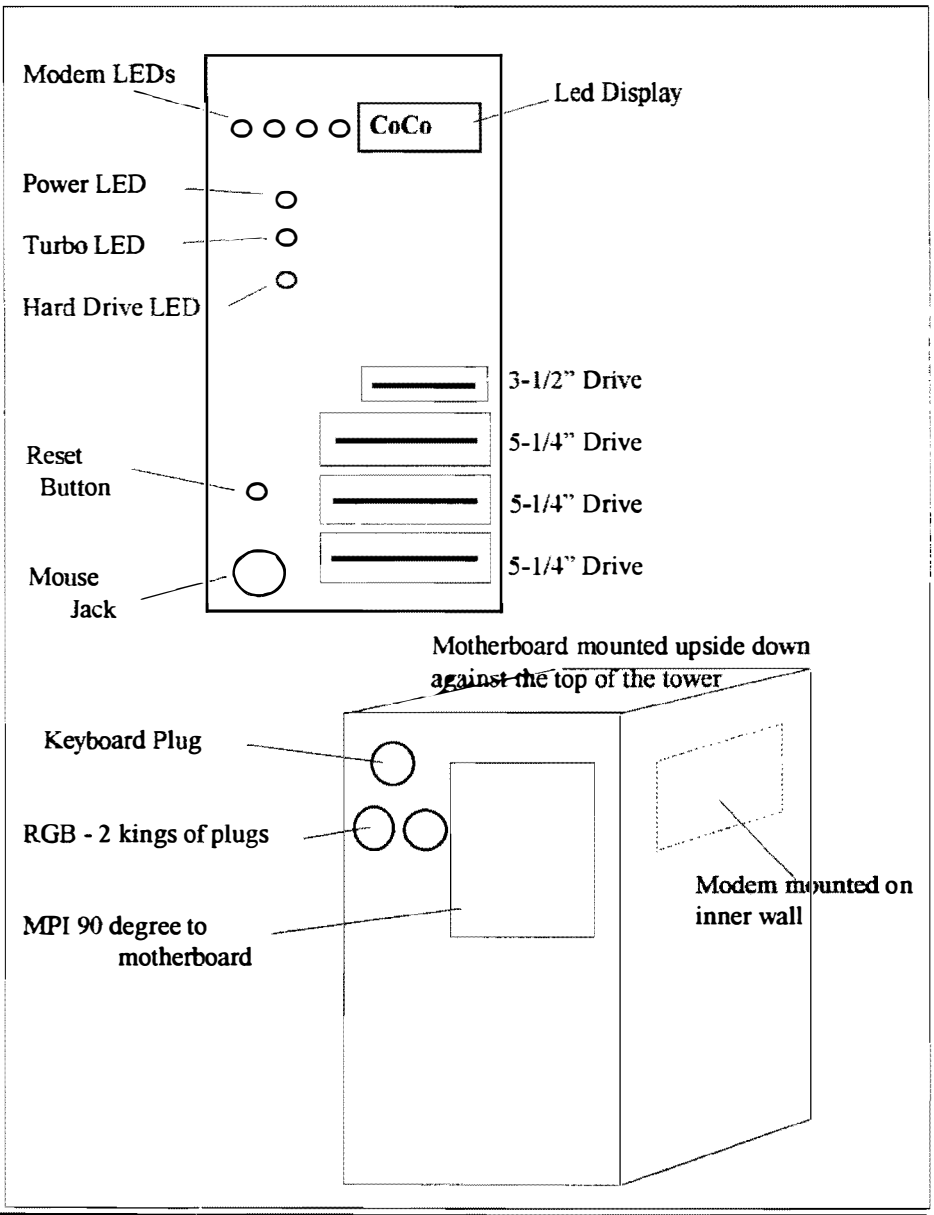
When Donald Zimmerman got to the meeting there was a note on the wall that Gene might be late or a no-show because of work. But he did make in a few minutes late. And that was the end of the "stampede" for the meeting.

The two of us talked about the process on the club's tower. First there is the placement of power supply. Looking from the back, the power supply block has been turned sideways to give greater depth room for the long and somewhat wide boards used in the Color Computer. Also on the back, Gene has replaced the slots with a solid plate and then installed the various plugs from the back on the CoCo. See diagram #1.

The front of the tower is more or less normal except for the LED area. It gives the name, not the number. "COCO" lights up as we power up. What could be better! See diagram #2.

Our next meeting is November 21st. We hope to see more of the gang then.

==Donald Zimmerman==



Seattle 68xxxMUG

The October meeting featured a demonstration by **Bob Bielka** of *Rick's Computer Enterprises* ***"CoCo Users Database"***. Most club members have sent in their registration forms so that they would be included in this database of CoCo users from all over the world. The RSDOS database software put together by "Rick's" was very professional and featured a 127 character screen that is not fully displayed on the screen. The user simply presses either the left or right arrow keys to scroll from the right side of the 127 character display or the left side. The left side of the display listed user names and could be sorted by any field selected. The right side of the screen presented a detailed information form on the individual highlighted from the left side of the screen.

A bonus to the database was the "flippy" side of the disk that contained CoCo text or graphic files of "Vendors". The list of vendors was much larger then I expected, but then I realized that vendors ranged from individual suppliers and supporters to commercial retailers.

The database will be updated three times a year and a fee of $12 gets you all three updates. The address for registering you name on the database or purchasing the database disk (3 updates) is: PO Box 276, Liberty KY. 42539.

Donald Zongker brought his CoCo in a tower to the meeting for a boot upgrade. Donald wanted to be able to read and write to PC (MS-DOS) formatted disk. The stock version of the *CC3Disk* module uses only 256 byte read/write buffer and the PC format uses a 512 byte buffer. Donald did have D.P. Johnson's *Sdisk3* driver but was concerned about it not being compatible to 6309 native mode upgrades.

Rodger Alexander used Burke & Burke's *EZGen* to "update" Donald's original *CC3Disk* version 1 with version 11. Rodger also spoke briefly about the new Blobstop_10 archive that supposedly includes versions 12b & 12c of *CC3Disk*. These newer versions include patches that eliminated the BLOB problems on CoCo-3's using the standard 1773 Floppy Disk Controller Chip.

Jesse Obereuter helped out by replacing the original *Shell* in the CMDS directory with *Shell Plus*. This was more difficult then expected because of the larger size of *Shell Plus*. This resulted in the loss of several files that were merged in with the original shell. By creating a ***UTILS*** file containing the missing utility modules, that were part of the original *shell* file, and rewriting the *STARTUP* file to instruct OS-9 to load the *UTILS* file, we were finally able to get Donald's computer to boot up properly and function as it did before we started modifying his system disk.

Scott Honaker brought a Battery Backup Power Supply that provides power to your computer for 20 minutes after you loose your standard household electricity. It looked very sophisticated and Scott had written his own controller program for the power supply. Unfortunately, Scott had to leave at 9:30 so we did not get to see his power supply work.

The meeting adjourned at 9:30 and we got back to Bellingham before midnight.

==Barbara Alexander==

TERMINATION NOTICE

The ***OS-9 Newsletter*** will publish it's last issue at the end of this year, 1994. Renewal will only be honored for the remaining two issues. Customers who have already renewed their subscription into the next year will be reimbursed for the balance of their subscription or may choose to trade the balance of their subscription for OS-9 products available from the ***OS-9 Users Forum***. Direct your questions to *Rodger Alexander, OS-9 Newsletter, 3404 Illinois Lane, Bellingham, WA.*

Washington State BBS List

COLUMBIA HTS. BBS
-- Longview/Kelso --
RiBBS (FidoNET)
(206) 425-5804

DATA WAREHOUSE BBS
-- Spokane --
RiBBS (FidoNET)
(509) 325-6787

BARBEQUED RIBBS
-- Bellingham --
PC-Board (PC-Net) - CoCo Conference #5
(206) 676-5787

PERMANENT CREW REST
-- Tacoma --
RiBBS (FidoNET)
(206) 472-6805

ULTIMATE EXPERIENCE BBS
-- Anacortes --
RiBBS (MaxNET)
(206) 299-0491

OS-9 Newsletter
3404 Illinois Lane
Bellingham, WA 98226-4238