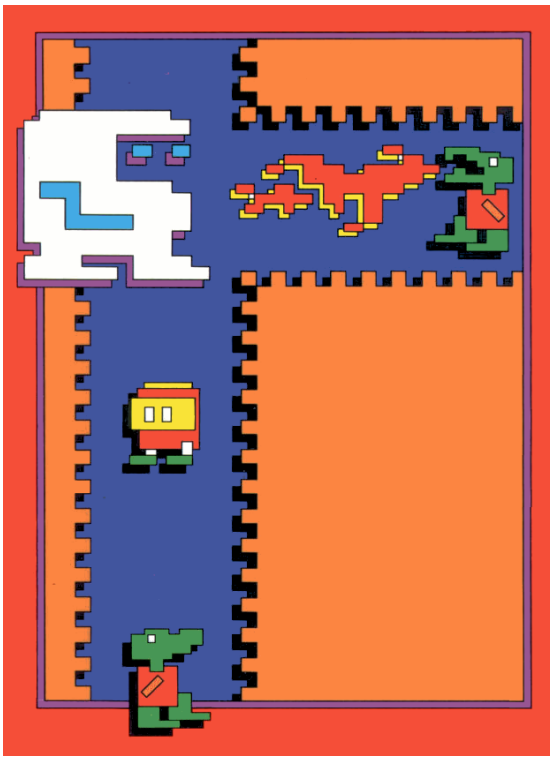
DIG DUG



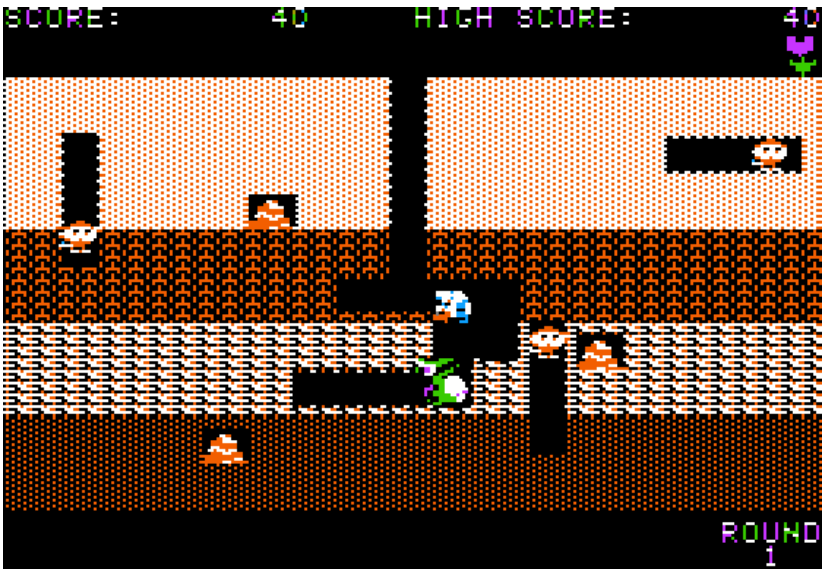
2015-05-05

4am
to deprotect
and preserve

Contents

----------Dig Dug----------
A 4am crack 2015-05-05

Name: Dig Dug
Genre: arcade
Year: 1983
Publisher: Thunder Mountain
Media: single-sided 5.25-inch floppy
OS: DOS 3.3 derivative (T02,S02 contains the string "C1984 RPS" backwards)
Other versions: several file-based cracks, but most (all?) are based on Atari's version



Chapter 0
In Which Various Automated Tools Fail In Interesting Ways

COPYA
  disk read error on first pass

Locksmith Fast Disk Backup
  reads T00,S00..T02,S04, but nothing
  beyond that (not even the rest of the
  sectors on track $02)

EDD 4 bit copy (no sync, no count)
  no errors, but copy grinds and
  eventually crashes before loading DOS

Copy][+ nibble editor
  T00-T02 appear normal (not sure why
  my EDD bit copy couldn't read itself)
  T03+ have a modified data epilogue
  (not consistent)

Disk Fixer
  ["0" -> "Input/Output Control"]
    set CHECKSUM ENABLED to NO
  all tracks readable, but sector data
  appears to be garbled/encrypted

For example, here is T11,S0F:

```
                --v--

-------------- DISK EDIT --------------
TRACK $11/SECTOR $0F/VOLUME $FE/BYTE$00
---------------------------------------
$00:>FF<EE F1 FF 00 FF 00 FF    .nq.@.@.
$08: 00 FF 00 12 0F 02 C8 C5   @.@ROBHE
$10: CC 30 CF A0 5C A0 5C A3   L0O \ \#
$18: 5F A0 5C A0 5C A0 5C A3   _ \ \ \#
$20: 5F A3 5F A3 5F A0 5C A0   _#_#_ \
$28: 5C A0 5F A0 02 FC EF F3   \ _ B|os
$30: F8 2E 29 32 CE C5 D2 A0   x.)2NER
$38: 5C A0 5C A3 5C A3 5C A0   \ \#\#\
$40: 5C A3 5F A3 5C A0 5C A0   \#_#\ \
$48: 5C A0 5C A3 5C A3 5C FF   \ \#\#\.
$50: 00 14 0F 04 D7 D3 A3 5F   @TODWS#_
$58: A3 5F A0 5F A0 5F A0 5F   #_ _ _ _
$60: A0 5C A0 5C A0 5C A0 5C    \ \ \ \
$68: A0 5C A0 5C A0 5F A3 5C    \ \ _#\
$70: A0 5C DE FC EB F0 FB 2D    \^|kp€-
$78: 3A 33 CC C3 3D 28 39 5C   :3LC=(9\
---------------------------------------
BUFFER 0/SLOT 6/DRIVE 1/MASK OFF/NORMAL

---------------------------------------
COMMAND : _

                --^--
```

Why didn't COPYA work?
  modified epilogue bytes (T02,S05+)

Why didn't Locksmith FDB work?
  modified epilogue bytes (T02,S05+)

Why didn't my EDD copy work?
  unclear; maybe a funky RWTS?

Next steps:

  1. AUTOTRACE to capture RWTS
  2. Advanced Demuffin to convert disk
     to standard format
  3. Patch the RWTS

Chapter 1
In Which We Attempt To Use The Original Disk As A Weapon Against Itself, And It Goes Very Badly

```
[S6,D1=original disk]
[S6,D2=blank disk]
[S5,D1=my work disk]

]PR#5
...
CAPTURING BOOT0
...reboots slot 6...
...reboots slot 5...
SAVING BOOT0
CAPTURING BOOT1
...reboots slot 6...
...reboots slot 5...
SAVING BOOT1
SAVING RWTS

]BRUN ADVANCED DEMUFFIN 1.5

["5" to switch to slot 5]

["R" to load a new RWTS module]
  --> At $B8, load "RWTS" from drive 1

["6" to switch to slot 6]

["C" to convert disk]

...Advanced Demuffin crashes...

Wait, what?
```

```
]PR#5
]BLOAD BOOT1,A$2600
]CALL -151

*FE89G FE93G      ; disconnect DOS
*B600<2600.2FFFM ; move RWTS into place
*B700L
.
. nothing unusual, until...
.
B747-   4C C1 B3    JMP   $B3C1
```

That's definitelys suspicious, but it's called too late in the boot process to be related to my current problem.

```
*B800L
.
. nothing unusual, until...
.
; check for data epilogue bytes
B92F-   BD 8C C0     LDA   $C08C,X
B932-   10 FB        BPL   $B92F
B934-   C9 DE        CMP   #$DE
B936-   D0 0A        BNE   $B942
B938-   EA           NOP
B939-   BD 8C C0     LDA   $C08C,X
B93C-   10 FB        BPL   $B939
B93E-   4C B3 B6     JMP   $B6B3   <-- !
B941-   EA           NOP
B942-   38           SEC
B943-   60           RTS
```

Well that explains it. The RWTS is calling a custom routine outside the $B800..$BFFF range. That part wasn't loaded into memory when I ran Advanced Demuffin and loaded the "RWTS" file, so the RWTS crashed and brought Advanced Demuffin down with it.

```
*B6B3L

; check for second epilogue byte as
; usual
B6B3-   C9 AA       CMP   #$AA

; but then branch to do... something
B6B5-   F0 02       BEQ   $B6B9
B6B7-   60          RTS
B6B8-   00          BRK

; get another byte
B6B9-   BD 8C C0    LDA   $C08C,X
B6BC-   10 FB       BPL   $B6B9

; kill some time (6 cycles for the JSR
; and other 6 for the RTS on the other
; end)
B6BE-   20 B7 B6    JSR   $B6B7

; and again
B6C1-   20 B7 B6    JSR   $B6B7
B6C4-   EA          NOP

; read the data latch (note: no loop
; here, we're just reading it once)
B6C5-   BD 8C C0    LDA   $C08C,X

; if high bit is set, branch
B6C8-   30 04       BMI   $B6CE

; a counter of some kind?
B6CA-   EE 8A B7    INC   $B78A
B6CD-   18          CLC
B6CE-   60          RTS
```

```
*B78A

B78A- 00          ; counter starts at 0
```

This explains two things at once:

1. Advanced Demuffin crashed because the RWTS is calling a routine outside the "RWTS" file.

2. My EDD bit copy can't read itself because it's not preserving the timing bit that this routine is checking after the data epilogue.

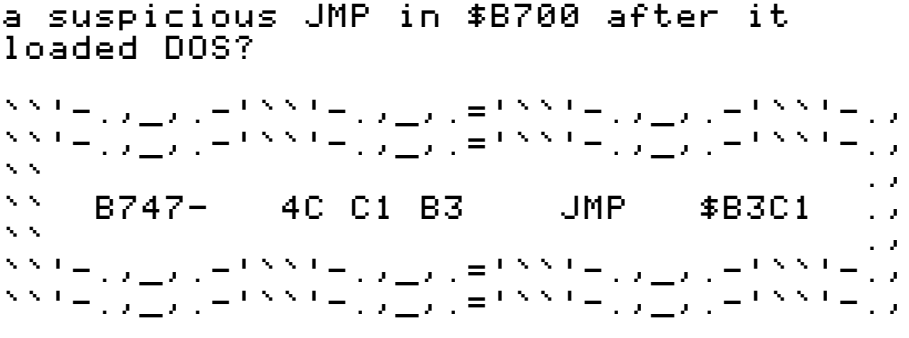
I'll re-run Advanced Demuffin, but this time I'll load the "BOOT1" file (which comprises $B600..$BFFF) instead of the "RWTS" file.

```
*C500G
...
]BRUN ADVANCED DEMUFFIN 1.5

["5" to switch to slot 5]

["R" to load a new RWTS module]
  --> At $B6, load "BOOT1" from drive 1

["6" to switch to slot 6]

["C" to convert disk]
```

```
                    --v--

ADVANCED DEMUFFIN 1.5    (C) 1983, 2014
ORIGINAL BY THE STACK    UPDATES BY 4AM
=======PRESS ANY KEY TO CONTINUE=======
TRK:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
+.5:
     0123456789ABCDEF0123456789ABCDEF012
SC0:...RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC1:...RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC2:...RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC3:...RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC4:...RRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC5:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC6:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC7:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC8:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SC9:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCA:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCB:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCC:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCD:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCE:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
SCF:..RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
=======================================
16SC $00,$00-$22,$0F BY1.0 S6,D1->S6,D2

                    --^--
```

Well, I suppose that's progress.

Chapter 2
In Which I Attempt To Use ASCII Art To Simulate A Flashback, And It Goes Very Badly

```
Something is modifying the RWTS after
DOS loads.

Heyyyyy, wait a minute. Wasn't there
a suspicious JMP in $B700 after it
loaded DOS?




I bet that's the routine that changes
the RWTS to read the rest of the disk.
But AUTOTRACE didn't capture it, so I'm
going to have to write a custom trace
like some kind of 20th century peasant.

]PR#5
]CALL -151

*9600<C600.C6FFM

; set up callback #1 after boot0 loads
; boot1/RWTS
96F8-   A9 4C       LDA   #$4C
96FA-   8D 4A 08    STA   $084A
96FD-   A9 0A       LDA   #$0A
96FF-   8D 4B 08    STA   $084B
9702-   A9 97       LDA   #$97
9704-   8D 4C 08    STA   $084C

; start the boot
9707-   4C 01 08    JMP   $0801
```

```
; callback #1 is here --
; set up callback #2 after boot1 loads
; DOS
970A-   A9 4C       LDA   #$4C
970C-   8D 47 B7    STA   $B747
970F-   A9 1C       LDA   #$1C
9711-   8D 48 B7    STA   $B748
9714-   A9 97       LDA   #$97
9716-   8D 49 B7    STA   $B749

; continue the boot
9719-   4C 00 B7    JMP   $B700

; callback #2 is here --
; copy DOS to graphics page so it
; survives a reboot
971C-   A2 25       LDX   #$25
971E-   A0 00       LDY   #$00
9720-   B9 00 9B    LDA   $9B00,Y
9723-   99 00 2B    STA   $2B00,Y
9726-   C8          INY
9727-   D0 F7       BNE   $9720
9729-   EE 22 97    INC   $9722
972C-   EE 25 97    INC   $9725
972F-   CA          DEX
9730-   D0 EE       BNE   $9720

; reboot to my work disk
9732-   4C 00 C5    JMP   $C500

*BSAVE TRACE2,A$9600,L$135
*9600G
...reboots slot 6...
...reboots slot 5...
```

```
]BSAVE BOOT2 9B00-BFFF,A$2B00,L$2500
]CALL -151

*FE89G FE93G
*9B00<2B00.4FFFM
*B3C1L

; overwrite the JMP instruction that
; got us here (to cover its tracks
; in memory)
B3C1-   A0 84       LDY   #$84
B3C3-   8C 48 B7    STY   $B748
B3C6-   A0 9D       LDY   #$9D
B3C8-   8C 49 B7    STY   $B749

; fiddle with nibble write table
B3CB-   A0 96       LDY   #$96
B3CD-   8C 68 BA    STY   $BA68
B3D0-   A0 FF       LDY   #$FF
B3D2-   8C 29 BA    STY   $BA29

; more RWTS fiddling
B3D5-   A0 00       LDY   #$00
B3D7-   8C FF BA    STY   $BAFF
B3DA-   A0 3F       LDY   #$3F
B3DC-   8C 96 BA    STY   $BA96

; check that mysterious counter that
; was incremented at $B6CA after
; checking the data epilogue
B3DF-   AD 8A B7    LDA   $B78A
B3E2-   C9 18       CMP   #$18

; looks like branching is good...
B3E4-   B0 25       BCS   $B40B

; because this is definitely bad
B3E6-   D8          CLD
```

```
; push $DFFF onto the stack
B3E7-   A9 DF        LDA   #$DF
B3E9-   48           PHA
B3EA-   A9 FF        LDA   #$FF
B3EC-   48           PHA

; set up a memory copy to wipe all of
; main memory (copying $0800 to $0801
; and incrementing)
B3ED-   A9 08        LDA   #$08
B3EF-   85 3D        STA   $3D
B3F1-   85 43        STA   $43
B3F3-   A9 BF        LDA   #$BF
B3F5-   85 3F        STA   $3F
B3F7-   A9 00        LDA   #$00
B3F9-   85 3C        STA   $3C
B3FB-   85 42        STA   $42
B3FD-   E6 42        INC   $42
B3FF-   A9 FE        LDA   #$FE
B401-   85 3E        STA   $3E
B403-   A0 00        LDY   #$00
B405-   AD 82 C0     LDA   $C082

; and exit via the memory copy routine
; (then pop $DFFF off the stack and
; land on a cold-started BASIC prompt)
B408-   4C 2C FE     JMP   $FE2C

; successful execution continues here
; (from $B3E2) --
; reset data epilogue RWTS code
B40B-   A9 C9        LDA   #$C9
B40D-   8D 3E B9     STA   $B93E
B410-   A9 AA        LDA   #$AA
B412-   8D 3F B9     STA   $B93F
B415-   A9 F0        LDA   #$F0
B417-   8D 40 B9     STA   $B940
B41A-   A9 5C        LDA   #$5C
B41C-   8D 41 B9     STA   $B941
```

```
; copy the code that was supposed to be
; at $B6B3 into place
B41F-   A0 00       LDY   #$00
B421-   B9 2F B4    LDA   $B42F,Y
B424-   F0 06       BEQ   $B42C
B426-   99 B3 B6    STA   $B6B3,Y
B429-   C8          INY
B42A-   D0 F5       BNE   $B421

; continue boot
B42C-   4C 84 9D    JMP   $9D84
```

Let's make an RWTS that can read the rest of the disk.

```
*B3DF:60
*B3C1G
*2800<B800.BFFFM
*C500G
...
]BSAVE RWTS 3+,A$2800,L$800

]BRUN ADVANCED DEMUFFIN 1.5

["5" to switch to slot 5]

["R" to load a new RWTS module]
  --> At $B8, load "RWTS 3+" from D1

["6" to switch to slot 6]

["C" to convert disk]

["Y" to change default values]
```

```
                    --v--

ADVANCED DEMUFFIN 1.5     (C) 1983, 2014
ORIGINAL BY THE STACK     UPDATES BY 4AM
========================================


INPUT ALL VALUES IN HEX


SECTORS PER TRACK? (13/16) 16

START TRACK: $02          <-- change this
START SECTOR: $05         <-- change this
END TRACK: $22
END SECTOR: $0F

INCREMENT: 1

MAX # OF RETRIES: 0

COPY FROM DRIVE 1
TO DRIVE: 2
========================================
16SC $02,$04-$22,$0F BY1.0 S6,D1->S6,D2

                    --^--

And here we go...
```

```
                --v--

ADVANCED DEMUFFIN 1.5    (C) 1983, 2014
ORIGINAL BY THE STACK    UPDATES BY 4AM
=======PRESS ANY KEY TO CONTINUE=======
TRK:  R.................................
+.5:
    0123456789ABCDEF0123456789ABCDEF012
SC0:   .................................
SC1:   .................................
SC2:   .................................
SC3:   .................................
SC4:   .................................
SC5:  ..................................
SC6:  ..................................
SC7:  ..................................
SC8:  ..................................
SC9:  ..................................
SCA:  R.................................
SCB:  ..................................
SCC:  ..................................
SCD:  ..................................
SCE:  ..................................
SCF:  ..................................
=======================================
16SC $02,$04-$22,$0F BY1.0 S6,D1->S6,D2

                --^--
```

Well, I suppose that's progress.

Chapter 3
In Which I Attempt To Patch The RWTS, And It Goes Very Badly

I'm not sure what's going on with T02,S0A, but look at this:

```
]PR#5
]CATALOG,S6,D2

C1983 DSR^C#254
353 FREE

 A 002 HELLO
 B 003 RUNNER
 B 034 TP
 B 002 RELOCATE
 B 003 LOADER
 B 066 OBJECT CODE
 B 033 PI.LOGO

]RUN HELLO
...works...
```

(Note: any classic cracker would have stopped at this point, since the files can be copied to a standard disk and the game is completely playable. But my story isn't quite finished yet.)

[S6,D1=demuffin'd copy]

```
]PR#6
...grinds...
```

Of course, the demuffin'd copy can't read itself, because it still has that wacky call to $B6B3 during the data epilogue check.

```
T00,S03,$3E change "4C B3 B6 EA"
                   to "C9 AA F0 5C"
```

]PR#6
...loads DOS then exits to prompt...

Now my copy can read its own DOS, but it immediately exits to a BASIC prompt. $B6B3 was the routine that incremented the counter at $B78A. Since that is no longer called, the comparison at $B3E2 fails and the branch at $B3E4 is never taken.

I need to bypass the first half of the routine at $B3C1 (that fiddles with the RWTS and checks the counter), but I do still want to call the second half (that copies the proper code back to $B6B3). That part starts at $B41F.

T00,S01,$48 change "C1 B3" to "1F B4"

]PR#6
...loads DOS, fills screen with quotation marks, and freezes...

Well, I suppose that's progress. But that means there is still more copy protection, probably in the HELLO program or shortly thereafter.

No, wait, it can't be there, because I successfully ran the HELLO program after booting from my work disk.

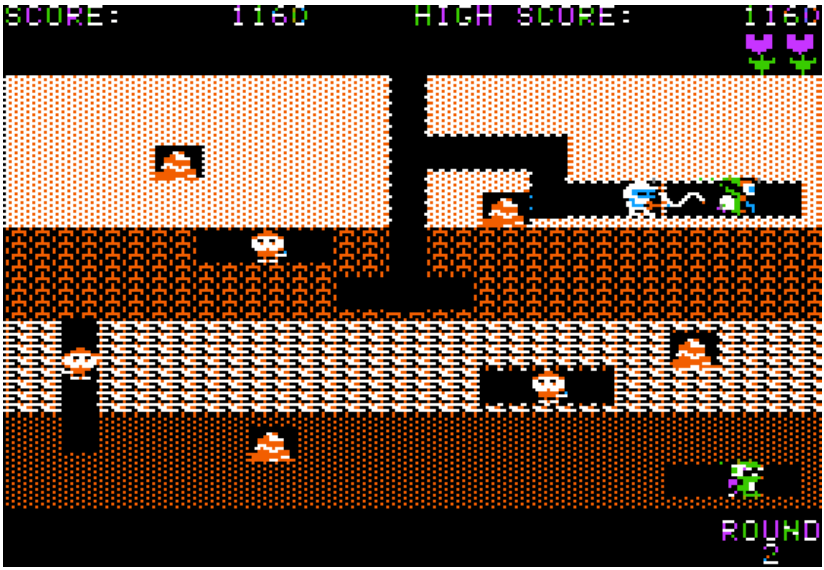
Then where the heck is it?

```
]PR#6
<Ctrl-C>
...quotation marks...

]PR#6
<Ctrl-Reset>
...reboots...

]PR#5
]LOAD HELLO,S6,D1
]LIST

 10  HOME : CLEAR
 20  PRINT  CHR$ (4);"BLOAD PI.LO
     GO"
 30 A =  PEEK (49239):A =  PEEK (
     49234):A =  PEEK (49237):A =
      PEEK (49232)
 40  FOR I = 1 TO 3000: NEXT I
 50  TEXT : HOME : CLEAR
 60  PRINT  CHR$ (4);"BRUN RUNNER
     "


]5 HGR: END
]SAVE HELLO
]CALL -151

*2000:FF N 2001<2000.3FFEM
```

I set the entire hi-res graphics page to white, so "HGR" should briefly show a white screen before clearing it in that classic HGR-y kind of way.

*C600G
...quotation marks...

It never executes the "HGR" command. Whatever is filling the screen with those blasted quotation marks, it happens after $B42C calls $9D84 to initialize DOS, but before DOS passes control to the HELLO program.



Chapter 4
In Which I Begrudgingly Agree To Trace The Entire DOS Boot, And It Goes Surprisingly Well

```
[S6,D1=demuffin'd copy without any
       changes to the HELLO program]

]PR#5
]BLOAD BOOT2 9B00-BFFF,A$2B00
]CALL -151

*FE89G FE93G
*B600<2600.4FFFM

*9D84L                  ; start of DOS init
.
. nothing unusual
.

*9E06L                  ; final DOS init
.
. nothing unusual
.
9E4D-   4C 80 A1    JMP   $A180
```

Rats. I was hoping that there would be a suspicious JMP at $9E4D. (I've seen this on other disks -- they jump to a nibble check instead of $A180.) But there's nothing suspicious here.

To confirm that my non-working copy is really getting this far, and that I haven't missed something, I'll do a temporary patch to change the JMP at $9E4D to $FF59 (unconditional break to monitor).

```
T00,S0D,$4E change "80 A1" to "59 FF"
```

```
*C600G
<beep>
...breaks into monitor...

OK, my non-working copy gets through
almost all of the DOS init, up to at
least $9E4D.

Let's take a look at $A180.

*A180L

A180-   20 5B A7    JSR   $A75B

*A75BL

; normal
A75B-   A0 00       LDY   #$00
A75D-   8C 51 AA    STY   $AA51
A760-   8C 52 AA    STY   $AA52
A763-   60          RTS

*A183L

A183-   20 AE A1    JSR   $A1AE

*A1AEL

; normal
A1AE-   A9 00       LDA   #$00
A1B0-   A0 16       LDY   #$16
A1B2-   99 BA B5    STA   $B5BA,Y
A1B5-   88          DEY
A1B6-   D0 FA       BNE   $A1B2
A1B8-   60          RTS
```

```
*A186L

; normal
A186-   AD 5F AA    LDA   $AA5F
A189-   AA          TAX
A18A-   BD 1F 9D    LDA   $9D1F,X
A18D-   48          PHA
A18E-   BD 1E 9D    LDA   $9D1E,X
A191-   48          PHA
A192-   60          RTS
```

End of the line. $AA5F acts as an index into a jump table, based on whether the autostart program is Applesoft BASIC, binary, or text.

```
*AA5F

AA5F- 06

*1E+06
=24

*9D24.9D25

9D24- D0 A4
```

So execution continues at $A4D1 (the address pushed to the stack, plus 1).

```
*A4D1L

A4D1-   AD B6 AA    LDA   $AAB6
A4D4-   F0 03       BEQ   $A4D9
A4D6-   8D B7 AA    STA   $AAB7
A4D9-   20 13 A4    JSR   $A413
A4DC-   20 C8 9F    JSR   $9FC8
A4DF-   20 51 A8    JSR   $A851
A4E2-   4C 8B B7    JMP   $B78B   <-- !
```

Wait, what? $B78B is not an entry point to anything. On an unprotected DOS 3.3 disk, it's the tail end of the routine at $B74A that writes DOS to tracks 0-2 at the end of the INIT command.

```
*B78BL

; unconditional branch backwards
B78B-   A9 00       LDA   #$00
B78D-   F0 BC       BEQ   $B74B

; this is another unconditional branch
; backwards (might be another entry
; point for something nefarious, but
; it's never executed right now)
B78F-   A9 01       LDA   #$01
B791-   D0 B8       BNE   $B74B

*B74BL

; save the marker that was set at
; either $B78B or $B78F
B74B-   48          PHA
```

```
; read T02,S05
B74C-   A9 02       LDA   #$02
B74E-   8D EC B7    STA   $B7EC
B751-   A9 00       LDA   #$00
B753-   8D EB B7    STA   $B7EB
B756-   8D F2 B7    STA   $B7F2
B759-   A9 05       LDA   #$05
B75B-   8D ED B7    STA   $B7ED

; into $B436
B75E-   A9 36       LDA   #$36
B760-   8D F0 B7    STA   $B7F0
B763-   A9 B4       LDA   #$B4
B765-   8D F1 B7    STA   $B7F1
B768-   A9 01       LDA   #$01
B76A-   8D E1 B7    STA   $B7E1
B76D-   8D F4 B7    STA   $B7F4
B770-   20 93 B7    JSR   $B793

; branch if read was successful
B773-   90 09       BCC   $B77E

; pseudo-reset if read failed
B775-   4C 62 FA    JMP   $FA62

;[always skipped]
;B778-   EA          NOP
;B779-   EA          NOP
;B77A-   EA          NOP
;B77B-   EA          NOP
;B77C-   EA          NOP
;B77D-   EA          NOP

; call the code from T02,S05 that we
; just read
B77E-   20 36 B4    JSR   $B436
```

```
; pop the marker
B781-   68          PLA

; exit via one of these indirect jumps
B782-   F0 03       BEQ   $B787
B784-   6C 72 AA    JMP   ($AA72)
B787-   6C 58 9D    JMP   ($9D58)
```

Chapter 5
In Which I Run Into An Old Friend, And Our Adventure Concludes

I've captured the entire DOS on this disk, but I don't have the additional code that's loaded at $B436.

```
*C500G
...
]CALL -151

*9600<C600.C6FFM

; set up callback #1 after boot0
96F8-   A9 4C       LDA   #$4C
96FA-   8D 4A 08    STA   $084A
96FD-   A9 0A       LDA   #$0A
96FF-   8D 4B 08    STA   $084B
9702-   A9 97       LDA   #$97
9704-   8D 4C 08    STA   $084C

; start the boot
9707-   4C 01 08    JMP   $0801

; callback #1 is here --
; set up callback #2 after T02,S05 is
; loaded
970A-   A9 4C       LDA   #$4C
970C-   8D 7E B7    STA   $B77E
970F-   A9 1C       LDA   #$1C
9711-   8D 7F B7    STA   $B77F
9714-   A9 97       LDA   #$97
9716-   8D 80 B7    STA   $B780

; continue the boot
9719-   4C 00 B7    JMP   $B700
```

```
; callback #2 is here --
; copy the mystery sector to the hi-res
; graphics page so it survives a reboot
971C-   A0 00       LDY   #$00
971E-   B9 36 B4    LDA   $B436,Y
9721-   99 36 24    STA   $2436,Y
9724-   C8          INY
9725-   D0 F7       BNE   $971E

; reboot to my work disk
9727-   4C 00 C5    JMP   $C500

*BSAVE TRACE3,A$9600,L$12A
*9600G
...reboots slot 6...
<beep>
```

Oops, I forgot I patched T00,S0D to unconditionally break to the monitor.

```
T00,S0D,$4E change "59 FF" to "80 A1"

]PR#5
]BRUN TRACE3
...reboots slot 6...
...crashes at $971E...
```

Right. DOS has loaded the HELLO file from disk by now, which means my trace program at $9600 got overwritten by the DOS file buffer.

```
*C500G
...
]BLOAD TRACE3
]CALL -151
```

```
96F8-   A9 4C       LDA   #$4C
96FA-   8D 4A 08    STA   $084A
96FD-   A9 0A       LDA   #$0A
96FF-   8D 4B 08    STA   $084B
9702-   A9 97       LDA   #$97
9704-   8D 4C 08    STA   $084C
9707-   4C 01 08    JMP   $0801

; force $B77E to jump to the monitor
; instead of my callback
970A-   A9 4C       LDA   #$4C
970C-   8D 7E B7    STA   $B77E
970F-   A9 59       LDA   #$59
9711-   8D 7F B7    STA   $B77F
9714-   A9 FF       LDA   #$FF
9716-   8D 80 B7    STA   $B780
9719-   4C 00 B7    JMP   $B700

*BSAVE TRACE3A,A$9600,L$11C
*9600G
...reboots slot 6...
<beep>

*2436<B436.B535M
*C500G
...
]BSAVE BOOT2 B436,A$2436,L$100
]CALL -151

*B436<2436.2535M
```

```
*B436L

; probably an address ($FE) -> $B4F3
B436-   A9 F3       LDA   #$F3
B438-   85 FE       STA   $FE
B43A-   A9 B4       LDA   #$B4
B43C-   85 FF       STA   $FF

; probably a death counter
B43E-   A9 80       LDA   #$80
B440-   85 50       STA   $50

; turn on drive motor manually
B442-   AE E9 B7    LDX   $B7E9
B445-   BD 89 C0    LDA   $C089,X
B448-   BD 8A C0    LDA   $C08A,X

; if death counter hits 0, give up
B44B-   C6 50       DEC   $50
B44D-   F0 65       BEQ   $B4B4

; get next address field (still on T02)
B44F-   20 44 B9    JSR   $B944

; if that failed, give up
B452-   B0 60       BCS   $B4B4

; check for sector $0A (the unreadable
; sector on track $02!)
B454-   A5 2D       LDA   $2D
B456-   C9 0A       CMP   #$0A

; loop until found
B458-   D0 F1       BNE   $B44B
```

```
; look for $AD nibble
B45A-   A0 00        LDY   #$00
B45C-   BD 8C C0     LDA   $C08C,X
B45F-   10 FB        BPL   $B45C
B461-   88           DEY
B462-   F0 50        BEQ   $B4B4    ; die
B464-   C9 AD        CMP   #$AD
B466-   D0 F4        BNE   $B45C

; look for $E7 $E7 $E7 sequence
B468-   A0 00        LDY   #$00
B46A-   BD 8C C0     LDA   $C08C,X
B46D-   10 FB        BPL   $B46A
B46F-   88           DEY
B470-   F0 42        BEQ   $B4B4    ; die
B472-   C9 E7        CMP   #$E7
B474-   D0 F4        BNE   $B46A
B476-   BD 8C C0     LDA   $C08C,X
B479-   10 FB        BPL   $B476
B47B-   C9 E7        CMP   #$E7
B47D-   D0 35        BNE   $B4B4    ; die
B47F-   BD 8C C0     LDA   $C08C,X
B482-   10 FB        BPL   $B47F
B484-   C9 E7        CMP   #$E7
B486-   D0 2C        BNE   $B4B4    ; die

; waste some cycles to get out of sync
; with the "proper" start of nibbles
B488-   BD 8D C0     LDA   $C08D,X
B48B-   A0 10        LDY   #$10
```

```
; now start looking for nibbles that
; don't really exist (except they do,
; because we're out of sync and reading
; timing bits as data)
B48D-   BD 8C C0    LDA   $C08C,X
B490-   10 FB       BPL   $B48D
B492-   88          DEY
B493-   F0 1F       BEQ   $B4B4
B495-   C9 E7       CMP   #$E7
B497-   D0 F4       BNE   $B48D

; check for nibble sequence stored
; in reverse order at $B4FE, via ($FE)
B499-   A0 0F       LDY   #$0F
B49B-   BD 8C C0    LDA   $C08C,X
B49E-   10 FB       BPL   $B49B
B4A0-   D1 FE       CMP   ($FE),Y
B4A2-   D0 10       BNE   $B4B4   ; die
B4A4-   88          DEY
B4A5-   D0 F4       BNE   $B49B

; nibble check passed, turn off drive
; motor and exit gracefully (no flags)
B4A7-   BD 88 C0    LDA   $C088,X
B4AA-   60          RTS

; The Badlands (but the entry point is
; at $B4B4)
B4AB-   91 00       STA   ($00),Y
B4AD-   C8          INY
B4AE-   D0 FB       BNE   $B4AB
B4B0-   E6 01       INC   $01
B4B2-   D0 F7       BNE   $B4AB
```

```
; turn off drive motor
B4B4-   BD 88 C0    LDA   $C088,X
B4B7-   8A          TXA
B4B8-   A9 02       LDA   #$02
B4BA-   85 01       STA   $01

; check counter that was originally
; incremented during $B6B3 routine
B4BC-   AD 8A B7    LDA   $B78A
B4BF-   C9 18       CMP   #$18

; if it's set, you get a free pass,
; even though the nibble check failed
B4C1-   B0 15       BCS   $B4D8

; otherwise, munge the slot x16 into
; a $Cx byte
B4C3-   8A          TXA
B4C4-   4A          LSR
B4C5-   4A          LSR
B4C6-   4A          LSR
B4C7-   4A          LSR
B4C8-   18          CLC
B4C9-   69 C0       ADC   #$C0
B4CB-   8D D0 B4    STA   $B4D0

; and check a specific byte in the disk
; controller ROM routine
B4CE-   AD 04 C6    LDA   $C604
B4D1-   C9 A2       CMP   #$A2

; under some circumstances, you get
; another free pass
B4D3-   D0 03       BNE   $B4D8

; otherwise, jump to final memory wipe
B4D5-   4C AB B4    JMP   $B4AB
```

```
; free pass -- fiddle some I/O switches
; and exit gracefully
B4D8-   BD 8F C0    LDA   $C08F,X
B4DB-   A9 00       LDA   #$00
B4DD-   9D 8F C0    STA   $C08F,X
B4E0-   BD 8E C0    LDA   $C08E,X
B4E3-   BD 8C C0    LDA   $C08C,X
B4E6-   60          RTS
```

I have no idea what's going on here. I suspect the developers discovered that their copy protection didn't work on certain drives. But rather than give up, they added checks for the drive itself and let it go, relying on the structural protection alone. But on standard Apple drive controllers, a failed nibble check ends up at $B4AB and fills all of memory with value $A2, which, unsurprisingly, is a quotation mark.

The nibble check has no side effects. If I change the JSR at $B77E to a BIT, it will bypass the entire thing.

```
T00,S01,$7E change "20" to "2C"

]PR#6
...works...
```

Well, I suppose that's it.

Quod erat liberandum.

A 4am crack No. 305

EOF