CoCo~123

NOVEMBER 2002!

Glenside Color Computer Club — Carpentersville, Illinois
Volume 22, Number 3 — November 2002

CoCo ~123 HighLights

TOPIC — *PAGE*

Articles

What is DLOAD/DLOADM, Anyway?
12 years and we still haven't learned !!!
BEWARE of THE BLOB
BLOBSTOP
Thoughts on Disk BASIC and Hard Drives
CoCo OS-9 Source at Sourceforge.net
DOCTOR! DOCTOR!
EOF

G.C.C.C. Inc. OFFICERS

Here is the list of 2002-2003 club officers and how to contact them.
If you have questions about our club, call one of the officers for the answers.

POSITION	NAME	PHONE	PRIMARY FUNCTION
President:	Howard Luckey	708-748-5320	The buck stops here
V. Pres.:	Justin Wagner	630-393-7072	Co-ordination
V. Pres.:	Scott Montgomery	773-282-6044	Navigation
V. Pres.:	Brian Goers	708-754-4921	Special Events
Treasurer:	George Schneeweiss	815-832-5571	Dues and Purchasing
Secretary:	Tony Podraza	847-428-3576	Recording/reporting


Howard Luckey


Justin Wagner


Scott Montgomery


George Schneeweiss


Brian Goers


Tony Podraza

CoCo ~ 123 INFORMATION

The CoCo ~123 is the newsletter of the Glenside Color Computer Club. Your annual contribution of $15.00 keeps our club going. Send your check to the Glenside Treasurer:

George L Schneeweiss
13450 N 2700 E Road
Forrest IL 61741-9629

Our treasury provides newsletters, local meeting room and good times with fellow CoCo users at our annual Chicago CoCoFEST.

CoCo~123 CONTRIBUTIONS

If you have any suggestions for the newsletter or would like to submit an article, please contact the CoCo ~123 Editor:

Howard Luckey
4 Gibson Rd
Park Forest IL 60466-1723

CONTRIBUTORS TO THIS ISSUE

George Schneeweiss	**Allen Huffman**
Howard Luckey	**Bob Swoger**
Neil Morrison	**Marty Goodman**
Boisy G. Pitre	**Gene Heskett**
Michael Shell	**Tony Podraza**

G. C. C. C. MEETINGS

The Glenside Color Computer Club meets the second Thursday of each month at the Schaumburg Heights Public Library at 7:30 PM. See our WWW Glenside Homepage at:

http://members.aol.com/clubbbs/glenside

if you need a map. A social get-together always occurs at the nearby Sante's Restaurant.

FROM THE PRESIDENT'S DISK

This issue's disk was held to the refrigerator with a magnet....oops.

Howard Luckey, President
Glenside Color Computer Club

The V-P's Platen

Sorry, the V-P's printer broke. There is no "Platen" article for this issue.

Glenside Color Computer Club Has A Web Page!

The Glenside Color Computer Club has a web site. Look for us at:
`http://members.aol.com/clubbbs/glenside`

Bob Swoger, Webmaster
Glenside Color Computer Club, Inc.

TREASURY NOTES

As of September 12, 2002, the balance in our checking account is in excess of $4100.00. We can we afford to have another CoCoFEST!

George Schneeweiss, Treasurer
Glenside Color Computer Club

THE SECRETARY'S NOTEBOOK

July 11, 2002

Howard called the meeting to order at 7:57 PM at the Schaumburg Township District Library. Present were Bob Swoger, Rich Ekstrom (Surprise visitor) Howard Luckey, Brian Goers and Tony Podraza.

Bob handed out calendars.
Old business:
IDE boards: Roy R justice bought a board. Recently - past month.

List of "shipped to" boards was received from Carl Boll. (We have data for the IDE project that dates from list from 1997 until 1998. We have a list of names that should total up to a subtotal on the same page as of that date.)

FEST BUSINESS: $2417.60 total receipts.
Tony explained the debits and receipts from the FEST! After all costs were subtracted, the FEST balance is $842.81 to the good!

May 17 & 18, 2003 is next FEST!

Annual Picnic: Sept 21 Saturday at Swoger's.

New business:
A motion was made to add a second authorized person to be registered with the bank to deal with checks. Carried. Candidates are Justin Wagner and Howard Luckey.

Newsletter to get out by end of October. 5 Pages is the suggested length.

Notes taken on laptop computer by Bob Swoger.

Adjourn at 8:37 p.m.

The July newsletters were put together in preparation for mailing. Bob will plan on a couple of html lessons. We adjourned to Sante's Restaurant

---

August 8, 2002

Howard called the meeting to order at 7:50 PM at the Schaumburg Township District Library. Present were Scott Montgomery, Brother Jeremy, Bob Swoger, Howard Luckey, Rich Bair, Brian Goers and Tony Podraza.

Howard read his notes regarding last month's meeting as Bob had not yet gotten them to Tony.

Old business:

Previous month's motion to add second authorized person to registered names to deal with checks. Justin or Howard not yet authorized with the bank due to George's absence.

Annual Picnic confirmed to be Sept 21 Saturday.

May 17 & 18, 2003 is next FEST! Steve Noskowicz will probably return, Burke & Burke asked to come. Tony has been in contact by e-mail. Ask if Burke & Burke can get Frank Davis to put out a Burke & Burke CD. (This may be in conflict with Chris' plan to put the software up on his website.
Bob could not read the Dave Kiel CoCo emulator disk. Tony says Chris Burke is asking for help retrieving lost programs. Tony & Brother Jeremy are planning to help.

New business:
IDE boards: Rich Bair requested from Brian an IDE board.

Charlie Strong goes home to the Lord.
Bob Swoger retires from Motorola.
Brother Jeremy leaves his temporary job at UPS.

Howard submits 3 pieces of art for next FEST! Bob will post them all in the future.
Next year's theme by Scott Montgomery: "12 years and still haven't learned". (Tentative)

Brian Goers wants to do the next FEST! show guide.

Brian Goers brought in 3 event counters for demo. What to count:
Fridge door openings
Toilet seat lifts.
(Talk about exotic uses)....

The "business meeting" adjourned at 8:48 p.m.
Howard handed out more calendars.

We all regrouped at Sante's Restaurant after the meeting.

Update on Charley Strong
Date: 8/5/02 8:25:51 PM Central Daylight Time
From: brjeremy@JUNO.COM
To: coco@Princeton.EDU
Sent from the Internet (Details)

Dear John:

God grant Charley eternal rest. We will remember him at Mass tomorrow.

Into your hands, O merciful Savior, we commend your servant Charley. Acknowledge, we humbly beseech you, a sheep of your own fold, a lamb of your own flock, a sinner of your own redeeming. Receive him into the arms of your mercy, into the blessed rest of everlasting peace, and into the glorious company of the saints in light. Amen.

May his soul and the souls of all the departed, through the mercy of God, rest in peace. Amen.

Your brother in Christ,
Jeremy, CSJW
brjeremy@juno.com

Sept 12 2002

The meeting was called to order at 7:45PM.

Present were Tony Podraza, Scott Montgomery, Howard Luckey and Bob Swoger, George Schneeweiss, Justin Wagner, Brother Jeremy.

Old business.
Didn't get in touch with Justin or George. Howard shall contact George regarding the checking account authorizations.

Must get count on those interested in attending the picnic:
Howard, Tony, Brian G, and bob Swoger will be at the picnic. Bro. Jeremy, John Chasteen.
Try to contact the following: Brooks, Kowalski, geo no
justin no
Brian Schubring, Mike Warns, Eddie Kuns, Carl Boll, Joel Mathew Hegberg, Rich Bair

Wednesday deadline for picnic attendees
.
Bob to get address from Tony to send card to the Strong family.

The treasurer reports that the bank account is healthy... QUITE!

New business
Geo told a Dell story – in short, his opinion is that Dell makes vacuum cleaners.
8:29 Business meeting adjourns. Fun followed.

Oct 10, 2002
20:50 (hmmmm, what's wrong with this picture?)

Present were Tony Podraza, Scott Montgomery, Howard Luckey and Bob Swoger, Brother Jeremy & JOHN CHASTEEN,.

Old business:
Didn't get in touch with Justin or George (bank account). Howard shall contact George.

Picnic Attendees: Brian Schubring, Mike Warns, Bob Swoger, Rich Bair, Larry Sauter, Tony & Linda Podraza, Annette Swoger, David Kiel
Get card to Strong
New business:
Moving the FEST! to the Best Western. (we just wanted to see who was reading this... The location is the same..but the hotel affiliation has changed from Ramada to Best Western...look a bit further on for more details)

Letter from Greg Gillen to send us CoCo Goodies.

21:09 Adjourn

Just What is DLOAD/DLOADM, Anyway?

From: Neil Morrison

DLOAD works FWIW. DLOADM has a bug. There was a patch for DLOADM.
See http://www.getnet.net/~markdeb/ftp/coco/dload.txt
Also from Google:News
From: microsof!hanss (microsof!hanss)
Subject: CoCo DLOAD doc. 170 lines
Newsgroups: net.micro
View: (This is the only article in this thread) | Original Format
Date: 1982-12-05 11:14:39 PST

Having seen several requests for documentation on the Radio Shack color computer's DLOAD command, I asked the right person (Mark C, who does not read news, nor does he have access to uucp) and got the document below in reply. Earlier I tried to get it to a previous requestor by mail and asked for acknowledgement. Since I didn't get any I assume the paths didn't work. Submitting to news is a third and final resort.

I don't know anything about this stuff so please don't send letters about it to me. I'm just forwarding the info since I knew where it could be found. -Hans Spiller

Microsoft 6809 Extended Color BASIC

DLOAD/DLOADM Communications Protocol
Copyright (c) 1982 by Microsoft Corporation
Prepared by Mark L. Chamberlin
August 31, 1982
DLOAD/DLOADM Communications Protocol
Page 2

DLOAD and DLOADM send and receive packets of information to and from the host computer in order to download BASIC and machine language files. The protocol used for sending these packets was designed to facilitate detection and correction of transmission errors.

The process of downloading a file from the host involves opening a file, and then reading blocks of data from the file. The details of this protocol follow.

OPEN FILE -

1. BASIC to host - P.FILR
2. Host to BASIC - P.FILR
3. BASIC to host –
    1. 8 byte filename, left justified, blank filled
    2. XOR of the bytes in the filename
4. Host to BASIC -
    a) If no errors detected -
    1. P.ACK
    2. file type (0=BASIC program, 2=machine language, FF=file not found)
    3. ASCII flag (0=binary file, FF=ASCII)
    4. XOR of file type and ASCII flag.
    b) If errors detected, P.NAK.
5. BASIC - If errors then return to step 1.

READ BLOCK -

1. BASIC to host - P.BLKR
2. Host to BASIC - P.BLKR
3. BASIC to host -
    1. Block number (most significant 7 bits)
    2. Block number (least significant 7 bits)
    3. XOR of block number bytes

NOTE

The block number is a fourteen bit, unsigned integer in the range 0 through 16383. It is split into two seven bit values, each of which is transmitted in the least significant 7 bits of a byte. This insures that the most significant bit is not set except for the protocol control characters (e.g., P.BLKR). For example, a block number of 511 is transmitted as binary 00000011 and 01111111.

4. Host to BASIC -
   a) If no errors detected -
   1. P.ACK
   2. Block length in bytes (0 through 128, 0 indicating end of file)
   3. 128 bytes of data

NOTE

128 bytes of data must be sent, regardless of the block length. If the block length is less than 128 the extra bytes are read by BASIC but not used, so their values are of no concern.

   4. XOR of block length and data bytes
   b) If errors detected, P.NAK.
   5. BASIC - If errors then return to step 1.

The control character definitions are:
1. P.ACK - Acknowledge - C8 hex.
2. P.ABRT - Abort - BC hex.
3. P.BLKR - Block request - 97 hex.
4. P.FILR - File request - 8A hex.
5. P.NAK - Negative Acknowledge - DE hex.

Additional rules:

1. If errors occur during an OPEN or READ sequence, BASIC will retry the operation. For each sequence, a maximum of 5 tries is attempted. After 5 unsuccessful attempts, BASIC transmits a P.ABRT to the host and aborts the download, causing a BASIC IO error to occur.
2. If more than 10.4 seconds pass while BASIC is waiting for a byte from the host, a timeout error occurs. The operation is retried or aborted as described above.
3. If the host receives any unsolicited data, it quit transmitting and wait for BASIC to restart the sequence.
4. The host should never time out. It should just continue to wait.

BEWARE of THE BLOB

From: Gene Heskett
<gene_heskett@IOLINC.NET>
George Ramsover wrote:
" I went through the Boot List Order Bummer a long time ago. Finally, I got the boots I needed. But I never did understand how the BLOB occurs, what causes it and all that jazz.
Please explain this to me/us.
Why does it happen?"
George

Heck of a good question, George. The original Boot List Order Bug was/is a more or less bootfile size from head to cc3disk module 4 triggered function that when it was finally nailed down, turned out to be an ambiguity in understanding how the GIME worked in its internal IRQ prioritizing/handling. I had 4 different lengths of INIT modules, each a byte longer than the next smaller one, that I selectively inserted early in the bootfile to adjust things. It all worked as long as it was both the right length and in the bootfile ahead of cc3disk.

Once the GIME's IRQ register was properly reset, which guaranteed that the next IRQ would get thru, then the BLOB was supposedly a thing of the past. To have Alan bring it up in connection with an FDC chip that wasn't used that much anyway caught me totally off-guard.

The code that actually does that (the BLOB fix) is in fact in the clock module since that's the heart and soul of os9's IRQ per tick driven multitasking. Eddie K. was out first with the fix, then had to fix the fix a week or 3 later, see the rainbows of that time period, and all the clocks that I walked around in for the Disto 4n1 RTC should be correct. Likewise, the XT-RTC reworks I did should be correct, as I'm pretty sure the last one of those I released was after the grand old BLOB shooting party we all had at the time. ISTR I even bought a 6 pack to celebrate. :-)
Cheers, Gene

BLOBSTOP V1.0

Michael Shell
Technical Discussion of the BLOB
September 1, 1994

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.

*** 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 FDC. 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 cannot 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 command 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   lbsr    next1
next1   lbsr    next2
next2   lbsr    next3
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 "programmable".

This code really flies with a 6309 in native mode. The 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 xfer 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 an xfer 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 500kbits/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 occurrence 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 infinite 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. 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 further 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.

So, here is the 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.

Here is the 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
```

Here's 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
```

Here is the 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
*       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
```

Here's the 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
```

Here is the 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
```

Here's the 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 xfer speed. We could get almost twice that if 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 xfers 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
September 1, 1994

Thoughts on Disk BASIC and Hard Drives

Boisy G. Pitre

One possible concern that might arise from having so many virtual drives on one hard disk is the problem of retention. What happens if data integrity on the drive goes bad, or worse yet, a major crash of the disk?

I've been using an external SCSI ZIP-100 drive quite successfully under HDB-DOS for some time. ZIP100 disks have 196,608 ($30000) sectors, and I set an OS-9 offset of $008A00 in my HDB-DOS ROM. This gives OS-9 about 14 meg, which is plenty. The other 82 meg gives me 256 virtual drives ($30000-$8A00 = $27600, the total number of sectors needed for 256 virtual drives).

The advantages of using ZIP disks are (1) they are removable, and (2) they are cheap and readily available at any local superstore. Four ZIP disks configured in the above manner can hold 1,024 virtual drives!

One could devise an easy backup method if there were two ZIP drives hooked up to a CoCo, but those with Linux and a ZIP drive can use a more clever solution.

When I want to backup my CoCo's ZIP-100 disk, I insert it into the Linux ZIP drive and at the Linux shell prompt I type:

$ dd if=/dev/hdd of=/backup/cocozip

This command reads the entire /dev/hdd device (which is the ZIP drive) into a file called cocozip in the /backup directory. The result is a file that is exactly 100,663,296 bytes, a byte-for-byte image of the ZIP drive. I can insert a blank ZIP disk into the same drive and type the following command to make a copy of my original CoCo ZIP disk:

$ dd if=/backup/cocozip of=/dev/hdd

Note that the same thing could be done for hard drives, though a ZIP disk, being removable, is more convenient.

CoCo OS-9 Source at Sourceforge.net

Boisy Pitre <boisy@ACADIAN-EMBEDDED.COM>

Hello all,

I have made the entire collection of disassembled and commented source code for OS-9 Level One, OS-9 Level Two, and OS-9 Level Two Version 3 available at Sourceforge.net. This stems from work that Alan DeKok and I did some years back. Now we have a mostly complete disassembly of those packages, as well as other 3rd party OS-9 packages. This is a gold mine of information for those wanting to know the inner-workings of the CoCo OS-9 system, and is a base for expanding the operating system to the community's liking.

I would like to get NitrOS-9 sources there as well, but don't know who has the latest source.

If anyone is interested in actively helping with the continual work of disassembly and commenting of the source, please contact me with your sourceforge.net ID, and I'll add you to the list of developers.

The URL to the site is:
http://sourceforge.net/projects/cocoos9/

Those with access to CVS can use the following command to check out the source anonymously (you cannot commit any changes unless you are part of the development team):

```
cvs -
d:pserver:anonymous@cvs.sourceforge.net:/cvs
root/cocoos9 co os9
```

--

Boisy G. Pitre

id

DOCTOR! DOCTOR!

Marty Goodman

Sony KV1311 CR monitor

I have all the information you need for the Sony KV 1311 CR monitor.

BUT, there is a minor gotcha logistic involved in trying to make a cable between it and the CoCo 3:

The Sony KV 1311 CR DOES NOT have provisions for separate postive horizontal and vertical sync pulse inputs, of the sort the CoCo 3 provides. Instead, it wants to see a negative combined H &V sync input.

This signal is easily provided by running the CoCo 3's H and V sync inputs into the two inputs of a 74LS02 NOR gate, and taking the output of that NOR gate as the signal to send to the Sony KV1311CR composite sync input.

The only gotcha really is that you need 5 volts to power the 74LS02 chip, and this is NOT available on a stock Sony monitor's video connector, nor is it available on a stock CoCo 3's video output connector.

I've used a variety of different solutions to this problem:

(1) Put the added 74LS02 chip INSIDE the CoCo 3, powering it from the CoCo 3's main 5 volt regulated power supply, and running the composite sync output onto pin 10 of the CoCo 3's video connector (after DISCONNECTING pin 10 of the CoCo 3's video connector from the PIA chip... a function that was intended to at some future date provide some sort of monitor recognition, but never was actually imp limented by Tandy in any firm or soft ware or hardware.

(2) Go inside the Sony monitor and find a source of regulated +5 volts, then send that to pins 1 and 2 of the 34 pin connector on the Sony. Pins 1 and 2 are normally NOT CONNECTED on the Sony KV 1311 CR, but those pins DO carry +5 volts on some other Sony monitors from that period that use an otherwise identical pin assigment on the 34 pin connector to that used on the Sony KV 1311CR.

(3) Make a little adaptor box, with the 74LS02 in it, and power it via an external source of regulated +5 volts from a +5 regulated wall xformer or a 9 volt xformer and a 78LS05 regulator.

Caution! Do NOT (as I initially did in an early run of such cables) try to "steal" the needed 5 volts from the 5 volt line on the joystick connectors! This will result in the joystick not having a full range of numbers for the full range of throw of the stick, due to the use of a 100 ohm current limiting resistor in the joystick connector circuit where the 5 volts is supplied.

You COULD elect to use pin 10 of the video connector on the CoCo to bring out 5 volts regulated (after first cutting the trace that connects it to a PIA input).

Other tip: When making your cable, if you use ribbon cable, run a ground wire between ALL signal lines... especially have ground wires on all sides of the R,G, and B signal lines. And keep the cable length under 5 feet. This will guarantee good signal quality on the monitor.

Here's that pin out for the Sony KV1311 CR connector (or at !east the relevant part of it): (with notes on variations on some other Sony monitors that use the same 34 pin connector)

1,2 not connected (+5 volts on some other Sony models)
3 not connected (audio Right Channel input ground on some other Sony models)
4 ground
5 not connected (remote control ground on some other models)
6 ground (composite video output on some other models)
7 ground (audio Left Channel input ground on some other models)
8 -16 ground
17 not connected
18 not connected (Ext / Int Sync switch mode switch on some other models)
19 not connected
20 not connected (Audio Right Channel signal input on some other models)
21 not connected (Analog / digital mode select. High (open) = analog ground
Low (ground)= digital
This function is found NOT on th KV1311 but on some other Sony models)
22 not connected (remote control output on some other models)
23 composite video output
24 audio signal input (audio Left channel input on some other models)
25 Red analog video input
26 Green analog video input
27 Blue analog video input
28 not connected
29 fast blanking input
30 Composite Sync input (Horizontal or composite sync input on some other Sony models)
31 not connected (Vertical Sync input on some OTHER Sony models)

32 not connected (half blanking input on some other Sony monitors)
33 RGB / Normal select input
high (5 volts) = RGB input selected
low (ground) composite video input selected

34 audio select high (5 volts) audio is accepted from the 34 pin RGB multi input vide connector
low (ground) audio is accepted from the RCA Audio Input jacks on the monitor.

Note that pin numbering on this connector has all pins in one row be ODD, all pins in the OTHER row be EVEN. This is quite different from how pins are numbered on a DB type connector!!! On DB connectors pin numbering goes 1,2,3,4,5, etc... down one row, and continues on the next row. But on THIS dual row type connector, pin numbering goes 1,3,5,9 etc. on one row and 2,4,6,8, etc. on the other row.

Example of an "other" Sony model with the extra pin functions: KX-1901 "Profeel" Sony Trinatron component TV.

Obviously the pins you critically want to use are the R,G, and B analog video inputs, grounds, composite sync pins, and (if you want to be slick) the audio input and the audio input site select pin tied high... and perhaps the RGB / composite select pin, too (for if you don't tie that pin high, you must push in the RGB select button on the TV set in order for the analog RGB cable to work)

THERE you are! Pretty much all of the info I have on those inputs. Good luck!
Happy tinkering!

---marty

YOU SAY YOU'RE READY FOR A COCOFEST!?

WELL, DO I HAVE SOME NEWS FOR YOU, BUT YOU'LL HAVE TO GO TO THE NEXT PAGE TO READ IT...

12 years and we still haven't learned !!!

The Glenside Color Computer Club, Inc. presents
The TWELFTH Annual "Last" Chicago CoCoFEST!

May 17th & 18th, 2003

(Sat. 10am-5pm; Sun. 10am-4:30pm)
BEST WESTERN INN of ELGIN
(NOTE THE NAME CHANGE)
345 W. River Road (A city block from I-90 & IL-31)
Elgin, Illinois
(SAME GREAT LOCATION AS LAST 10 YEARS!)
Overnight single occupancy room rate: $59.00 (plus 10% tax)
There is a $10.00 surcharge for each additional person
(Example: 3 people = $79.00 per night; plus tax)
Call 1-847-695-5000 for reservations.
Be sure to ask for the "CoCoFEST!" rate.
THERE IS A LIMITED SUPPLY OF ROOMS BLOCKED OUT FOR THE FEST. RESERVE YOUR ROOM EARLY THESE ROOMS WILL BE RELEASED FOR REGULAR RESERVATIONS ON May 8, 2003 AND WILL NOT, !NOT! !NOT! BE AVAILABLE TO THE FEST ATTENDEES
YOU MUST REGISTER UNDER "COCOFEST!" TO GET THIS RATE

WHY? WHY? OH WHHYYYYY? DO WE DO THIS?
A. To provide vendor support to the CoCo Community
B. To provide Community support to the CoCo Vendors
C. To provide educational support to new users.
D. TO HAVE AN OUTRAGEOUSLY GOOD TIME!!!!!

(HOW MUCH?
1) General Admission, ALL ATTENDEES: $5.00/day, $7.00 whole show
Children 10 and under - FREE

Advance ticket sales available between 2/01/2003 and 4/20/2003 from:
George Schneeweiss
13450 N 2700 E Road
Forrest, IL 61741
Include a Self-Addressed-STAMPED-Envelope (SASE)

For further information, general or exhibitor, contact:
Brian Goers, VP, Spcl Evnts, GCCCI
708-754-4921, VOICE
bgoers@ais.net
or
Tony Podraza, Secretary, GCCCI
87-428-3576, VOICE
tonypodraza@juno.com

Be sure to visit GLENSIDE"S Website @
"http://members.aol.com/clubbbs/glenside/"

EOF

"Getting off of this dirty bus; first time I understood; It's got to be the goin', not the getting there, that's good. That's a thought for keeping, if I could..."
–Harry Chapin

I would be greatly remiss if I did not let you know of the passing of a very dear, if not close, personal friend of mine and of the CoCo Community. Charley Strong, after a valiant struggle against leukemia, which included a marrow transplant in early June, left this world on August 5th @ 4:00PM. If I remember correctly, it was Charley's love of gaming that led his brother, John, into programming games for the Color Computer, and from there, the birth of Strongware.
You couldn't miss Charley's smiling face at the FEST!s From the moment you first spoke with him, you were his best friend. He could always find the rainbow behind ANY cloud. While he was in the hospital, waiting for the results of tests, and getting ready for his marrow transplant procedure, he called me to let me know how he was doing. To hear him speak, you wouldn't guess that he was about to undergo such a serious operation. His cheerful voice never wavered when he told me about his faith in his Savior and his witnessing to the hospital personnel, who were very much surprised that he would be so cheerful and caring about THEIR relationships with God, instead of worried about his own physical troubles. You see, he knew that whatever the outcome of the situation, his own destination was already taken care of. All he needed to do to make sure that his trip to the destination was a good one was to tell others how they could be sure of their destination, too.

It was a good trip, Charley, it surely was.
"...It's got to be the goin', not the getting there, that's good."

Until next time....
I bid you Peace.
-tony podraza