

A Technical Overview of Commodore Copy Protection

Glenn Holmer (“ShadowM”)

www.lyonlabs.org/commodore/c64.html

World of Commodore Expo, 12/01/2007

Why Talk About This?

- These skills were a “black art” to begin with and are in danger of being forgotten
- There's a lot of myth and legend about this stuff, but the truth is out there
- Knowing how data are written helps to understand common disk errors
- It's really cool

What is Copy Protection?

- A software loader that is able to read non-standard data on the surface of the disk
- Used to keep users from making copies of software instead of buying it
- Usually used with encryption, undocumented opcodes, etc.
- I'll only discuss physical protection

What I'll Cover

- How “normal” data are physically encoded on the surface of the disk
- Types of copy protection (how protection is encoded on the disk)
- Disk programming (technique and examples)
- Samples (disassemblies of code from protected disks)

Disk Structure

- To understand copy protection, you have to understand how data are written to the disk
- Data are stored in sectors within tracks (what we see with a sector editor)
- Number of sectors varies by zone
- Head is moved from track to track by a stepper motor (1/2 track at a time)

Data On the Disk

- Disk is covered in iron oxide (rust); write head polarizes magnetic domains to create information
- '1' detected by change in polarity, '0' by no change within a certain time
- *What about data with many '0' bits?*
- How do we tell what track we're on, or where a sector begins? (need a unique marker that's not '1' or '0')

The Sync Mark

- Ten or more '1' bits in a row are detected in hardware (sets bit 7 of register \$1C00 in the drive)
- Used to mark the beginning of each sector's header and data blocks
- Gaps between blocks give the hardware time to “switch gears” before the next sync mark is written or read
- *What about data with many '1' bits?*

GCR Encoding

- Every nybble (4 bits) converted to 5 bits of GCR
- Conversion table set up so that no more than eight consecutive one bits or two consecutive zero bits are ever written to the disk as data
- Solves both issues:
 - ▶ too many zeros cause timing errors
 - ▶ too many ones make a sync

Block Structure

Header Block

sync mark
header ID (\$08)
checksum
sector no.
track no.
disk ID 2
disk ID 1
\$0F byte
\$0F byte
header gap (\$55)

Data Block

sync mark
data ID (\$07)
data (256 bytes)
data checksum
\$00 byte
\$00 byte
tail gap (\$55)

Header Block

OF HEADERS: \$26

TRACK:

18

BIT RATE:

\$40

SECTOR: 01				HEADER \$02				HEADER				
52	55	25	29	72	73	9E	E5	08	02	00	12	44
55	D7	25	2D	CB	52	96	FA	07	12	01	41	00
52	55	35	2D	72	73	9E	E5	08	03	01	12	44
55	D4	AA	D5	2B	5A	D4	A7	07	00	FF	81	11
52	54	A5	49	72	73	9E	E5	08	00	02	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	54	B5	4D	72	73	9E	E5	08	01	03	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	55	65	39	72	73	9E	E5	08	06	04	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	55	75	3D	72	73	9E	E5	08	07	05	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	54	E5	59	72	73	9E	E5	08	04	06	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	54	F5	5D	72	73	9E	E5	08	05	07	12	44
55	DD	B5	2D	4B	52	D4	B5	07	4B	01	01	01
52	55	A5	25	72	73	9E	E5	08	0A	08	12	44

Header Block Detail

HEADER: \$02 SYNC: 0005 LENGTH: \$0013

TRACK:

18

BIT RATE:

\$40

POS: \$0000

5255352D72739EE55555
555555555555555574..

0803011244540F0F
0F0F0F0F0F0F0DFF

Data Block

OF HEADERS: \$26

TRACK:

18

BIT RATE:

\$40

SECTOR: 01								HEADER \$03		DATA				
52	55	25	29	72	73	9E	E5			08	02	00	12	44
55	D7	25	2D	CB	52	96	FA			07	12	01	41	00
52	55	35	2D	72	73	9E	E5			08	03	01	12	44
55	D4	AA	D5	2B	5A	D4	A7			07	00	FF	81	11
52	54	A5	49	72	73	9E	E5			08	00	02	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	54	B5	4D	72	73	9E	E5			08	01	03	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	55	65	39	72	73	9E	E5			08	06	04	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	55	75	3D	72	73	9E	E5			08	07	05	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	54	E5	59	72	73	9E	E5			08	04	06	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	54	F5	5D	72	73	9E	E5			08	05	07	12	44
55	DD	B5	2D	4B	52	D4	B5			07	4B	01	01	01
52	55	A5	25	72	73	9E	E5			08	0A	08	12	44

100

HEADER: \$03 SYNC: 0006 LENGTH: \$0154

TRACK: 18 BIT RATE: 540

POS: \$0000

[illegible]

100

HEADER: \$03 SYNC: 0006 LENGTH: \$0154

TRACK: 18 BIT RATE: 540

POS: \$014F

[illegible]

Types of Copy Protection

- simple errors
- unusual header/gap data
- extra tracks (36-40)
- no-sync track
- wrong density
- fat tracks
- spiral tracking
- custom formats

GEOS Tail Gap

HEADER: \$01 SYNC: 0005 LENGTH: \$014C

TRACK: 18 BIT RATE: \$40

POS: \$0148

52D55AFABB52D6BAD6B5	010FFEFB0111FFFF
52DF3BE6F3BBACFB774A	0153797374656DA0
D2B4AD2B4AD2B4AD2B4A	A0A0A0A0A0A0A0A0
D2B4AD29CD76B4A9C9CB	A0A0A04C4AA03241
D2B4AD2B4A5CD4975DCB	A0A0A0A013084745
755F392AD6B56F2B76CB	4F5320666F726D61
BB04A7DA6B97A6A7492A	742056312E304280
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5294A5294A	0000000000000000
5294A5294A5555556755	0000000000F0F09AF
5555.....	0F0F0F0F.....

100

100

Defender of the Crown #2

TRACK:	14	RATE:	40	NORM	LENGTH:	????
SRC:	00 00	DST:	00 00	SPEED:	????	
GAP:	00 00 00 00 00	SYNC:	ON			
HDR:	00 00 00 00 00 00 00 00 00	FILL:	55			

: 03C0	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03C8	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03D0	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03D8	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03E0	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03E8	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03F0	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 03F8	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0400	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0408	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0410	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0418	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0420	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0428	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0430	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0438	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0440	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0448	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0450	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
: 0458	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33

ENTER COMMAND: <←> TO EXIT

Defender of the Crown #3

TRACK:	14	RATE:	40	NORM	LENGTH:	????
SRC:	00 00	DST:	00 00	SPEED:	????	
GAP:	00 00 00 00 00	SYNC:	ON			
HDR:	00 00 00 00 00 00 00 00 00	FILL:	55			

:0460	99	99	99	99	99	99	99	99	.	.	.	.	.	.
:0468	99	99	99	99	99	99	99	99	.	.	.	.	.	.
:0470	99	99	99	99	99	99	99	99	.	.	.	.	.	.
:0478	99	99	99	99	99	99	99	99	.	.	.	.	.	.
:0480	9C	B4	AD	2D	2D	2D	2D	2D	.	.	-	-	-	-
:0488	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:0490	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:0498	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04A0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04A8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04B0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04B8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04C0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04C8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04D0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04D8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04E0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04E8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04F0	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-
:04F8	2D	2D	2D	2D	2D	2D	2D	2D	-	-	-	-	-	-

ENTER COMMAND: <←> TO EXIT

The Hardware

- Both the '64 and the 1541 have CPU, RAM, ROM, and I/O chips
- Commodore drives are “intelligent peripherals”; half the operating system is in the drives (no disk buffers in main memory)
- Code can be sent to the drive and run there: *this is the heart and soul of copy protection*

1541 Memory Map

- \$0000-\$07FF: RAM
 - ▶ \$0000-\$0015: job queue
 - ▶ \$0100-\$01FF: CPU stack
 - ▶ \$0200-\$0229: command buffer
 - ▶ \$0300-\$07FF: data buffers
- \$1800-\$180E: VIA #1 (serial controller)
- \$1C00-\$1C0E: VIA #2 (disk controller)
- \$C100-\$FFFFFF: DOS ROM

Disk Programming

- Direct-access (“channel commands”): M-R, M-W, B-E, etc. -- some are used in conjunction with “#” file
- Placing commands in the drive's job queue and letting the controller execute them
- Custom code executed in the drive: requires knowledge of hardware registers and DOS internals

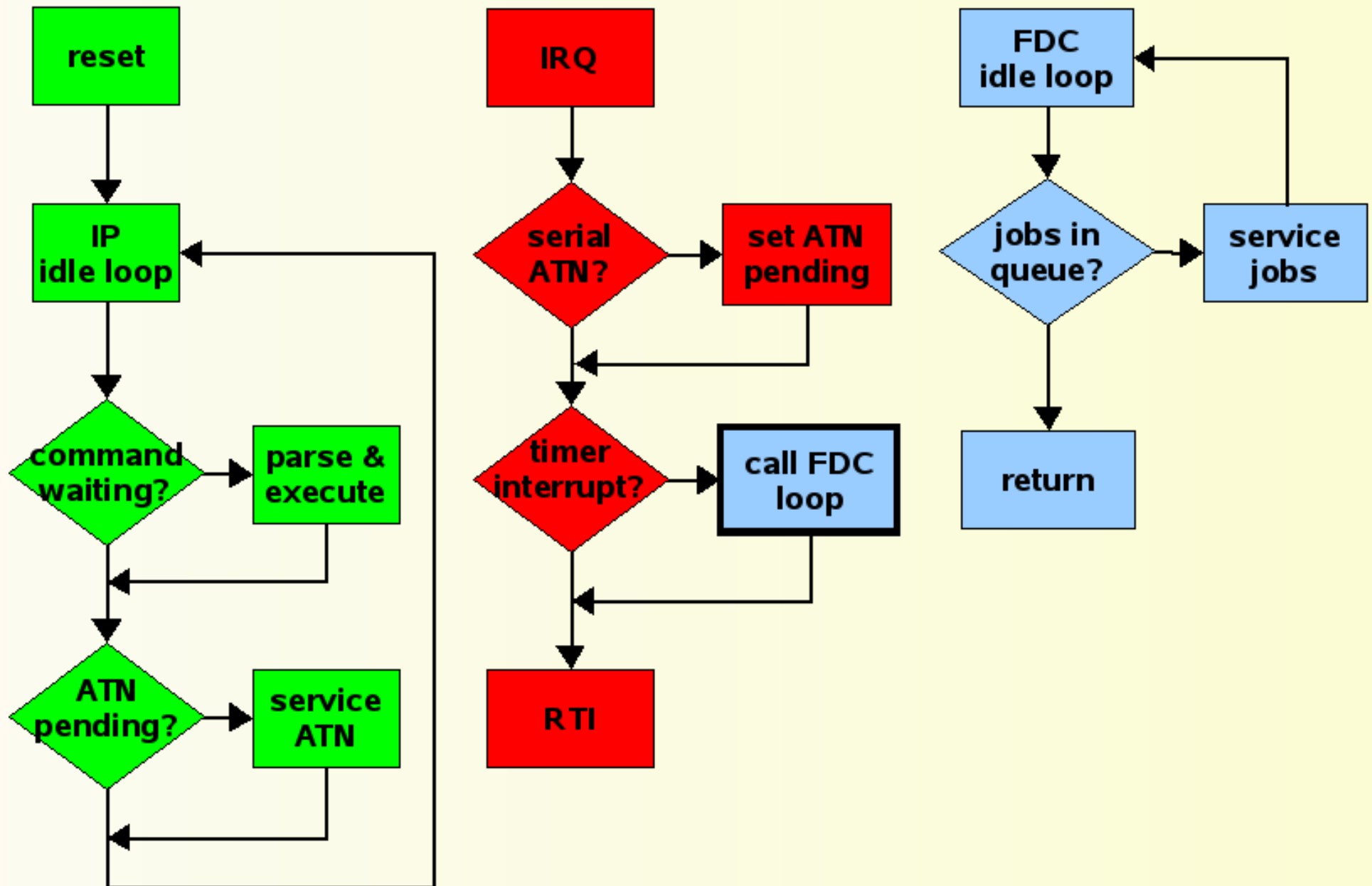
Direct Access

- Channel commands: require opening “#” file (U1, U2, B-E, B-P, etc.)
 - ▶ B-E <channel> <unit> <track> <sector>
 - ▶ B-P <channel> <position>
- Direct commands
 - ▶ M-R <lo> <hi> <no. bytes>
 - ▶ M-W <lo> <hi> <no. bytes> <data>
 - ▶ M-E <lo> <hi>

IP/FDC Processing

- Older Commodore drives had multiple CPUs; 1541 emulates with interrupts
- IP (interface processor) mode: communication with the '64, file handling (sets up FDC jobs)
- FDC (floppy disk controller) mode: triggered every 10 ms by an interrupt

DOS Flowchart



Working the Job Queue

	\$0300	\$0400	\$0500	\$0600	\$0700
job	\$00	\$01	\$02	\$03	\$04
track	\$06	\$08	\$0A	\$0C	\$0E
sector	\$07	\$09	\$0B	\$0D	\$0F

\$80 READ
\$90 WRITE
\$A0 VERIFY
\$B0 SEEK
\$C0 BUMP
\$D0 JUMP
\$E0 EXECUTE

Job Queue Example

This code reads an illegal track into the buffer at \$0300.

```
        LDA #$25 ;track 37
        STA $06
        LDA #$0F ;sector 15
        STA $07
        LDA #$80 ;READ job code
        STA $00  ;put job code for FDC
loop    LDA $00
        BMI loop ;wait for FDC to clear
```

Down on the Bare Vinyl

- Requires knowledge of hardware registers and DOS internals (API and zero-page variable usage)
- Must stay aware of FDC/IP interface
- There is no safety net; anything the hardware is physically capable of can be requested!

Disk Controller Registers

\$1C00 (data port B)

- bit 7: sync detect
- bits 5,6: density
- bit 4: WP sense
- bit 3: LED on/off
- bit 2: motor on/off
- bits 0,1: head phase

\$1C03 (data direction: \$00 = input, \$FF = output)

\$1C0C (peripheral control register)

	<u>7</u>	<u>6</u>	<u>5</u>	<u>4</u>	<u>3</u>	<u>2</u>	<u>1</u>	<u>0</u>
read mode:	1	1	1	x	1	1	1	x
write mode:	1	1	0	x	1	1	1	x

(bits 1, 2, and 3 select overflow enable to CPU)

Reading/Writing Sync

read sync:

```
loop BIT $1C00  
      BMI loop  
      CLV  
      LDA $1C01
```

write sync:

```
LDA #$FF  
LDX #$05  
STA $1C01  
CLV  
loop BVC loop  
      CLV  
      DEX  
      BNE loop
```

Reading/Writing Data

read data:

```
LDA $1C0C
ORA #$E0
STA $1C0C ;PCR
LDA #$00
STA $1C03 ;DDR
CLV
loop BVC loop
CLV
LDA $1C01
STA $data
```

write data:

```
LDX #$FF
STA $1C03 ;DDR
LDA $1C0C
AND #$1F
ORA #$C0
STA $1C0C ;PCR
LDA $data
STA $1C01
CLV
loop BVC loop
CLV
```


Setting Density

set density for tracks 18-24:

```
LDA $1C00 ;density is bits 5, 6  
AND #$9F  ;mask off density bits  
ORA #$40   ;set density  
STA $1C00
```

density zone bit values:

\$60	tracks	1-17	(sectors 0-20)
\$40	tracks	18-24	(sectors 0-18)
\$20	tracks	25-30	(sectors 0-17)
\$00	tracks	31-35	(sectors 0-16)

Stepping the Head

```
LDX $1C00 ;head phase is bits 0, 1
INX      ;step inward
TXA
AND #$03 ;mask phase bits only
STA temp
LDA $1C00
AND #$FC ;mask off phase bits
ORA temp ;apply new phase bits
STA $1C00
```

DEX on second line to step outward.
Make sure to allow the head time to settle.

Abacus Assembler loader

loader pseudocode:

- 1) open 2,8,2,"#2"
(buffer 2 is at \$0500 in drive)
- 2) open 1,8,15,"B-E 2 0 18 5"
(block-execute track 18, sector 5)
- 3) close 2
- 4) open 2,8,2,"#0"
(buffer 0 is at \$0300 in drive)
- 5) print #1,"B-P 2 0"
- 6) input #2 (256 bytes)
- 7) if 27 sectors have been read, goto 9
- 8) print #1,"U3" (execute \$0500), goto 5
- 9) decrypt loaded code with final sector

Abacus drive code #1

```
LDA $053A
STA $80      ;track
LDA $053C
STA $81      ;sector
LDA #0       ;buffer 0
JSR $D6D3    ;set up job queue
LDX #0
LDA #$80     ;READ job code
JSR $D57D    ;to job queue
JSR $D599    ;wait for completion
```

```
:053A 21 00 0F 0F 0A 02 0E 08
:0542 0C 06 0B 05 09 03 0D 07
:054A 80
```

Abacus drive code #2

```
INC $053B ;sector counter
LDX $053B
LDA $053D,X
BMI done ;read all sectors?
STA $053C ;set up next read
RTS
done LDA $053D ;reset first sector no.
STA $053C
LDA #0 ;reset sector counter
STA $053B
INC $053A ;increment track number
INC $053A ;increment track number
RTS
```

Zorkquest drive code #1

```
        LDX #2          ;counter
        STX $08
        LDA #$23        ;track 35
        STA $06
redo    LDA #$10        ;sector 16
        STA $07
read    LDA #$80        ;READ job code
        STA $00
        JSR $046F       ;wait for completion
        CMP #$01        ;completed without error?
        BEQ next
        STA $09         ;save error code
next    DEC $07         ;another sector
        BPL read
```

Zorkquest drive code #2

```
        JSR step    ;step head in one phase
        DEC $08     ;(do three times)
        BPL redo
        LDY #2
out     JSR $0476    ;restore head position
        DEY
        BPL out
        LDA $09
        BNE done
        LDA #$FF
        STA $01FF   ;success flag for loader
        JMP done    ;erase this code, return
```

Zorkquest drive code #3

```
step LDX $1C00 ;head phase is bits 0, 1
      INX      ;step inward
      TXA
      AND #$03 ;mask phase bits only
      STA $44
      LDA $1C00
      AND #$FC ;mask off phase bits
      ORA $44   ;apply new phase bits
      STA $1C00
      LDA #$D0
      STA $1805 ;set timer
wait  BIT $1805 ;allow head to settle
      BMI wait
      RTS
```


Tools for Exploration

- Simple tools: disk monitors like Disk/Extramon and SuperSnapshot (to read/write drive memory)
- More powerful tools: Hacker's Utility Kit, Maverick GCR Editor, Datel utilities (capable of directly editing the disk at the GCR level)
- Hardware assistance: RAMBOard and friends

Resources

- Inside Commodore DOS, Immers & Neufeld
(excellent commentary, no DOS listings)
<http://project64.ath.cx/misc/index.html> (item 16)
- Anatomy of the 1541, Abacus Software
(less commentary, complete DOS listings)
- CSM Program Protection Manual, vol. II
(fills in a lot of blanks, has good sample code)
- Kracker Jax Revealed (three booklets)
detailed explanation of RAPIDLOK and V-MAX
<http://members.aol.com/c64dir/kjtr/index.htm>