

*DDW*

# DOS 3.2

Flow of Control  
Anotated Disassemblies  
Notes

By Don D Worth  
Victor Tolomei

**COPY 1 OF ~~6~~8**

**DO NOT REPRODUCE**



DOS BOOT FLOW OF CONTROL (VERSION 3.2)

1) USER ENTERS "s^P" TO MONITOR OR "PR#s" TO BASIC (WHERE s IS THE DISK SLOT, 1-7, AND ^P IS CONTROL-P)

(a) MONITOR ROUTINE GETS CONTROL, CONVERTS s TO ADDRESS OF ROM ON BOARD IN SLOT s, \$Cs00. THIS ADDRESS IS PLACED AT CSWL (\$36) AS THE "OUTPUT INTERCEPT ADDRESS". WHENEVER OUTPUT IS TO BE PLACED ON THE VIDEO SCREEN, THE ROUTINE AT \$Cs00 WILL RECEIVE CONTROL.

(b) STANDARD MONITOR PROCESSING CONTINUES. WHEN THE PROMPT (EITHER ">" OR "\*") IS TO BE DISPLAYED, ...

2) THE DISK CONTROLLER CARD ROM BOOT PROGRAM AT \$Cs00 BEGINS EXECUTION.

(a) LET'S CALL THIS 256-BYTE ROM PROGRAM "BOOT PHASE 1". BOOT PHASE 1 TURNS DRIVE 1 IN SLOT s ON AND "RECALIBRATES" THE DISK ARM (MOVES IT TO TRACK 0 AT THE OUTMOST CIRCUMFERENCE). THIS CREATES THE CLICKING SOUND FOR A FEW SECONDS.

(b) PHASE 1 THEN READS THE FIRST 256-BYTE SECTOR ON THAT TRACK (TRACK 0, SECTOR 0) INTO RAM PAGE 3 (\$3000-\$3FF).

(c) IF THIS IS SUCCESSFUL (A CHECKSUM IS CALCULATED AND COMPARED WITH ONE AT \$3000), THE RAM BOOT PHASE 2 PROGRAM HAS BEEN LOADED. PHASE 1 (AT \$C5F9) JUMPS TO THIS ROUTINE TO CONTINUE DOS BOOT...

(3) RAM BOOT PHASE 2 AT \$301 BEGINS EXECUTION.

(a) USING THE SUBROUTINE LOCATED IN ROM BOOT PHASE 1 AT \$C55D WHICH READS DISK SECTORS ON TRACK 0, RAM BOOT PHASE 2 READS THE NEXT 9 SECTORS ON TRACK 0 (TRACK 0 SECTOR 1 TO TRACK 0 SECTOR 9) TO RAM PAGES \$36 TO \$3F (\$36000-\$3FFFF) FOR A "MASTER" DISKETTE OR 16K "SLAVE", TO \$76000-\$7FFFF FOR A 32K SLAVE, OR TO \$B6000-\$BFFFF FOR A 48K SLAVE

(b) THESE LOCATIONS \$76000-\$7FFFF PERFORM TWO FUNCTIONS

- IT WILL EVENTUALLY BE THE SECOND HALF OF DOS, INCLUDING "RWTS"

(THE "READ-WRITE-TRACK-SECTOR" PROGRAM)

- THE ROUTINE AT \$77000 IS NEEDED FOR FINAL LOADING



(C) RAM PHASE 2 (AT \$343) JUMPS TO \$2700 TO FINISH BOOTING ...

(L) RAM BOOT PHASE 3 AT \$2700 BEGINS EXECUTION

(a) THIS WILL LOAD ALL REMAINING PARTS OF DOS WHICH ARE DEPENDENT ON DOS RELEASE (VERSION NUMBER) AND WHETHER DISKETTE IS A "SLAVE" OR A "MASTER"

(b) IF MASTER: USING THE RWTS ROUTINE AT \$2000, RAM BOOT PHASE 3 LOADS THE FIRST HALF OF DOS INTO RAM. THIS COMPRISES THE NEXT 27 SECTORS ON THE DISK (TRACK 0 SECTOR \$A TO TRACK 2 SECTOR \$D). THESE ARE READ INTO RAM PAGES \$1B TO \$35 (\$1B00 TO \$35FF), BUMPING UP AGAINST DOS 2ND HALF AT \$3600 LOADED IN STEP (3) ABOVE. DOS IS NOW COMPLETELY LOADED. BUT, THE ACTUAL DOS BEGINS AT \$1D00 AND EXTENDS TO \$3FFF, SET UP TO RUN AS IS ON A 16K MACHINE ONLY. A "MASTER" DOS IS TO RUN ON ANY SIZE APPLE (ABOVE 16K SO A "DOS RELOCATOR" PROGRAM WAS INCLUDED AT \$1B00-\$1CFF ABOVE THIS DOS. ROM BOOT PHASE 3 (AT \$3747)

JUMPS TO \$1B03 (RELOCATOR ENTRY POINT) TO CREATE A DOS FOR THE RIGHT SIZE MACHINE, SEE STEP (5).

(C) IF SLAVE! FOR A SLAVE DISKETTE, RAM BOOT PHASE 3 LOADS A PRE-RELOCATED FIRST HALF OF DOS (CURRENT AT THE TIME OF THE DISK "INIT") DIRECTLY INTO \$1D00-\$35FF (16K), OR \$5D00-\$75FF (32K), OR \$9D00-\$B5FF (48K). THIS IS OBTAINED FROM THE NEXT 25 DISK SECTORS TRACK 0 SECTOR \$A TO TRACK 2 SECTOR \$B. THE RELOCATOR IS NOT INCLUDED AND NOT USED, SINCE THIS DOS IS ALREADY RELOCATED. (NOTE: IT MAY BE THE WRONG DOS FOR THE SIZE MACHINE). HERE RAM BOOT PHASE 3 AT \$2747 DOES NOT JUMP TO \$1B03 RELOCATOR BUT DIRECTLY TO DOS COLDSTART PROCESSING AT \$2D84. BOOT IS COMPLETE, SKIP TO STEP (6).



(5) DOS RELOCATOR AT \$1B00 BEGINS EXECUTION (MASTER DISKETTE ONLY).

(a) RELOCATOR IS IN RAM \$1B00-\$1CFF. UNRELOCATED DOS IS IN RAM \$1D00-\$3FFF. RELOCATOR FIRST FINDS THE LAST PAGE WHERE RAM ACTUALLY EXISTS (3F ON 16K, \$7F ON 32K, \$BF ON 48K MACHINES).

(b)

(c) TO FORCE DOS TO BE AT THE HIGHEST RAM LOCATIONS POSSIBLE, \$1D00-\$3FFF IS RELOCATED TO \$5D00-\$7FFF (32K) OR \$9D00-\$BFFF (48K). IF 16K MACHINE IS USED, RELOCATOR HAS NOTHING TO DO BUT JUMP (AT \$1B61) TO DOS COLDSTART. (SEE STEP (6)).

(d) (MACHINES LARGER THAN 16K ONLY): USING A TABLE OF ADDRESS RANGES (AT \$1C29), RELOCATOR FIRST RELOCATES ALL 2-BYTE DOS ADDRESS CONSTANTS.

(e) USING ANOTHER SIMILAR TABLE (AT \$1C5A) RELOCATOR RELOCATES ALL 3-BYTE G502 INSTRUCTIONS WHOSE ABSOLUTE ADDRESSES ARE WITHIN PAGES \$1D-\$3F.

(f) RELOCATOR COPIES THE NEW DOS (\$1D00-\$3F) TO THE TOP OF RAM WHERE IT SHOULD RESIDE (\$5D00-\$7FFF OR \$9D00-\$EFFF)

(g) RELOCATOR JUMPS (AT \$1C25) TO DOS COLDSTART TO INITIALIZE DOS,

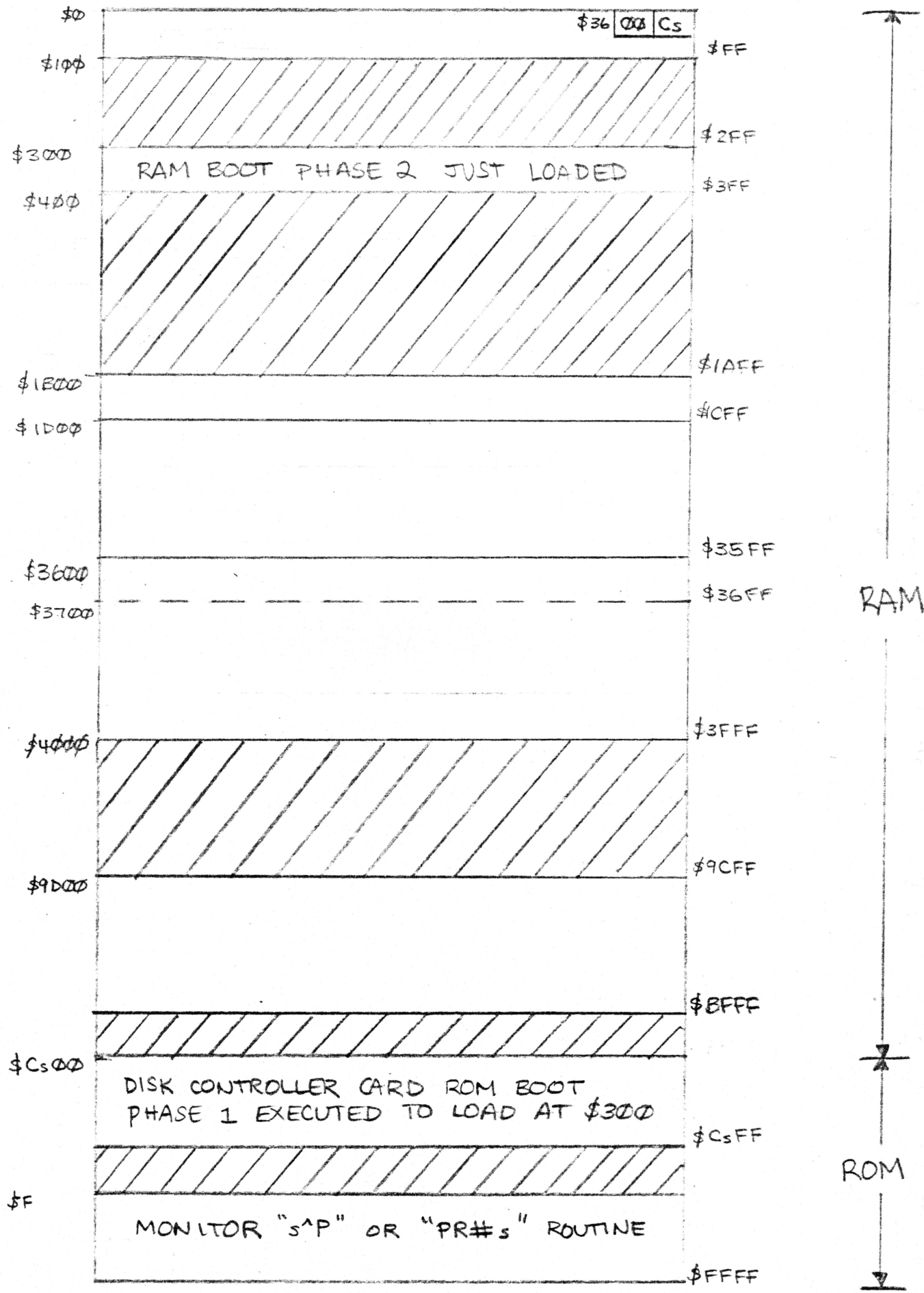
(6) DOS COLDSTART AT \$2D84 BEGINS EXECUTION

(a) "COLDSTART" INITIALIZES THE NOW BOOTED AND RELOCATED OPERATING SYSTEM

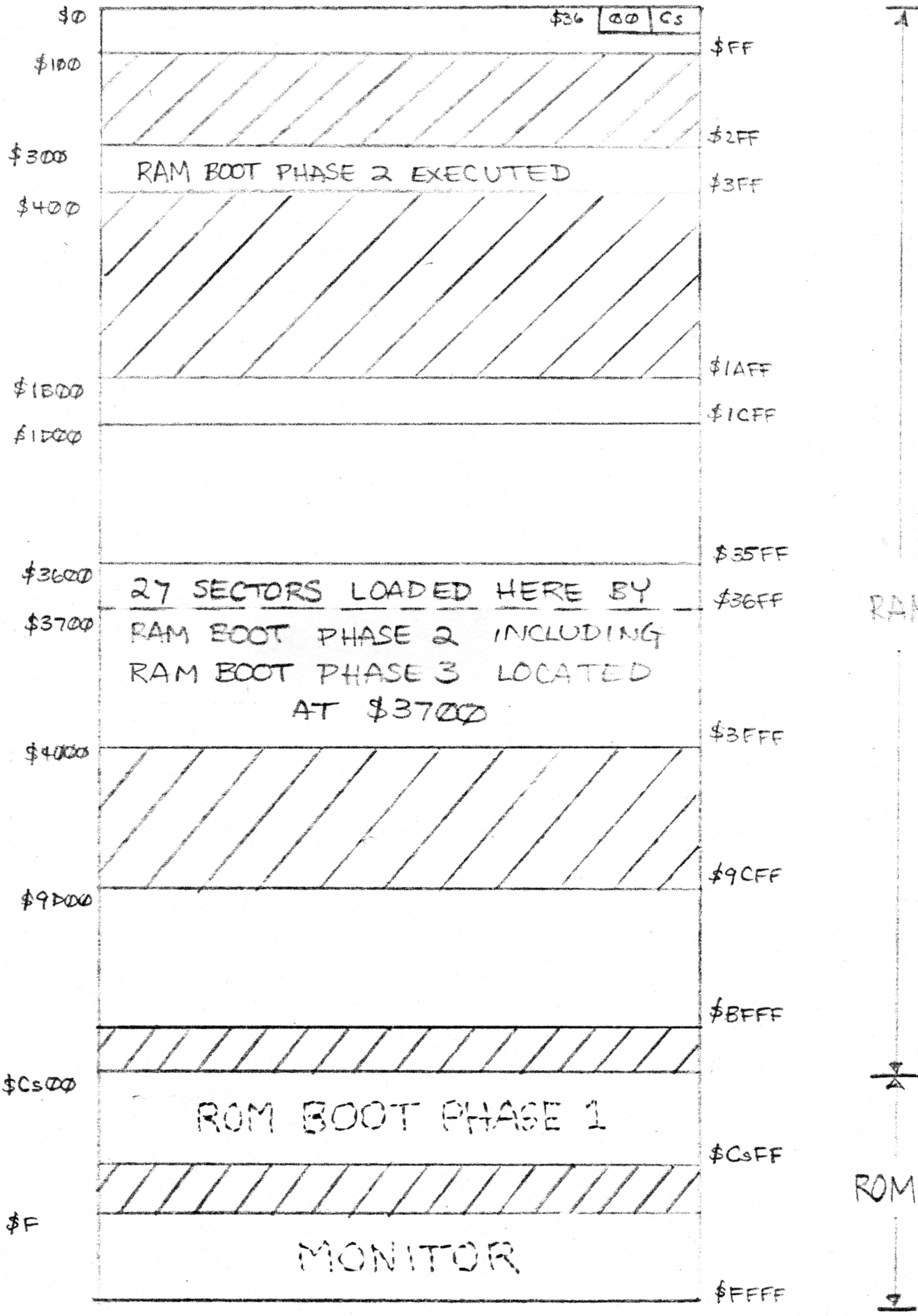
(b) INITIALIZATION OCCURS AND PROPER BASIC (INTEGER OR FLOATING, ROM OR ROM CARD) IS GIVEN CONTROL



MEMORY MAP ON 48K APPLE AFTER STEPS (1) AND (2) OF A DOS MASTER DISKETTE BOOT

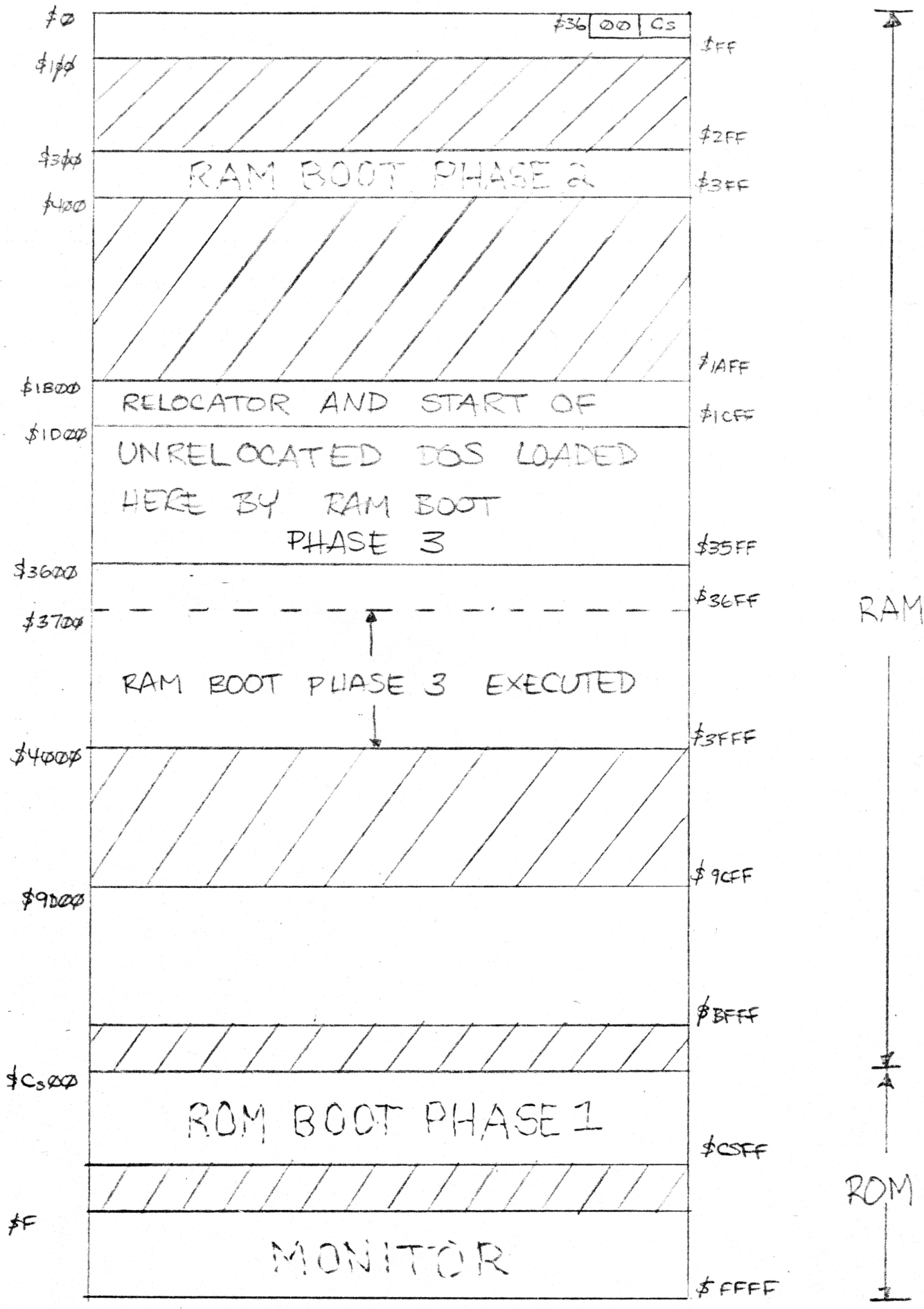


MEMORY MAP ON 48K APPLE AFTER STEP (3) OF DOS MASTER BOOT.

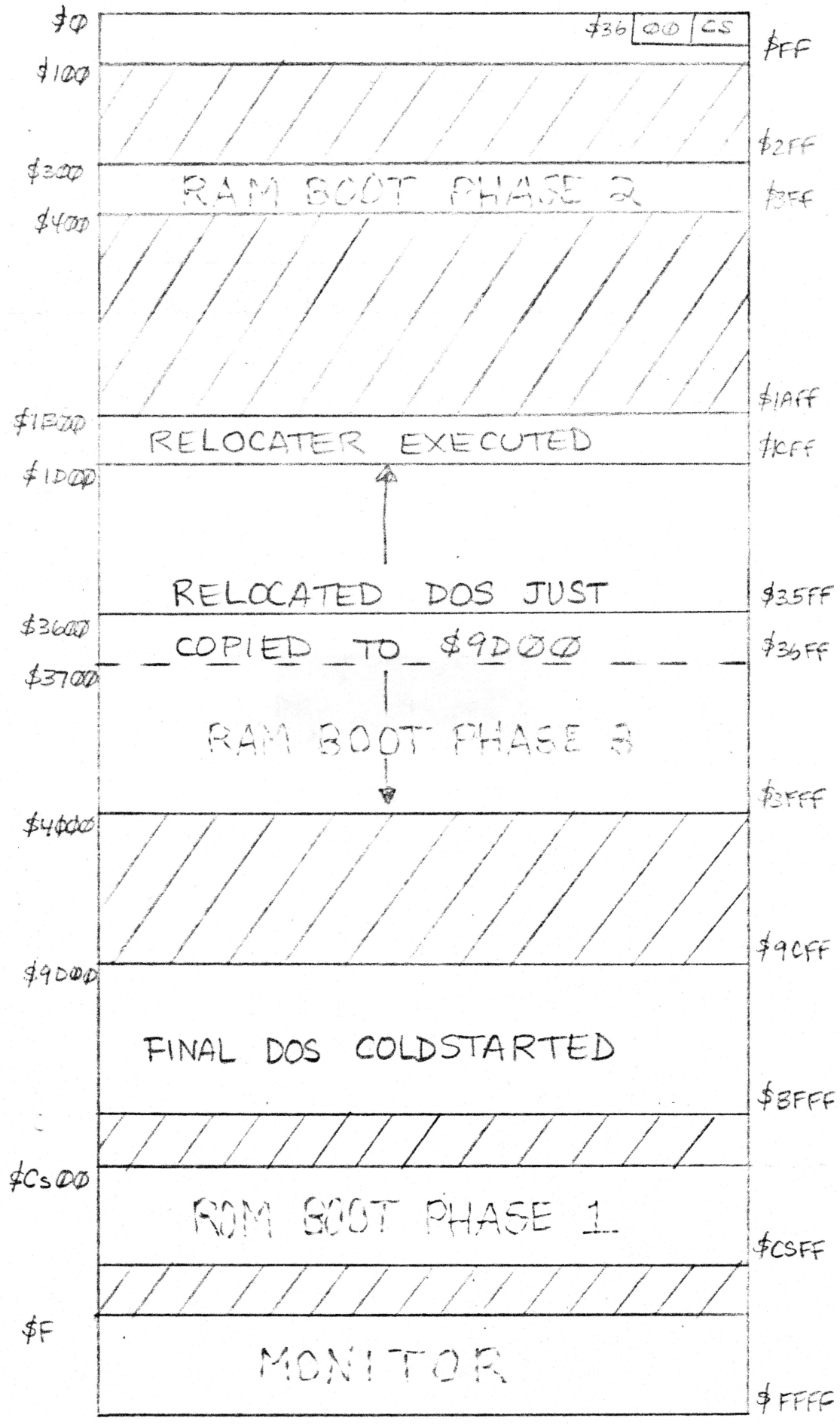




MEMORY MAP ON 48K APPLE AFTER STEP (4) OF DOS MASTER BOOT



MEMORY MAP ON 48K APPLE AFTER STEP (5) & (6) OF DOS MASTER BOOT



RAM

ROM

DOS DISK II ROM BOOTSTRAP

(\$Cs00-\$CsFF, s=slot)

# DOS DISK II ROM BOOTSTRAP 12

C700-	A2 20	LDX	##20
C702-	A0 00	LDY	##00
C704-	A9 03	LDA	##03
C706-	85 3C	STA	\$3C
C708-	18	CLC	
C709-	88	DEY	
C70A-	98	TYA	
C70B-	24 3C	BIT	\$3C
C70D-	F0 F5	BEQ	\$C704
C70F-	26 3C	ROL	\$3C
C711-	90 F8	BCC	\$C70B
C713-	C0 D5	CPY	##D5
C715-	F0 ED	BEQ	\$C704
C717-	CA	DEX	
C718-	8A	TXA	
C719-	99 00 08	STA	\$0800, Y
C71C-	D0 E6	BNE	\$C704
C71E-	20 58 FF	JSR	\$FF58
C721-	BA	TSX	
C722-	BD 00 01	LDA	\$0100, X
C725-	48	FHA	
C726-	0A	ASL	
C727-	0A	ASL	
C728-	0A	ASL	
C729-	0A	ASL	
C72A-	85 2B	STA	\$2B
C72C-	AA	TAX	
C72D-	A9 D0	LDA	##D0
C72F-	48	FHA	
C730-	BD 8E C0	LDA	\$C08E, X
C733-	BD 8C C0	LDA	\$C08C, X
C736-	BD 8A C0	LDA	\$C08A, X
C739-	BD 89 C0	LDA	\$C089, X
C73C-	A0 50	LDY	##50
C73E-	BD 80 C0	LDA	\$C080, X
C741-	98	TYA	
C742-	29 03	AND	##03
C744-	0A	ASL	
C745-	05 2B	ORA	\$2B
C747-	AA	TAX	
C748-	BD 81 C0	LDA	\$C081, X
C74B-	A9 56	LDA	##56
C74D-	20 A8 FC	JSR	\$FCAS
C750-	88	DEY	
C751-	10 EB	BPL	\$C73E
C753-	A9 03	LDA	##03
C755-	85 27	STA	\$27
C757-	A9 00	LDA	##00
C759-	85 26	STA	\$26
C75B-	85 3D	STA	\$3D

DYNAMICALLY BUILD TRANSLATE TABLE FOR DISK CODES AT HIGH END OF PAGE 8 (\$8AA-\$8FF)

EG: +A8 = 00  
AD = 01  
AE = 02  
AF = 03  
B5 = 04  
B6 = 05  
etc.

(RTS)  
GET RETURN ADDRESS FROM STACK TO DETERMINE THE SLOT NUMBER OF THIS DISK UNIT.

SAVE SP (EG: 70)  
PRETEND C75D SUBROUTINE ENTERED VIA JSR. EXIT TO C7D1  
CLEAR DISK LATCHES (READ MODE)  
SELECT DRIVE 1  
TURN IT ON

RECAL DISK ARM TO TRACK 0

(DELAY)

READ SECTOR 0 (TRACK 0) TO LOCATION \$300  
(IMPLIED JSR)

LOCATE & READ SECTOR

C75D-	18	CLC	
C75E-	08	PHP	
C75F-	BD 8C C0	LDA	\$C08C, X
C762-	10 FB	BPL	\$C75F
C764-	49 D5	EOR	##D5
C766-	D0 F7	BNE	\$C75F
C768-	BD 8C C0	LDA	\$C08C, X
C76B-	10 FB	BPL	\$C768
C76D-	C9 AA	CMP	##AA
C76F-	D0 F3	BNE	\$C764
C771-	EA	NOF	

READ DISK LOOKING FOR (D5) HEADER

NEXT FIND (AA)

```

C772- BD 8C C0 LDA $C08C, X ←
C775- 10 FB BPL $C772 ←
C777- C9 B5 CMP ##B5
C779- F0 09 BEQ $C784 ←
C77B- 28 PLP
C77C- 90 DF BCC $C75D ←
C77E- 49 AD EOR ##AD
C780- F0 1F BEQ $C7A1 ←
C782- D0 D9 BNE $C75D ←
C784- A0 03 → LDY ##03
C786- 84 2A STY $2A
C788- BD 8C C0 LDA $C08C, X ←
C78B- 10 FB BPL $C788 ←
C78D- 2A ROL
C78E- 85 3C STA $3C
C790- BD 8C C0 LDA $C08C, X ←
C793- 10 FB BPL $C790 ←
C795- 25 3C AND $3C
C797- 88 DEY
C798- D0 EE BNE $C788 ←
C79A- 28 PLP
C79B- C5 3D CMP $3D
C79D- D0 BE BNE $C75D ←
C79F- B0 BD BCS $C75E ←
C7A1- A0 9A LDY ##9A ←
C7A3- 84 3C STY $3C
C7A5- BC 8C C0 LDY $C08C, X ←
C7A8- 10 FB BPL $C7A5 ←
C7AA- 59 00 08 EOR $0800, Y
C7AD- A4 3C LDY $3C
C7AF- 88 DEY
C7B0- 99 00 08 STA $0800, Y
C7B3- D0 EE BNE $C7A3 ←
C7B5- 84 3C STY $3C ←
C7B7- BC 8C C0 LDY $C08C, X ←
C7BA- 10 FB BPL $C7B7 ←
C7BC- 59 00 08 EOR $0800, Y
C7BF- A4 3C LDY $3C
C7C1- 91 26 STA ($26), Y
C7C3- C8 INY
C7C4- D0 EF BNE $C7B5 ←
C7C6- BC 8C C0 LDY $C08C, X ←
C7C9- 10 FB BPL $C7C6 ←
C7CB- 59 00 08 EOR $0800, Y
C7CE- D0 8D BNE $C75D ←
C7D0- 60 RTS

```

IF NEXT IS **(B5)** THIS IS A SECTOR ADDRESS  
(WHICH DO WE WANT?)

IF **(AD)** THIS IS SECTOR DATA  
SECTOR ADDRESS - WHICH SECTOR?

READ VOL#, TRK#, SECTOR #  
(STORED AS DOUBLE BYTES OF ALTERNATING BITS)  
(3 ITEMS)

IS THIS THE SECTOR?  
NO, KEEP LOOKING AT SECTOR ADDRESSES  
YES, FIND SECTOR DATA NOW  
SECTOR DATA

FILL 153 BYTE SECONDARY DATA BUFFER AT \$800

FILL 256 BYTE PRIMARY DATA BUFFER AT [\$26, \$27]

GET CHECKSUM  
VALID? (NO I/O ERROR)  
EXIT SUBROUTINE

```

C7D1- A8 TAY
C7D2- A2 00 LDX ##00 ←
C7D4- B9 00 08 LDA $0800, Y ←
C7D7- 4A LSR
C7D8- 3E CC 03 ROL $03CC, X
C7DB- 4A LSR
C7DC- 3E 99 03 ROL $0399, X
C7DF- 85 3C STA $3C
C7E1- B1 26 LDA ($26), Y
C7E3- 0A ASL
C7E4- 0A ASL
C7E5- 0A ASL
C7E6- 05 3C ORA $3C
C7E8- 91 26 STA ($26), Y
C7EA- C8 INY
C7EB- E8 INX
C7EC- E0 33 CPX ##33

```

(RESUME ROM BOOT)

000 A1 A2 A3 A4 A5  
0000 A1 A2 A3 A4 C=A5  
00 F1 F2 F3 F4 F5 A5  
0000 0 A1 A2 A3 C=A4  
00 E1 E2 E3 E4 E5 A4

000 D1 D2 D3 D4 D5

D1 D2 D3 D4 D5 A1 A2 A3

DO 3 SEGMENTS OF 51

MERGE DATA BITS FROM SECONDARY TO PRIMARY BUFFER



C7EE-	D0 E4	BNE	\$C7D4
C7F0-	C6 2A	DEC	\$2A
C7F2-	D0 DE	BNE	\$C7D2
C7F4-	CC 00 03	CPY	\$0300
C7F7-	D0 03	BNE	\$C7FC
C7F9-	4C 01 03	JMP	\$0301
C7FC-	4C 2D FF	JMP	\$FF2D
C7FF-	FF	???	

CHECK DATA FOR INTEGRITY

IF GOOD, GO TO RAM BOOT LOADER  
ELSE, "ERR" AND BELL

DOS 3.2 RAM BOOTSTRAP LOADER & EPA VECTOR

(\$300-\$3FF)

# DOS 3.2 BOOTSTRAP VECTOR

(TRACK 0 SECTOR 0)

CHECKSUM

```

0300- 99 B9 00 LDA $800,Y ←
0303- 08
0304- 0A ASL
0305- 0A ASL
0306- 0A ASL
0307- 99 00 08 STA $0800,Y
030A- C8 INY
030B- D0 F4 BNE $0301
030D- A6 2B LDX $2B
030F- A9 09 LDA #$09
0311- 85 27 STA $27
0313- AD CC 03 LDA $03CC
0316- 85 41 STA $41
0318- 84 40 STY $40
031A- 8A TXA
031B- 4A LSR
031C- 4A LSR
031D- 4A LSR
031E- 4A LSR
031F- 09 C0 ORA #$C0
0321- 85 3F STA $3F
0323- A9 5D LDA #$5D
0325- 85 3E STA $3E
0327- 20 43 03 JSR $0343 ←
032A- 20 46 03 JSR $0346
032D- A5 3D LDA $3D
032F- 4D FF 03 EOR $03FF
0332- F0 06 BEQ $033A
0334- E6 41 INC $41
0336- E6 3D INC $3D
0338- D0 ED BNE $0327
033A- 85 3E STA $3E
033C- AD CC 03 LDA $03CC
033F- 85 3F STA $3F
0341- E6 3F INC $3F
0343- 6C 3E 00 JMP ($003E)
0346- A2 32 LDX #$32
0348- A0 00 LDY #$00
034A- BD 00 08 LDA $0800,X ←
034D- 4A LSR
034E- 4A LSR
034F- 4A LSR
0350- 85 3C STA $3C
0352- 4A LSR
0353- 85 2A STA $2A
0355- 4A LSR
0356- 1D 00 09 ORA $0900,X
0359- 91 40 STA ($40),Y
035B- C8 INY
035C- BD 33 08 LDA $0833,X
035F- 4A LSR
0360- 4A LSR
0361- 4A LSR
0362- 4A LSR
0363- 26 3C ROL $3C
0365- 4A LSR
0366- 26 2A ROL $2A
0368- 1D 33 09 ORA $0933,X
036B- 91 40 STA ($40),Y
036D- C8 INY
    
```

SHIFT DISK CODE TRANSLATE TABLE LEFT 3 BIT POSITIONS

GET SLOT\*16 (S#)

USE \$900 FOR DISK BUFFER

WHERE TO PUT DOS AT HIMEM (16K IN THIS CASE) PRIOR TO RELOCATION)

CONSTRUCT ADDRESS OF DISK SECTOR READ SUBROUTINE IN DISK ROM

Cs5D: s = slot #

JSR \$Cs5D - READ NEXT SECTOR MERGE BUFFERS TO [40,41]

WHEN SECTOR=9, ALL ARE READ

ELSE, NEXT PAGE/SECTOR

GET LOAD POINT AGAIN (\$3600)

GO TO DOS LOADER (\$3700)

## DISK BUFFER MERGE SUBR.

A<sub>1</sub>A<sub>2</sub>A<sub>3</sub>A<sub>4</sub> A<sub>5</sub>000 MERGE \$800/\$900 TO [840,841]

000A, A<sub>2</sub>A<sub>3</sub>A<sub>4</sub>A<sub>5</sub>

0000 A<sub>1</sub>A<sub>2</sub>A<sub>3</sub>A<sub>4</sub>

0000 0A<sub>1</sub>A<sub>2</sub>A<sub>3</sub>

D<sub>1</sub>D<sub>2</sub>D<sub>3</sub>D<sub>4</sub> D<sub>5</sub>A<sub>1</sub>A<sub>2</sub>A<sub>3</sub>

— 1ST ALTERNATING BYTE —

B<sub>1</sub>B<sub>2</sub>B<sub>3</sub>B<sub>4</sub> B<sub>5</sub>000

0000 B<sub>1</sub>B<sub>2</sub>B<sub>3</sub>B<sub>4</sub> C=B<sub>5</sub>

000A, A<sub>2</sub> A<sub>3</sub>A<sub>4</sub>A<sub>5</sub> B<sub>5</sub>

0000 0B<sub>1</sub>B<sub>2</sub>B<sub>3</sub> C=B<sub>4</sub>

000A, A<sub>2</sub>A<sub>3</sub>A<sub>4</sub>B<sub>4</sub>

D<sub>1</sub>D<sub>2</sub>D<sub>3</sub>D<sub>4</sub> D<sub>5</sub>B<sub>1</sub>B<sub>2</sub>B<sub>3</sub>

— 2ND ALTERNATING BYTE —

036E-	BD 66 08	LDA	#0866,X
0371-	4A	LSR	
0372-	4A	LSR	
0373-	4A	LSR	
0374-	4A	LSR	
0375-	26 3C	ROL	#3C
0377-	4A	LSR	
0378-	26 2A	ROL	#2A
037A-	1D 66 09	ORA	#0966,X
037D-	91 40	STA	(\$40),Y
037F-	C8	INY	
0380-	A5 2A	LDA	#2A
0382-	29 07	AND	#07
0384-	1D 99 09	ORA	#0999,X
0387-	91 40	STA	(\$40),Y
0389-	C8	INY	
038A-	A5 3C	LDA	#3C
038C-	29 07	AND	#07
038E-	1D CC 09	ORA	#09CC,X
0391-	91 40	STA	(\$40),Y
0393-	C8	INY	
0394-	CA	DEX	
0395-	10 B3	BPL	#034A
0397-	AD 99 08	LDA	#0899
039A-	4A	LSR	
039B-	4A	LSR	
039C-	4A	LSR	
039D-	0D FF 09	ORA	#09FF
03A0-	91 40	STA	(\$40),Y
03A2-	A6 2B	LDX	#2B
03A4-	60	RTS	

C<sub>1</sub>C<sub>2</sub>C<sub>3</sub>C<sub>4</sub> C<sub>5</sub>∅∅∅

∅∅∅∅ C<sub>1</sub>C<sub>2</sub>C<sub>3</sub>C<sub>4</sub> C=C<sub>5</sub>

∅A<sub>1</sub>A<sub>2</sub>A<sub>3</sub> A<sub>4</sub>A<sub>5</sub>B<sub>5</sub>C<sub>5</sub>

∅∅∅∅ ∅C<sub>1</sub>C<sub>2</sub>C<sub>3</sub> C=C<sub>4</sub>

∅∅A<sub>1</sub>A<sub>2</sub> A<sub>3</sub>A<sub>4</sub>B<sub>4</sub>C<sub>4</sub>

D<sub>1</sub>D<sub>2</sub>D<sub>3</sub>D<sub>4</sub> D<sub>5</sub>C<sub>1</sub>C<sub>2</sub>C<sub>3</sub>

— 3<sup>rd</sup> ALTERNATING BYTE —

∅∅∅∅ ∅A<sub>4</sub>B<sub>4</sub>C<sub>4</sub>

E<sub>1</sub>E<sub>2</sub>E<sub>3</sub>E<sub>4</sub> E<sub>5</sub>A<sub>4</sub>B<sub>4</sub>C<sub>4</sub>

— 4<sup>th</sup> ALTERNATING BYTE —

∅∅∅∅ ∅A<sub>5</sub>B<sub>5</sub>C<sub>5</sub>

F<sub>1</sub>F<sub>2</sub>F<sub>3</sub>F<sub>4</sub> F<sub>5</sub>A<sub>5</sub>B<sub>5</sub>C<sub>5</sub>

— 5<sup>th</sup> ALTERNATING BYTE —

DO 51 GROUPS OF 5 BYTES

THEN THE LAST BYTE TO MAKE  
256

RESTORE SLOT #  
AND EXIT

03A5-	FF	???
03A6-	FF	???
03A7-	FF	???
03A8-	FF	???
03A9-	FF	???
03AA-	FF	???
03AB-	FF	???
03AC-	FF	???
03AD-	FF	???
03AE-	FF	???
03AF-	FF	???
03B0-	FF	???
03B1-	FF	???
03B2-	FF	???
03B3-	FF	???
03B4-	FF	???
03B5-	FF	???
03B6-	FF	???
03B7-	FF	???
03B8-	FF	???
03B9-	FF	???
03BA-	FF	???
03BB-	FF	???
03BC-	FF	???
03BD-	FF	???
03BE-	FF	???
03BF-	FF	???
03C0-	FF	???
03C1-	FF	???
03C2-	FF	???
03C3-	FF	???
03C4-	FF	???

03C5-	FF	???			
03C6-	FF	???			
03C7-	FF	???			
03C8-	FF	???			
03C9-	FF	???			
03CA-	FF	???			
03CB-	FF	???			
03CC-	3A FF	R0L		\$FF,X	
03CE-	FF	???			
03CF-	FF	???			
03D0-	4C BF 9D	JMP	\$9DBF	DOS WARMSTART	DOS VECTOR
03D3-	4C 84 9D	JMP	\$9D84	DOS COLDSTART	
03D6-	4C FD AA	JMP	\$AAFD	FILE MANAGER (FIO)	
03D9-	4C B5 B7	JMP	\$B7B5	READ/WRITE TRACK/SECTOR (RWTS)	
03DC-	AD 0F 9D	LDA	\$9D0F	} FIND FIO PARMLIST	
03DF-	AC 0E 9D	LDY	\$9D0E		
03E2-	60	RTS		} FIND RWTS PARMLIST	
03E3-	AD C2 AA	LDA	\$AAC2		
03E6-	AC C1 AA	LDY	\$AAC1		
03E9-	60	RTS			
*03EA-	4C 51 A8	JMP	\$A851	REPLACE DOS INTERCEPTS	
*03ED-	EA	NOP			
*03EE-	EA	NOP			
*03EF-	4C 59 FA	JMP	\$FA59		
*03F2-	BF	???		[DOS WARMSTART]	
*03F3-	9D 38 4C	JMP	\$FF58	RTS (FP & VECTOR)	
*03F6-	58				
*03F7-	FF				
*03F8-	4C 65 FF	JMP	\$FF65	MON (CTL-Y VECTOR)	
*03FB-	4C 65 FF	JMP	\$FF65	MON (NMI VECTOR)	
*03FE-	65 FF	ADC	\$FF	[MON] (IRQ VECTOR)	

DOS 3.2 RELOCATER  
(\$1B00-\$1C7C)



# DOS RELOCATER 3.2

1B00-	4C 84 1D	JMP	#1D84
→ 1B03-	A9 BF	LDA	##BF
EPA 1B05-	85 41	STA	#41
1B07-	A2 00	LDX	##00
1B09-	86 40	STX	#40
1B0B-	A0 00	LDY	##00
1B0D-	A1 40	LDA	(\$40, X)
1B0F-	85 26	STA	#26
1B11-	98	TYA	
1B12-	45 26	EOR	#26
1B14-	85 26	STA	#26
1B16-	98	TYA	
1B17-	41 40	EOR	(\$40, X)
1B19-	81 40	STA	(\$40, X)
1B1B-	C5 26	CMP	#26
1B1D-	D0 05	BNE	#1B24
1B1F-	C8	INY	
1B20-	D0 EF	BNE	#1B11
1B22-	F0 04	BEQ	#1B28
1B24-	C6 41	DEC	#41
1B26-	D0 E3	BNE	#1B0B
1B28-	A5 41	LDA	#41
1B2A-	29 DF	AND	##DF
1B2C-	85 43	STA	#43
1B2E-	86 42	STX	#42
1B30-	A1 42	LDA	(\$42, X)
1B32-	48	PHA	
1B33-	85 26	STA	#26
1B35-	98	TYA	
1B36-	45 26	EOR	#26
1B38-	85 26	STA	#26
1B3A-	98	TYA	
1B3B-	41 40	EOR	(\$40, X)
1B3D-	81 42	STA	(\$42, X)
1B3F-	C5 26	CMP	#26
1B41-	D0 09	BNE	#1B4C
1B43-	C8	INY	
1B44-	D0 EF	BNE	#1B35
1B46-	A4 43	LDY	#43
1B48-	68	PLA	
1B49-	4C 51 1B	JMP	#1B51
1B4C-	68	PLA	
1B4D-	81 42	STA	(\$42, X)
1B4F-	A4 41	LDY	#41
1B51-	C8	INY	
1B52-	8C 79 1C	STY	#1C79
1B55-	38	SEC	
1B56-	98	TYA	
1B57-	ED 7A 1C	SBC	#1C7A
1B5A-	8D 78 1C	STA	#1C78
1B5D-	38	SEC	
1B5E-	ED 76 1C	SBC	#1C76
1B61-	F0 9D	BEQ	#1B00
1B63-	8D 7B 1C	STA	#1C7B
1B66-	AD 76 1C	LDA	#1C76
1B69-	8D 0D 1D	STA	#1D0D
1B6C-	A9 1D	LDA	##1D
1B6E-	8D 49 37	STA	#3749
1B71-	A9 84	LDA	##84
1B73-	8D 48 37	STA	#3748
1B76-	A2 00	LDX	##00

GO TO DOS COLDSTART  
 ENTER FROM FINAL DOS BOOT (3747)  
 40, 41 → \$BF00, LAST RAM PAGE

INIT TEST VAL \* TO 0  
 GET BYTE  
 SAVE  
 A = TEST VALUE  
 RAM BYTE = TEST XOR RAM  
 GET TEST VAL AGAIN  
 RAM BYTE = TEST XOR RAM  
 PUT BACK  
 SAME? RAM THERE?  
 NO, NO SUCH PAGE, NEXT UP  
 YES, MAKE SURE WITH ALL VALUES  
 CONTINUE  
 ALL IS WELL, 41 = LAST RAM PAGE  
 NEXT PAGE

FIND  
 LAST  
 RAM  
 PAGE

GET GOOD PAGE (3F, 7F, EF) Y=0  
 CONVERT TO (1F, 5F, 9F) (-8K)

42, 43 = RAM FIRST ADDR  
 GET BYTE AT TOP  
 SAVE  
 SAVE  
 TEST VALUE

RAM XOR TEST  
 TEST AGAIN

RAM XOR TEST  
 SAME?  
 NO, BAD RAM  
 YES, SO FAR, NEXT TEST  
 CONTINUE  
 ALL IS WELL, Y=1st PAGE  
 FIX STACK

LAST RAM  
 SAVE 1st NON RAM

END - LEN DOS (23) = FIRST DOS PAGE  
 SAVE

(1D84) ALREADY AT 1D00  
 YES, NO RELOC NEEDED, COLDSTART, IDN0  
 RELOCATION VALUE  
 1D, CURRENT DOS START  
 TO DOS EPA

TO LOADER "JMP" TO ALLOW COLDSTART  
 SO BOOT → RELOC BECOMES  
 BOOT TO xD84 (COLDSTART)  
 SET 40/41 BASE

1B78-	86 40		STX	\$40
1B7A-	BD 29 1C	→	LDA	\$1C29, X
1B7D-	A8		TAY	
1B7E-	BD 2A 1C	→	LDA	\$1C2A, X
1B81-	85 41		STA	\$41
1B83-	4C 93 1B	→	JMP	\$1B93
1B86-	18		CLC	
1B87-	B1 40		LDA	(\$40), Y
1B89-	6D 7B 1C	→	ADC	\$1C7B
1B8C-	91 40		STA	(\$40), Y
1B8E-	C8		INX	
1B8F-	D0 02	←	BNE	\$1B93
1B91-	E6 41		INC	\$41
1B93-	C8	→	INX	
1B94-	D0 02	←	BNE	\$1B98
1B96-	E6 41		INC	\$41
1B98-	A5 41	→	LDA	\$41
1B9A-	DD 2C 1C	→	CMP	\$1C2C, X
1B9D-	90 E7		BCC	\$1B86 →
1B9F-	98		TYA	
1BA0-	DD 2B 1C	→	CMP	\$1C2B, X
1BA3-	90 E1		BCC	\$1B86 →
1BA5-	8A		TXA	
1BA6-	18		CLC	
1BA7-	69 04		ADC	##04
1BA9-	AA		TAX	
1BAA-	EC 28 1C	→	CPX	\$1C28
1BAD-	90 CB	←	BCC	\$1B7A
1BAF-	A2 00		LDX	##00
1BB1-	8E 9C 33	→	STX	\$339C
1BB4-	BD 5A 1C	→	LDA	\$1C5A, X
1BB7-	85 40		STA	\$40
1BB9-	BD 5B 1C	→	LDA	\$1C5B, X
1BBC-	85 41		STA	\$41
1BBE-	A2 00	→	LDX	##00
1BC0-	A1 40		LDA	(\$40, X)
1BC2-	20 8E F8	→	JSR	\$F88E
1BC5-	A4 2F		LDY	\$2F
1BC7-	C0 02		CPY	##02
1BC9-	D0 11		BNE	\$1BDC
1BCB-	B1 40		LDA	(\$40), Y
1BCD-	CD 76 1C	→	CMP	\$1C76
1BD0-	90 0A	←	BCC	\$1BDC
1BD2-	CD 77 1C	→	CMP	\$1C77
1BD5-	B0 05	←	BCS	\$1BDC
1BD7-	6D 7B 1C	→	ADC	\$1C7B
1BDA-	91 40		STA	(\$40), Y
1BDC-	38	→	SEC	
1BDD-	A5 2F		LDA	\$2F
1BDF-	65 40		ADC	\$40
1BE1-	85 40		STA	\$40
1BE3-	A9 00		LDA	##00
1BE5-	65 41		ADC	\$41
1BE7-	85 41		STA	\$41
1BE9-	AE 9C 33	→	LDX	\$339C
1BEC-	DD 5D 1C	→	CMP	\$1C5D, X
1BEF-	90 CD	←	BCC	\$1BBE
1BF1-	A5 40		LDA	\$40
1BF3-	DD 5C 1C	→	CMP	\$1C5C, X
1BF6-	90 C6	←	BCC	\$1BBE
1BF8-	8A		TXA	
1BF9-	18		CLC	
1BFA-	69 04		ADC	##04

40, 41 → INSTRUCTIONS

GET LOW ADDR  
 SAVE IN Y  
 GET HI  
 40, 41 → HI  
 PROCESS 1ST BYTE IN RANGE  
 FOR ADD  
 GET ADDR HI  
 RELOCATE  
 REPLACE  
 TO LOW  
 MORE THIS PAGE  
 NEXT PAGE  
 NEXT BYTE THIS PAGE (HI)  
 MORE THIS PAGE  
 NEXT PAGE  
 GET HI  
 COMPARE TO RANGE END  
 < MORE TO DO  
 ≥ PAST  
 CHECK LOW  
 < MORE TO DO  
 ≥ GET TABLE INDEX  
 X=X+4, NEXT ENTRY  
 DONE (9 ENTRIES)?  
 NO, NEXT TABLE ENTRY  
 YES, NEXT TABLE  
 SAVE INDEX

RELOCATE  
 ALL  
 ADDRESSES  
 CONSTRA

40, 41 → BEGIN OF RANGE

FOR INDEXING  
 OPCODE  
 INDSB2, GET INSTRUCTION LENGTH-1  
 LENGTH (-1)  
 3-BYTE?  
 NO  
 YES, GET HI PART OF ABS ADDR  
 BEFORE DOS  
 YES, IGNORE  
 AFTER DOS  
 YES, IGNORE  
 NO, RELOCATE  
 RESTORE  
 +1 FOR LENGTH  
 LENGTH-1

RELOCATE  
 ALL  
 CODE

NEXT INSTRUCTION

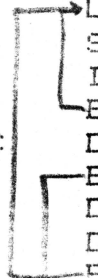
REGET INDEX  
 END RANGE  
 NO  
 REALLY?

NO  
YES

X=X+4, NEXT TABLE ENTRY

1BFC-	AA		TAX	
1BFD-	EC 59	1C	CPX	\$1C59
1C00-	90 AF		BCC	\$1BB1
1C02-	A9 3F		LDA	#\$3F
1C04-	85 41		STA	\$41
1C06-	AC 79	1C	LDY	\$1C79
1C09-	88		DEY	
1C0A-	84 43		STY	\$43
1C0C-	A9 00		LDA	#\$00
1C0E-	85 40		STA	\$40
1C10-	85 42		STA	\$42
1C12-	A8		TAY	
1C13-	B1 40		LDA	(\$40),Y
1C15-	91 42		STA	(\$42),Y
1C17-	C8		INY	
1C18-	D0 F9		BNE	\$1C13
1C1A-	CE 7C	1C	DEC	\$1C7C
1C1D-	F0 06		BEQ	\$1C25
1C1F-	C6 41		DEC	\$41
1C21-	C6 43		DEC	\$43
1C23-	D0 EE		BNE	\$1C13
1C25-	4C 54	1E	JMP	\$1E54

DONE (6 ENTRIES)  
 NO  
 YES, } 40,41 → 3F00 END SOURCE  
 } 42,43 → BF00 END TARGET  
 LAST D0S }



COPY RELOCATED  
 VERSION TO TOP  
 OF RAM

GO TO COLDSTART (SEE IMAGE)

**ADDR RELOC TABLE**

1C28-	24	00	BIT	\$00
1C2A-	1D	56	ORA	\$1D56, X
1C2D-	58		CLI	
1C2E-	1D	5A	ORA	\$1D5A, X
1C31-	64		???	
1C32-	1D	66	ORA	\$1D66, X
1C35-	6C	1D	JMP	(\$701D)
1C38-	1D	78	ORA	\$1D78, X
1C3B-	7C		???	
1C3C-	1D	7E	ORA	\$1D7E, X
1C3F-	80		???	
1C40-	1D	C1	ORA	\$2AC1, X
1C43-	FD	2A	SBC	\$E42A, X
1C46-	37		???	
1C47-	E8		INX	
1C48-	37		???	
1C49-	EE	37	INC	\$F037
1C4C-	37		???	
1C4D-	00		BRK	
1C4E-	00		BRK	
1C4F-	00		BRK	
1C50-	00		BRK	
1C51-	00		BRK	
1C52-	00		BRK	
1C53-	00		BRK	
1C54-	00		BRK	
1C55-	00		BRK	
1C56-	00		BRK	
1C57-	00		BRK	
1C58-	00		BRK	

- 1D00, 1D56
  - 1D58, 1D5A
  - 1D64, 1D66
  - 1D6C, 1D70
  - 1D78, 1D7C
  - 1D7E, 1D80
  - 2AC1, 2AFD
  - 37E4, 37E8
  - 37EE, 37F0
- START, END

Bytes within START, END  
 ranges treated as address  
 constants to be relocated

1C59-	18		CLC	
1C5A-	84	1D	STY	\$1D
1C5C-	84	28	STY	\$28
1C5E-	FD	2A	SBC	\$972A, X
1C61-	33		???	
1C62-	00		BRK	
1C63-	37		???	
1C64-	E0	37	CPX	#\$37
1C66-	FE	35	INC	\$FE35, X
1C69-	35	00	AND	\$00, X
1C6B-	38		SEC	

- 1D84, 2884
- 2AFD, 3397
- 3700, 37E0
- 35FE, 35FE
- 3800, 3A8F
- 3D00, 3FFF

**CODE RELOC TABLE**

- 1D84, 2884
- 2AFD, 3397
- 365D, 37E0
- 3C56, 3CDF
- 3800, 3A11
- 3D00, 3FA8
- 3FC8, 3FFF

1C6C- 8F ???  
 1C6D- 3A ???  
 1C6E- 00 BRK  
 1C6F- 3D FF 3F AND \$3FFF, X  
 1C72- 00 BRK  
 1C73- 00 BRK  
 1C74- 00 BRK  
 1C75- 00 BRK

1C76- 1D 40 9D DRA \$9D40, X  
 1C79- C0 23 CPY ##23  
 1C7B- 80 ???  
 1C7C- 00 BRK

1C76:	LOWEST DOS	DATA
1C77:	HIGHEST DOS	
1C78:	FIRST DOS PAGE	
1C79:	FIRST NON-RAM PAGE	
1C7A:	35, LENGTH OF DOS (IN PAGES)	
1C7B:	RELOCATION VALUE	
1C7C:	COPY INDEX	

DOS 3.2

(xD00-yFFF)

x= \$1D	y= \$3F	(16K)
\$3D	\$5F	(24K)
\$5D	\$7F	(32K)
\$7D	\$9F	(40K)
\$9D	\$BF	(48K)

PRELIMINARY DOS 3.2 MAP (48K)

- 9D00 - 9D83 : START DOS , TABLES
- 9D84 - 9DEB : COLDSTART
- 9DEE - 9DE9 : WARMSTART
- 9DEA - 9E50 : ENTRY PROCESSING
- 9E51 - 9E80 : PAGE 3 \$3D0 VECTOR IMAGE
- 9E81 - 9FC7 : KEYBOARD/VIDEO INTERCEPTS
- 9FC8 - A192 : DOS COMMAND PARSING/PROCESSING
- A193 - A228 : MISC. UTILITY ROUTINES
- A229 - A60D : COMMAND HANDLING
- A60E - A719 : MISC. UTILITY ROUTINES
- A71A - A850 : BUFFER MANAGEMENT
- A851 - A883 : SET DOS CSWL/KSWL INTERCEPTS
- A884 - A970 : COMMAND/KEYWORD TABLES
- A971 - AA4E : MESSAGES
- AA4F - AA74 : CONSTANTS, WORKAREAS
- AA75 - AAB0 : FILE NAME BUFFERS
- AAB1 - AAC8 : WORKAREA
- AAC9 - B3BA : FIO, CMDS, PROCESSING
- B3BB - B4BA : VTOC BUFFER
- B4BB - B5BA : CATALOG BUFFER
- B5BB - B5FF : FIO PARMS, WORKAREA
- B600 - B6FF : TRACK 0 / SECTOR 0 PAGE 3 BOOT IMAGE
- B700 - B7B4 : DOS BOOT PHASE 3
- B7B5 - BFFF : RWTS, BUFFERS, TABLES



9D00-	D3	???
9D01-	9C	???
9D02-	81 9E	STA (\$9E, X)
9D04-	8D 9E 75	LDA \$759E, X
9D07-	AA	TAX
9D08-	93	???
9D09-	AA	TAX
9D0A-	60	RTS
9D0B-	AA	TAX
9D0C-	00	BRK
9D0D-	9D BB B5	STA \$B5BB, X
9D10-	EA	NOP
9D11-	9E	???
9D12-	11 9F	ORA (\$9F), Y
9D14-	22	???
9D15-	9F	???
9D16-	2E 9F 51	ROL \$519F
9D19-	9F	???
9D1A-	60	RTS
9D1B-	9F	???
9D1C-	70 9F	BVS \$9CBD
9D1E-	4E A5 12	LSR \$12A5
9D21-	A4 96	LDY \$96
9D23-	A3	???
9D24-	D0 A4	BNE \$9CCA
9D26-	EF	???
9D27-	A4 62	LDY \$62
9D29-	A2 70	LDX #\$70
9D2B-	A2 74	LDX #\$74
9D2D-	A2 E9	LDX #\$E9
9D2F-	A2 1A	LDX #\$1A
9D31-	A5 C5	LDA \$C5
9D33-	A5 OF	LDA \$OF
9D35-	A5 DC	LDA \$DC
9D37-	A5 A2	LDA \$A2
9D39-	A2 97	LDX #\$97
9D3B-	A2 80	LDX #\$80
9D3D-	A2 6D	LDX #\$6D
9D3F-	A5 32	LDA \$32
9D41-	A2 3C	LDX #\$3C
9D43-	A2 28	LDX #\$28
9D45-	A2 2D	LDX #\$2D
9D47-	A2 50	LDX #\$50
9D49-	A2 79	LDX #\$79
9D4B-	A5 9D	LDA \$9D
9D4D-	A5 30	LDA \$30
9D4F-	A3	???
9D50-	5C	???
9D51-	A3	???
9D52-	8D A3 7C	STA \$7CA3
9D55-	A2 36	LDX #\$36
9D57-	E8	INX
9D58-	E5 A4	SBC \$A4
9D5A-	E3	???
9D5B-	E3	???
9D5C-	00	BRK
9D5D-	E0 03	CPX #\$03
* 9D5F-	E0 00	CPX #\$00
* 9D61-	00	BRK
9D62-	36 E8	ROL \$E8, X

9D00: FIRST BUFFER LINK ↑  
 9D02: DOS KBD INTERCEPT EP  
 9D04: DOS VID INTERCEPT EP  
 9D06: PRIMARY FILE NAME ↑  
 9D08: SECONDARY FILE NAME ↑  
 9D0A: ↑ RANGE LENGTH  
 9D0C: DOS LOAD POINT (1800, 1000, XDC)  
 9D0E: FIO PARMLIST LOCATION

DOS CSWL INTERCEPT STATE HANDLER TABLE

STATE	EPA
0	9EEB
1	9F12
2	9F23
3	9F2F
4	9F52
5	9FG1
6	9F71

COMMAND HANDLER EPA TABLE

+00	INIT	A54F
02	LOAD	A413
04	SAVE	A397
06	RUN	A4D1
08	CHAIN	A4F0
0A	DELETE	A263
0C	LOCK	A271
0E	UNLOCK	A275
10	CLOSE	A2EA
12	READ	A51B
14	EXEC	A5C6
16	WRITE	A510
18	POSITION	A5DD
1A	OPEN	A2A3
1C	APPEND	A298
1E	RENAME	A281
20	CATALOG	A56E
22	MON	A233
24	NOMON	A23D
26	PK#	A229
28	IN#	A22E
2A	MAXFILES	A251
2C	FP	A57A
2E	INT	A59E
30	BSAVE	A331
32	BLOAD	A35D
34	BRUN	A38E
36	VERIFY	A27D

ACTIVE BASIC ENTRY POINT VECTOR  
 +0 CHAIN  
 +2 RUN  
 +4 ERROR EPA  
 +6 COLDSTART  
 +8 WARMSTART  
 +A RELOCATE PGM (FP ONLY)

INT BASIC EPA IMAGE

9D64-	E5 A4	SBC	\$A4
9D66-	E3	???	
9D67-	E3	???	
9D68-	00	BRK	
9D69-	E0 03	CPX	##03
9D6B-	E0 FC	CPX	##FC
9D6D-	A4 FC	LDY	\$FC
9D6F-	A4 65	LDY	\$65
9D71-	D8	CLD	
9D72-	00	BRK	
9D73-	E0 3C	CPX	##3C
9D75-	D4	???	
9D76-	F2	???	
9D77-	D4	???	

FP ROM BASIC EPA IMAGE

9D78-	06 A5	ASL	\$A5
9D7A-	06 A5	ASL	\$A5
9D7C-	67	???	
9D7D-	10 84	BPL	\$9D03
9D7F-	9D 3C 0C	STA	\$0C3C, X
9D82-	F2	???	
9D83-	0C	???	

FP RAM BASIC EPA IMAGE

9D84-	AD E9 B7	LDA	\$B7E9
9D87-	4A	LSR	
9D88-	4A	LSR	
9D89-	4A	LSR	
9D8A-	4A	LSR	
9D8B-	8D 6A AA	STA	\$AA6A
9D8E-	AD EA B7	LDA	\$B7EA
9D91-	8D 68 AA	STA	\$AA68
9D94-	AD 00 E0	LDA	\$E000
9D97-	49 20	EOR	##20
9D99-	D0 11	BNE	\$9DAC
9D9B-	8D B6 AA	STA	\$AAB6
9D9E-	A2 0A	LDX	##0A
9DA0-	BD 61 9D	LDA	\$9D61, X
9DA3-	9D 55 9D	STA	\$9D55, X
9DA6-	CA	DEX	
9DA7-	D0 F7	BNE	\$9DA0
9DA9-	4C BC 9D	JMP	\$9DBC
9DAC-	A9 40	LDA	##40
9DAE-	8D B6 AA	STA	\$AAB6
* 9DB1-	A2 0C	LDX	##0C
9DB3-	BD 6B 9D	LDA	\$9D6B, X
9DB6-	9D 55 9D	STA	\$9D55, X
9DB9-	CA	DEX	
9DBA-	D0 F7	BNE	\$9DB3
9DBC-	38	SEC	
9DBD-	B0 12	BCS	\$9DD1

GET SLOT\*16

DOS COLDSTART

116

SAVE SLOT #

SAVE DRIVE #

IS APPLESOFT ROM ACTIVE?

REMEMBER TYPE

COPY INT BASIC EPAs TO ACTIVE BASIC ENTRY POINT VECTOR

REMEMBER TYPE

COPY FP ROM BASIC EPAs to Active BASIC ENTRY POINT VECTOR

DOS WARMSTART

GET BASIC TYPE NOT INT?

INT ROM NEEDED

IF RAM FP, DON'T FOOL WITH ROM CARD

FP ROM NEEDED

GO SET APPROPRIATE ROM

REMEMBER COLD OR WARM ENTRY REPLACE DOS KBD/VID INTERCEPTS

NOMON C.I, 0

STATE = 0

COLD OR WARM?

\*

\*

\*

\*

\*

\*

\*

\*

\*

9DDE-	6A		ROR	
9DDF-	8D 51 AA		STA	AA51
9DE2-	30 03		BMI	9DE7
9DE4-	6C 5E 9D		JMP	(9D5E)
9DE7-	6C 5C 9D		JMP	(9D5C)
9DEA-	0A		ASL	
9DEB-	10 19		BPL	9E06
9DED-	8D B6 AA		STA	AAB6
*9DF0-	A2 0C		LDX	#0C
9DF2-	BD 77 9D		LDA	9D77,X
9DF5-	9D 55 9D		STA	9D55,X
9DF8-	CA		DEX	
9DF9-	D0 F7		BNE	9DF2
9DFB-	A2 1D		LDX	#1D
9DFD-	BD 93 AA		LDA	AA93,X
9E00-	9D 75 AA		STA	AA75,X
9E03-	CA		DEX	
9E04-	10 F7		BPL	9DFD
9E06-	AD B1 AA		LDA	AAB1
9E09-	8D 57 AA		STA	AA57
9E0C-	20 D4 A7		JSR	A7D4
9E0F-	AD B3 AA		LDA	AAB3
9E12-	F0 09		BEQ	9E1D
9E14-	48		PHA	
9E15-	20 9D A6		JSR	A69D
9E18-	68		PLA	
9E19-	A0 00		LDY	#00
9E1B-	91 40		STA	(40),Y
9E1D-	20 5B A7		JSR	A75B
9E20-	AD 5F AA		LDA	AA5F
9E23-	D0 20		BNE	9E45
*9E25-	A2 2F		LDX	#2F
9E27-	BD 51 9E		LDA	9E51,X
9E2A-	9D D0 03		STA	03D0,X
9E2D-	CA		DEX	
9E2E-	10 F7		BPL	9E27
*9E30-	AD 53 9E		LDA	9E53
*9E33-	8D F3 03		STA	03F3
*9E36-	49 A5		EOR	#A5
*9E38-	8D F4 03		STA	03F4
*9E3B-	AD 52 9E		LDA	9E52
*9E3E-	8D F2 03		STA	03F2
*9E41-	A9 06		LDA	#06
*9E43-	D0 05		BNE	9E4A
9E45-	AD 62 AA		LDA	AA62
9E48-	F0 06		BEQ	9E50
9E4A-	8D 5F AA		STA	AA5F
*9E4D-	4C 80 A1		JMP	A180
*9E50-	60		RTS	

00 - WARM 80 - COLD

SET ENTRY VALUE

WARMSTART - PROPER BASIC  
COLDSTART -

IN RAM FP? FIRST ENTRY PROC

SAY FP RAM (80) ACTIVE

COPY ITS EPA'S TO ACTIVE BASIC VECTOR

BLANK OUT GREETING FILE NAME (CAN'T HAVE ONE WITH RAM FP)

SET MAXFILES TO DEFAULT VALUE (3)

GO DO BUFFER INITIALIZATION

EXEC ACTIVE?

YES...

[#40] -> EXEC FILE

CLOSE EXEC FILE

STATE = 0 / WARMSTART STATUS ANY CMDS PARSED YET?

COPY DOS VECTOR TO #3D0

#3F2, #3F3 = DOS WARMSTART ↑

#3F4 = "9D" XOR "A5" = "38"

"RUN" COMMAND FOR GREETING FILE

GET PENDING COMMAND (IF ANY)

NONE?

SET COMMAND TO DO

GO DO IT

9E51-3D0	4C BF 9D	JMP	9DBF	DOS WARMSTART
9E54-3D3	4C 84 9D	JMP	9D84	DOS COLDSTART
9E57-3D6	4C FD AA	JMP	AAFD	FILE MGR (FIO)
9E5A-3D9	4C B5 B7	JMP	B7B5	RWTS
9E5D-3DC	AD 0F 9D	LDA	9D0F	} FIND FIO PARMLIST
9E60-	AC 0E 9D	LDY	9D0E	
9E63-	60	RTS		
9E64-3E3	AD C2 AA	LDA	AA62	} FIND RWTS PARMLIST
9E67-	AC C1 AA	LDY	AA61	
9E6A-	60	RTS		
9E6B-3EA	4C 51 A8	JMP	A851	REPLACE DOS INTERCEPTS
9E6E-	EA	NOP		
9E6F-	EA	NOP		
9E70-3EF	4C 59 FA	JMP	FA59	AUTO START ROM BREAK HANDLER

IMAGE OF DOS VECTOR MOVED TO #3D0

9E73-3F2	4C 65	FF	JMP	\$FF65	MON (AUTOSTART ROM POWER UP)
9E76-3F5	4C 58	FF	JMP	\$FF58	RTS (FP & LOCATION)
9E79-3F8	4C 65	FF	JMP	\$FF65	MON (CTL-Y VECTOR)
9E7C-3FB	4C 65	FF	JMP	\$FF65	MON (NMI VECTOR)
9E7F-3FE	65 FF		ADC	\$FF	(MON) (IRQ VECTOR)
9E81-	20 D1	9E	JSR	\$9ED1	SAVE REGS
9E84-	AD 51	AA	LDA	\$AA51	ENTRY VALUE
9E87-	F0 15		BEQ	\$9E9E	WARM?
*9E89-	48		PHA		
9E8A-	AD 5C	AA	LDA	\$AA5C	GET ACC CONTENTS
9E8D-	91 28		STA	(\$28),Y	ECHO IT (ERASE CURSOR)
*9E8F-	68		PLA		
9E90-	30 03		BMI	\$9E95	NOW COLDSTART IF WANTED
9E92-	4C 26	A6	JMP	\$A626	IN READ STATE, GET DISK DATA
*9E95-	20 EA	9D	JSR	\$9DEA	DO FIRST TIME PROCESSING
9E98-	A4 24		LDY	\$24	CH
9E9A-	A9 60		LDA	##60	FLASHING BLANK (CURSOR)
9E9C-	91 28		STA	(\$28),Y	PUT ON SCREEN
9E9E-	AD B3	AA	LDA	\$AAB3	EXECING?
9EA1-	F0 03		BEQ	\$9EA6	
9EA3-	20 82	A6	JSR	\$A682	YES, GET A BYTE FROM EXEC FILE
9EA6-	A9 03		LDA	##03	STATE=3
9EA8-	8D 52	AA	STA	\$AA52	RESTORE ORIG. REGS
9EAB-	20 BA	9F	JSR	\$9FBA	GO TO TRUE KBD INPUT
9EAE-	20 BA	9E	JSR	\$9EBA	SAVE INPUT CHARACTER
9EB1-	8D 5C	AA	STA	\$AA5C	AND NEW X REG
9EB4-	8E 5A	AA	STX	\$AA5A	
9EB7-	4C B3	9F	JMP	\$9FB3	AND EXIT
9EBA-	6C 38	00	JMP	(\$0038)	TRUE KSWL
9EBD-	20 D1	9E	JSR	\$9ED1	SAVE REGS
9EC0-	AD 52	AA	LDA	\$AA52	GET STATE
9EC3-	0A		ASL		*2 FOR TABLE IDX
9EC4-	AA		TAX		
9EC5-	BD 11	9D	LDA	\$9D11,X	} FIND THIS STATE'S HANDLER ROUTINE
9EC8-	48		PHA		
9EC9-	BD 10	9D	LDA	\$9D10,X	
9ECC-	48		PHA		
9ECD-	AD 5C	AA	LDA	\$AA5C	GET OUTPUTTED CHARACTER
9ED0-	60		RTS		GO TO STATE HANDLER
9ED1-	8D 5C	AA	STA	\$AA5C	} COMMON INTERCEPT HANDLING
9ED4-	8E 5A	AA	STX	\$AA5A	
9ED7-	8C 5B	AA	STY	\$AA5B	
9EDA-	BA		TSX		} SAVE A,X,Y,S REGS
9EDB-	E8		INX		
9EDC-	E8		INX		
9EDD-	8E 59	AA	STX	\$AA59	
9EE0-	A2 03		LDX	##03	} RESTORE TRUE I/O HANDLERS WHILE IN DOS
9EE2-	BD 53	AA	LDA	\$AA53,X	
9EE5-	95 36		STA	\$36,X	
9EE7-	CA		DEX		
9EE8-	10 F8		BPL	\$9EE2	
9EEA-	60		RTS		
*9EEB-	AE B7	AA	LDX	\$AAB7	RUN INTERRUPTED?
*9EEE-	F0 03		BEQ	\$9EF3	STATE #0
*9EF0-	4C 78	9F	JMP	\$9F78	IF SO, GO COMPLETE IT
9EF3-	AE 51	AA	LDX	\$AA51	READING?
9EF6-	F0 08		BEQ	\$9F00	
9EF8-	C9 BF		CMP	##BF	OUTPUTTING A "?" (BASIC INPUT STMT)
9EFA-	F0 75		BEQ	\$9F71	YES, STATE=6 (SKIP THIS CHARACTER)
9EFC-	C5 33		CMP	\$33	ELSE IS IT A PROMPT?
*9EFE-	F0 27		BEQ	\$9F27	YES, STATE=2 (IGNORE LINE)
9F00-	A2 02		LDX	##02	
9F02-	8E 52	AA	STX	\$AA52	STATE=2

DOS KBD INTER.

DOS VID INTER.

COMMON INTERCEPT HANDLING

STATE #0

START OF LINE



9F05-	CD B2 AA	CMP	\$AAB2	CTL-D?		
9F08-	D0 19	BNE	\$9F23	NO, STATE 2 NOW (IGNORE LINE)		
9F0A-	CA	DEX		} STATE=1 (COLLECT DOS COMMAND LINE)		
9F0B-	8E 52 AA	STX	\$AA52			
9F0E-	CA	DEX		} FIRST CHARACTER OF LINE		
9F0F-	8E 5D AA	STX	\$AA5D			
9F12-	AE 5D AA	LDX	\$AA5D	GET POSITION	STATE #1	
9F15-	9D 00 02	STA	\$0200, X	UPDATE INPUT BUFFER	DOS CMD COLLECT	
9F18-	E8	INX		} NEXT POSITION		
9F19-	8E 5D AA	STX	\$AA5D			
9F1C-	C9 8D	CMP	##8D	RETURN CHAR?		
9F1E-	D0 75	BNE	\$9F95	NO, GO OUT VIA ECHO		
9F20-	4C CD 9F	JMP	\$9FCD	YES, TIME TO SCAN IT FOR A DOS CMD		
9F23-	C9 8D	CMP	##8D	RETURN?	STATE #2	
9F25-	D0 7D	BNE	\$9FA4	NOT YET, ECHO	NON-DOS CMD IGNORE	
9F27-	A2 00	LDX	##00	ELSE, BACK TO STATE 0		
9F29-	8E 52 AA	STX	\$AA52			
9F2C-	4C A4 9F	JMP	\$9FA4	ECHO AND EXIT		
9F2F-	A2 00	LDX	##00	} STATE = 0 IF INPUT ENDS	STATE #3	
9F31-	8E 52 AA	STX	\$AA52			
9F34-	C9 8D	CMP	##8D	RETURN?	INPUT ECHO	
9F36-	F0 07	BEQ	\$9F3F			
9F38-	AD B3 AA	LDA	\$AAB3	EXECING?	} (KSWL WILL SET) STATE=3.	
9F3B-	F0 67	BEQ	\$9FA4	NO, ECHO UNCONDITIONALLY		
9F3D-	D0 5E	BNE	\$9F9D	YES, ECHO IF "MON I" SET		
9F3F-	48	PHA		(IF EXECING ALWAYS COLLECT DOS CMD)		
9F40-	38	SEC		EXECING?		
9F41-	AD B3 AA	LDA	\$AAB3	YES		
9F44-	D0 03	BNE	\$9F49	NO, BASIC EXECUTING?		
9F46-	20 5E A6	JSR	\$A65E			
9F49-	68	PLA				
9F4A-	90 EC	BCC	\$9F38	INPUT STMT - DON'T COLLECT DOS CMD		
9F4C-	AE 5A AA	LDX	\$AA5A	} IF BASIC IMMEDIATE OR EXECING, COLLECT		
9F4F-	4C 15 9F	JMP	\$9F15			
9F52-	C9 8D	CMP	##8D	RETURN?	STATE #4	
9F54-	D0 05	BNE	\$9F5B	NO	WRITE MODE	
9F56-	A9 05	LDA	##05	} ELSE, STATE 5 NEXT		
9F58-	8D 52 AA	STA	\$AA52			
9F5B-	20 0E A6	JSR	\$A60E	WRITE THE BYTE		
9F5E-	4C 99 9F	JMP	\$9F99	ECHO AND OUT		
9F61-	CD B2 AA	CMP	\$AAB2	CTL-D?	STATE #5	
9F64-	F0 85	BEQ	\$9EEB	YES, EXIT WRITE MODE	WRITE MODE LINE START	
9F66-	C9 8A	CMP	##8A	LINE FEED?		
9F68-	F0 F1	BEQ	\$9F5B	YES, STAY IN STATE 5		
9F6A-	A2 04	LDX	##04	STATE=4 FOR REMAINDER		
9F6C-	8E 52 AA	STX	\$AA52			
9F6F-	D0 E1	BNE	\$9F52			
9F71-	A9 00	LDA	##00	START OF LINE (STATE=0)	STATE #6	
9F73-	8D 52 AA	STA	\$AA52		SKIP PROMPT CHARACTER	
9F76-	F0 25	BEQ	\$9F9D	ECHO AND OUT		
*9F78-	A9 00	LDA	##00	} RESET FLAG		
*9F7A-	8D B7 AA	STA	\$AAB7			
*9F7D-	20 51 A8	JSR	\$A851	SET DOS INTERCEPTS		
*9F80-	4C DC A4	JMP	\$A4DC	GO FINISH "RUN"		
9F83-	AD 00 02	LDA	\$0200	GET FIRST CHAR OF CMD	DOS COMND SCAN EXIT	
9F86-	CD B2 AA	CMP	\$AAB2	CTL-D?		
9F89-	F0 0A	BEQ	\$9F95	YES, DON'T TOUCH IT		
*9F8B-	A9 8D	LDA	##8D	} NULL LINE SO BASIC WON'T SAY "SYNTAX ERR"		
9F8D-	8D 00 02	STA	\$0200			
9F90-	A2 00	LDX	##00	} ZERO LENGTH INPUT INDEX (X REG)		
9F92-	8E 5A AA	STX	\$AA5A			
9F95-	A9 40	LDA	##40	} ECHO IF MON C	ECHO AND EXIT	
9F97-	D0 06	BNE	\$9F9F			

9F99-	→ A9 10	LDA	##10	} ECHO IF <u>MON 0</u>	
9F9B-	D0 02	←BNE	\$9F9F		
9F9D-	→ A9 20	LDA	##20	} ECHO IF <u>MON I</u>	
9F9F-	2D 5E AA	→AND	\$AA5E		
9FA2-	F0 0F	←BEQ	\$9FB3		
9FA4-	→ 20 BA 9F	JSR	\$9FBA	RESTORE REGS	
9FA7-	20 C5 9F	JSR	\$9FC5	ECHO CHARACTER	
9FAA-	8D 5C AA	STA	\$AA5C	} SAVE REGS AFTER CSWL	
9FAD-	8C 5B AA	STY	\$AA5B		
9FBO-	8E 5A AA	STX	\$AA5A		
9FB3-	20 51 A8	→JSR	\$A851	PUT BACK INTERCEPTS	
9FB6-	AE 59 AA	LDX	\$AA59	RESTORE STACK	
9FB9-	9A	TXS			
9FBA-	AD 5C AA	LDA	\$AA5C	} RESTORE REGS	
9FBD-	AC 5B AA	LDY	\$AA5B		RESTORE A, Y & X
9FBO-	AE 5A AA	LDX	\$AA5A		
9FC0-	AE 5A AA	LDX	\$AA5A		
9FC3-	38	SEC			
9FC4-	60	RTS			
9FC5-	6C 36 00	JMP	(\$0036)	TRUE CSWL	
9FC8-	A9 8D	LDA	##8D	CR	
9FCA-	4C C5 9F	JMP	\$9FC5	PUT IT	
9FCD-	A0 FF	LDY	##FF	} INITIALIZE CMD NUMBER	
9FCF-	8C 5F AA	STY	\$AA5F		
9FD2-	C8	INY		} NO PENDING COMMAND	
9FD3-	8C 62 AA	STY	\$AA62		
9FD6-	EE 5F AA	→INC	\$AA5F	CMD NUMBER = CMD NUMBER + 1	
9FD9-	A2 00	LDX	##00	} ZERO FLAG CLEAR / X = 0	
9FDB-	08	PHP			
9FDC-	BD 00 02	LDA	\$0200, X	GET FIRST CHARACTER	
9FDF-	CD B2 AA	CMP	\$AAB2	CTL-D?	
9FE2-	D0 01	BNE	\$9FE5	} IF SO, SKIP OVER IT	
9FE4-	E8	INX			
9FE5-	8E 5D AA	STX	\$AA5D	START-OF-COMMAND INDEX	
9FE8-	20 A4 A1	→JSR	\$A1A4	FLUSH TO NEXT NON-BLANK	
9FEB-	29 7F	AND	##7F	MSB OFF	
9FED-	59 84 A8	EOR	\$A884, Y	COMPARE TO COMMAND TEXT TABLE	
9FF0-	C8	INY			
9FF1-	0A	ASL		IGNORE MISMATCHING MSB / C=END	
9FF2-	F0 02	BEQ	\$9FF6	} REMEMBER MISMATCHES	
9FF4-	68	PLA			
9FF5-	08	PHP		} END OF COMMAND?	
9FF6-	90 F0	←BCC	\$9FE8		
9FF8-	28	PLP		} WELL, DID IT MATCH?	
9FF9-	F0 20	BEQ	\$A01B		
9FFB-	B9 84 A8	→LDA	\$A884, Y	} MORE NAMES IN TABLE?	
9FFE-	D0 D6	←BNE	\$9FD6		
A000-	AD 00 02	LDA	\$0200	CTL-D?	
A003-	CD B2 AA	CMP	\$AAB2		
A006-	F0 03	BEQ	\$A00B	NOT DOS CMD, LEAVE IT ALONE	
A008-	4C A4 9F	JMP	\$9FA4	NULL DOS CMD?	
A00B-	AD 01 02	LDA	\$0201		
A00E-	C9 8D	CMP	##8D	IF NOT, SYNTAX ERROR	
A010-	D0 06	←BNE	\$A018	RESET STATE / WARMSTART	
A012-	20 5B A7	→JSR	\$A75B	ECHO COMMAND AND EXIT	
A015-	4C 95 9F	JMP	\$9F95	"COMMAND SYNTAX ERROR"	
A018-	4C C4 A6	→JMP	\$A6C4	CMDNUM * 2	
A01B-	0E 5F AA	→ASL	\$AA5F	INDEX TO OPERAND TABLE	
A01E-	AC 5F AA	LDY	\$AA5F	BASIC PGM EXECUTING?	
*A021-	20 5E A6	JSR	\$A65E	YES	
*A024-	90 0C	←BCC	\$A032	} DEFERRED COMMAND?	
*A026-	A9 02	LDA	##02		
*A028-	39 09 A9	AND	\$A909, Y		
*A02B-	F0 05	←BEQ	\$A032	NO	

DOS EXIT

RESTORE REGS

SKIP A LINE

COMMAND PARSE

COMMAND NAME LOOKUP



```

* A02D- A9 0F LDA ##0F } "NOT DIRECT COMMAND"
* A02F- 4C D2 A6 JMP $A6D2 }
* A032- C0 06 CPY ##06 RUN?
* A034- D0 02 BNE $A038
* A036- 84 33 STY $33 MAKE PROMPT NON PRINTING
A038- A9 20 LDA ##20
A03A- 39 09 A9 AND $A909, Y } FILENAME #1?
A03D- F0 61 BEQ $A0A0
A03F- 20 95 A0 JSR $A095 CLEAR FILENAME BUFFER
A042- 08 PHP (EQ)
A043- 20 A4 A1 JSR $A1A4 GET NEXT NON-BLANK
A046- F0 1E BEQ $A066 END?
A048- 0A ASL
A049- 90 05 BCC $A050 } IF FLASHING, NO GOOD
A04B- 30 03 BMI $A050
A04D- 4C 00 A0 JMP $A000
A050- 6A ROR
A051- 4C 59 A0 JMP $A059
A054- 20 93 A1 JSR $A193 ← NEXT INPUT CHAR
A057- F0 0D BEQ $A066 END?
A059- 99 75 AA STA $AA75, Y MOVE CHAR TO FILENAME BUFF
A05C- C8 INY NEXT
A05D- C0 3C CPY ##3C
A05F- 90 F3 BCC $A054
A061- 20 93 A1 JSR $A193 } FLUSH EXCESS FILENAME TO " "
A064- D0 FB BNE $A061
A066- 28 PLP } FINISHED BOTH NAMES?
A067- D0 0F BNE $A078
A069- AC 5F AA LDY $AA5F } FILENAME #2?
A06C- A9 10 LDA ##10
A06E- 39 09 A9 AND $A909, Y }
A071- F0 0C BEQ $A07F
A073- A0 1E LDY ##1E DISPLACEMENT TO IT
A075- 08 PHP (NE)
A076- D0 CB BNE $A043 PARSE IT
A078- AD 93 AA LDA $AA93 } FILENAME #2 = 3? (IF REQUIRED,)
A07B- C9 A0 CMP ##A0 "SYNTAX ERR"
A07D- F0 13 BEQ $A092
A07F- AD 75 AA LDA $AA75 } FILENAME #1 = 5?
A082- C9 A0 CMP ##A0
A084- D0 4B BNE $A0D1
A086- AC 5F AA LDY $AA5F } FILENAME OPTIONAL?
A089- A9 C0 LDA ##C0
A08B- 39 09 A9 AND $A909, Y }
A08E- F0 02 BEQ $A092 SYNTAX ERROR IF NOT OPTIONAL
A090- 10 3F BPL $A0D1 OPTIONAL, GO ON
A092- 4C 00 A0 JMP $A000 SYNTAX ERROR/OR PASS IT THROUGH
A095- A0 3C LDY ##3C 60 BLANK BOTH FILENAMES
A097- A9 A0 LDA ##A0 6
A099- 99 74 AA STA $AA74, Y } CLEAR 60 BYTES
A09C- 88 DEY
A09D- D0 FA BNE $A099
A09F- 60 RTS
A0A0- 8D 75 AA STA $AA75 NO FILENAME PARSED
A0A3- A9 0C LDA ##0C
A0A5- 39 09 A9 AND $A909, Y } NO POSITIONAL OPERAND?
A0A8- F0 27 BEQ $A0D1
A0AA- 20 B9 A1 JSR $A1B9 PARSE NUMERIC
A0AD- B0 1F BCS $A0CE SYNTAX ERROR IF NONE
A0AF- A8 TAY } ≥256? "RANGE ERROR"
A0B0- D0 17 BNE $A0C9 }
A0B2- E0 11 CPX ##11 }
A0B4- B0 13 BCS $A0C9 } ≥17? "RANGE ERROR"

```

FILENAMES PARSED

POSITIONAL NUMERIC

A0B6-	AC 5F AA	LDY	\$AA5F	} SLOT # ? (PR#, IN#)
A0B9-	A9 08	LDA	##08	
A0BB-	39 09 A9	AND	\$A909, Y	
A0BE-	F0 06	BEQ	\$A0C6	NO
A0C0-	E0 08	CPX	##08	} ≥ 8 ?
A0C2-	B0 CE	BCS	\$A092	
A0C4-	90 0B	BCC	\$A0D1	< 8
A0C6-	8A	TXA		} MAXFILES = 0 ? INVALID IF SO
A0C7-	D0 08	BNE	\$A0D1	
A0C9-	A9 02	LDA	##02	"RANGE ERROR"
A0CB-	4C D2 A6	JMP	\$A6D2	} "SYNTAX ERROR"
A0CE-	4C C4 A6	JMP	\$A6C4	
A0D1-	A9 00	LDA	##00	} SET DEFAULTS FOR KEYWORD OPERANDS
A0D3-	8D 65 AA	STA	\$AA65	
A0D6-	8D 74 AA	STA	\$AA74	
A0D9-	8D 66 AA	STA	\$AA66	
A0DC-	8D 6C AA	STA	\$AA6C	} SOME ADDITIONAL DEFAULTS
A0DF-	8D 6D AA	STA	\$AA6D	
A0E2-	20 DC BF	JSR	\$BFDC	GET LINE INDEX
A0E5-	AD 5D AA	LDA	\$AA5D	NEXT NON-BLANK
A0E8-	20 A4 A1	JSR	\$A1A4	} SKIP EXCESS COMMAS
A0EB-	D0 1F	BNE	\$A10C	
A0ED-	C9 8D	CMP	##8D	} END OF LINE, GET CMD # WHICH KEYWORDS WERE GIVEN? PLUS ALL VALID
A0EF-	D0 F7	BNE	\$A0E8	
A0F1-	AE 5F AA	LDX	\$AA5F	} WERE ANY KEYS GIVEN WHICH ARE UNEXPECTED FOR THIS CMD? RESCAN?
A0F4-	AD 65 AA	LDA	\$AA65	
A0F7-	1D 0A A9	ORA	\$A90A, X	} NORMAL COMMAND EXIT IF NOT CLEAR FLAG
A0FA-	5D 0A A9	EOR	\$A90A, X	
A0FD-	D0 93	BNE	\$A092	} SET LINE INDEX AND GO SCAN
A0FF-	AE 63 AA	LDX	\$AA63	
A102-	F0 76	BEQ	\$A17A	} 10 KEYWORDS
A104-	8D 63 AA	STA	\$AA63	
A107-	8E 5D AA	STX	\$AA5D	} LOOK UP KEYWORD IN TABLE
A10A-	D0 DC	BNE	\$A0E8	
A10C-	A2 0A	LDX	##0A	} NOT IN TABLE - "SYNTAX ERROR"
A10E-	DD 40 A9	CMP	\$A940, X	
A111-	F0 05	BEQ	\$A118	} GET ITS BIT POSITION NO NUMERIC VALUE (C, I or O)
A113-	CA	DEX		
A114-	D0 F8	BNE	\$A10E	} INDICATE THIS OPERAND PRESENT
A116-	F0 B6	BEQ	\$A0CE	
A118-	BD 4A A9	LDA	\$A94A, X	} KEYWORD INDEX SAVED
A11B-	30 47	BMI	\$A164	
A11D-	0D 65 AA	ORA	\$AA65	} GET NUMERIC VALUE "SYNTAX ERROR" IF BAD
A120-	8D 65 AA	STA	\$AA65	
A123-	CA	DEX		} Y = INDEX * 4
A124-	8E 64 AA	STX	\$AA64	
A127-	20 B9 A1	JSR	\$A1B9	} IS NUMBER WITHIN ALLOWABLE RANGE FOR THIS KEYWORD?
A12A-	B0 A2	BCS	\$A0CE	
A12C-	AD 64 AA	LDA	\$AA64	}
A12F-	0A	ASL		
A130-	0A	ASL		}
A131-	A8	TAY		
A132-	A5 45	LDA	\$45	}
A134-	D0 09	BNE	\$A13F	
A136-	A5 44	LDA	\$44	}
A138-	D9 55 A9	CMP	\$A955, Y	
A13B-	90 8C	BCC	\$A0C9	}
A13D-	A5 45	LDA	\$45	
A13F-	D9 58 A9	CMP	\$A958, Y	}
A142-	90 0B	BCC	\$A14F	
A144-	D0 83	BNE	\$A0C9	}
A146-	A5 44	LDA	\$44	
A148-	D9 57 A9	CMP	\$A957, Y	

KEYWORDS

A14B-	90 02	BCC	\$A14F	
A14D-	D0 F5	BNE	\$A144	
A14F-	AD 63 AA	LDA	\$AA63	FIRST SCAN (DON'T CHANGE KEYWORDS)?
A152-	D0 94	BNE	\$A0E8	YES
A154-	98	TYA		} KEYWORD INDEX AGAIN (*1)
A155-	4A	LSR		
A156-	A8	TAY		
A157-	A5 45	LDA	\$45	} SAVE VALUE IN TABLE
A159-	99 67 AA	STA	\$AA67, Y	
A15C-	A5 44	LDA	\$44	
A15E-	99 66 AA	STA	\$AA66, Y	
A161-	4C E8 A0	JMP	\$A0E8	GET NEXT KEYWORD
A164-	48	PHA		
A165-	A9 80	LDA	##80	} C, I or O parsed
A167-	0D 65 AA	ORA	\$AA65	
A16A-	8D 65 AA	STA	\$AA65	
A16D-	68	PLA		
A16E-	29 7F	AND	##7F	} UPDATE MON VALUE
A170-	0D 74 AA	ORA	\$AA74	
A173-	8D 74 AA	STA	\$AA74	
A176-	D0 E9	BNE	\$A161	GET NEXT KEYWORD
A178-	F0 9C	BEQ	\$A116	THIS CAN'T HAPPEN
A17A-	20 80 A1	JSR	\$A180	GO PROCESS COMMAND
A17D-	4C 83 9F	JMP	\$9F83	THEN EXIT VIA ECHO
A180-	20 5B A7	JSR	\$A75B	RESET STATE
A183-	20 AE A1	JSR	\$A1AE	CLEAR FIO PARMLIST
A186-	AD 5F AA	LDA	\$AA5F	COMMAND INDEX
A189-	AA	TAX		
A18A-	BD 1F 9D	LDA	\$9D1F, X	} PUT COMMAND HANDLER LOC ON STACK TO "RETURN" THRU IT
A18D-	48	PHA		
A18E-	BD 1E 9D	LDA	\$9D1E, X	
A191-	48	PHA		
A192-	60	RTS		GO TO COMMAND HANDLER
A193-	AE 5D AA	LDX	\$AA5D	GET LINE INDEX
A196-	BD 00 02	LDA	\$0200, X	GET NEXT CHARACTER
A199-	C9 8D	CMP	##8D	END OF LINE?
A19B-	F0 06	BEQ	\$A1A3	YES
A19D-	E8	INX		} UPDATE LINE INDEX
A19E-	8E 5D AA	STX	\$AA5D	
A1A1-	C9 AC	CMP	##AC	IS IT A COMMA?
A1A3-	60	RTS		OR CR?
A1A4-	20 93 A1	JSR	\$A193	GET NEXT CHARACTER
A1A7-	F0 FA	BEQ	\$A1A3	COMMA OR CR?
A1A9-	C9 A0	CMP	##A0	BLANK?
A1AB-	F0 F7	BEQ	\$A1A4	YES, SKIP IT
A1AD-	60	RTS		NO, NON BLANK
A1AE-	A9 00	LDA	##00	CLEAR FIO PARMLIST
A1B0-	A0 16	LDY	##16	
A1B2-	99 BA B5	STA	\$B5BA, Y	ZERO 22 BYTES
A1B5-	88	DEY		OF FIO PARMS
A1B6-	D0 FA	BNE	\$A1B2	
A1B8-	60	RTS		
A1B9-	A9 00	LDA	##00	} NUM = 0
A1BB-	85 44	STA	\$44	
A1BD-	85 45	STA	\$45	
A1BF-	20 A4 A1	JSR	\$A1A4	GET FIRST NON-0
A1C2-	08	PHP		REMEMBER STATUS
A1C3-	C9 A4	CMP	##A4	} "0" ?
A1C5-	F0 3C	BEQ	\$A203	
A1C7-	28	PLP		
A1C8-	4C CE A1	JMP	\$A1CE	NEXT CHAR
A1CB-	20 A4 A1	JSR	\$A1A4	
A1CE-	D0 06	BNE	\$A1D6	END?

DO COMMAND

GET NEXT CHAR ON LINE

FLUSH TO NE NON-BLANK

PARSE NUMERIC OPERAND

OUTPUT:  
X, A = VALUE

```

A1D0- A6 44 LDX $44
A1D2- A5 45 LDA $45
A1D4- 18 CLC
A1D5- 60 RTS
A1D6- 38 SEC
A1D7- E9 B0 SBC #$B0
A1D9- 30 21 BMI $A1FC
A1DB- C9 0A CMP #$0A
A1DD- B0 1D BCS $A1FC
A1DF- 20 FE A1 JSR $A1FE
A1E2- 65 44 ADC $44
A1E4- AA TAX
A1E5- A9 00 LDA #$00
A1E7- 65 45 ADC $45
A1E9- AB TAY
A1EA- 20 FE A1 JSR $A1FE
A1ED- 20 FE A1 JSR $A1FE
A1F0- 8A TXA
A1F1- 65 44 ADC $44
A1F3- 85 44 STA $44
A1F5- 98 TYA
A1F6- 65 45 ADC $45
A1F8- 85 45 STA $45
A1FA- 90 CF BCC $A1CB
A1FC- 38 SEC
A1FD- 60 RTS

```

GET NUM

EXIT

DECIMAL  
CONVERT

} CONVERT TO BINARY  
BAD!

} TOO HIGH?

NUM = NUM \* 2

X,Y = NUM + DIGIT

NUM =  
NUM \* 10  
+ DIGIT

NUM = NUM \* 4

NUM = NUM + X,Y

CONTINUE  
INVALID NUMERIC EXIT

```

A1FE- 06 44 ASL $44
A200- 26 45 ROL $45
A202- 60 RTS

```

ROTATE NUM

```

A203- 28 PLP
A204- 20 A4 A1 JSR $A1A4
A207- F0 C5 BEQ $A1CE
A209- 38 SEC
A20A- E9 B0 SBC #$B0
A20C- 30 EE BMI $A1FC
A20E- C9 0A CMP #$0A
A210- 90 08 BCC $A21A
A212- E9 07 SBC #$07
A214- 30 E6 BMI $A1FC
A216- C9 10 CMP #$10
A218- B0 E2 BCS $A1FC
A21A- A2 04 LDX #$04
A21C- 20 FE A1 JSR $A1FE
A21F- CA DEX
A220- D0 FA BNE $A21C
A222- 05 44 ORA $44
A224- 85 44 STA $44
A226- 4C 04 A2 JMP $A204

```

PARSE HEX  
NUMBER

GET NEXT CHARACTER  
END?

CONVERT DIGIT TO BINARY

} VALIDITY CHECK NUMERICS

CONVERT & CHECK A-F

SHIFT NUM UP 1 NIBBLE

} ADD IN THIS DIGIT

Go GET NEXT

```

A229- A5 44 LDA $44
A22B- 4C 95 FE JMP $FE95
A22E- A5 44 LDA $44
A230- 4C 8B FE JMP $FE8B

```

PR#

IN#

GET NUMERIC VALUE  
EXIT THRU "OUTPORT" IN ROM

GET NUMERIC VALUE  
EXIT THRU "INPORT" IN ROM

```

A233- AD 5E AA LDA $AA5E
A236- 0D 74 AA ORA $AA74
A239- 8D 5E AA STA $AA5E
A23C- 60 RTS

```

} ADD NEW MON FLAGS  
TO OLD

MON

```

A23D- 2C 74 AA BIT $AA74
A240- 50 03 BVC $A245

```

} 40 SET?

NOMON

```

A242- 20 C8 9F JSR $9FC8
A245- A9 70 LDA #$70
A247- 4D 74 AA EOR $AA74
A24A- 2D 5E AA AND $AA5E
A24D- 8D 5E AA STA $AA5E

```

YES, WILL ECHO THIS SO GO TO A NEW LINE

} MAKE MASK

} TURN OFF DESIRED BITS



A250-	60		RTS			
A251-	A9 00		LDA	##00	} NO EXEC FILE NOW	MAXFILES
A253-	8D B3 AA		STA	##AAB3		
A256-	A5 44		LDA	##44	GET NUMERIC VALUE	
A258-	48		PHA			
A259-	20 16 A3		JSR	##A316	CLOSE ALL OPEN FILES	
A25C-	68		PLA			
A25D-	8D 57 AA		STA	##AA57	SET NEW MAXFILES #	
A260-	4C D4 A7		JMP	##A7D4	REINITIALIZE FILE BUFFERS/THEN EXIT	
A263-	A9 05		LDA	##05	DELETE OPCODE	DELETE
A265-	20 AA A2		JSR	##A2AA	GO DO IT	
A268-	20 64 A7		JSR	##A764	LOCATE BUFFER USED	
A26B-	A0 00		LDY	##00	} AND FREE IT	
A26D-	98		TYA			
A26E-	91 40		STA	(#40),Y		
A270-	60		RTS			
A271-	A9 07		LDA	##07	LOCK OPCODE	LOCK
A273-	D0 02		BNE	##A277		
A275-	A9 08		LDA	##08	UNLOCK	UNLOCK
A277-	20 AA A2		JSR	##A2AA	CALL FIO TO DO IT	
A27A-	4C EA A2		JMP	##A2EA	NOW CLOSE	
A27D-	A9 0C		LDA	##0C	VERIFY OPCODE	VERIFY
A27F-	D0 F6		BNE	##A277		
A281-	AD 08 9D		LDA	##9D08	} PARM1 = ↑ FILENAME #2	RENAME
A284-	8D BD B5		STA	##B5BD		
A287-	AD 09 9D		LDA	##9D09		
A28A-	8D BE B5		STA	##B5BE		
A28D-	A9 09		LDA	##09	RENAME OPCODE	
A28F-	8D 63 AA		STA	##AA63		
A292-	20 C8 A2		JSR	##A2C8	CALL FIO	
A295-	4C EA A2		JMP	##A2EA	AND CLOSE	
A298-	20 A3 A2		JSR	##A2A3	OPEN FILE	APPEND
A29B-	20 8C A6		JSR	##A68C	} READ FILE TO FIRST 00	
A29E-	D0 FB		BNE	##A29B		
A2A0-	4C 46 A5		JMP	##A546	(DOS 3.3 PATCH)	
*A2A3-	A9 00		LDA	##00	OPEN A TEXT FILE ONLY	OPEN
*A2A5-	4C D5 A3		JMP	##A3D5		
A2A8-	A9 01		LDA	##01	} OPCODE = OPEN	
→A2AA-	8D 63 AA		STA	##AA63		
A2AD-	AD 6C AA		LDA	##AA6C	} IF NO L GIVEN, USE 0001	
A2B0-	D0 0A		BNE	##A2BC		
A2B2-	AD 6D AA		LDA	##AA6D		
A2B5-	D0 05		BNE	##A2BC		
A2B7-	A9 01		LDA	##01		
A2B9-	8D 6C AA		STA	##AA6C	} PARM1 = L VALUE	
A2BC-	AD 6C AA		→LDA	##AA6C		
A2BF-	8D BD B5		STA	##B5BD		
A2C2-	AD 6D AA		LDA	##AA6D		
A2C5-	8D BE B5		STA	##B5BE		
-A2C8-	20 EA A2		JSR	##A2EA	CLOSE IT IF OPEN ALREADY	
A2CB-	A5 45		LDA	##45	} FOUND A BUFFER?	
A2CD-	D0 03		BNE	##A2D2		
A2CF-	4C C8 A6		JMP	##A6C8	"NO FILE BUFS AVAILABLE"	
A2D2-	85 41		→STA	##41	} [A40] → BUFFER	
A2D4-	A5 44		LDA	##44		
A2D6-	85 40		STA	##40		
A2D8-	20 43 A7		JSR	##A743	COPY FILENAME TO BUFFER (ALLOCATES IT)	
A2DB-	20 4E A7		JSR	##A74E	COPY BUFFER PTRS TO FIO PARM LIST	
A2DE-	20 1A A7		JSR	##A71A	FINISH FIO PARMLIST (V,D,S, FILENAME↑)	
A2E1-	AD 63 AA		LDA	##AA63	} SET OPERATION CODE	
A2E4-	8D BB B5		STA	##B5BB		
A2E7-	4C A8 A6		JMP	##A6A8	EXIT THROUGH FIO DRIVER	
A2EA-	AD 75 AA		LDA	##AA75	FIRST CHR OF FILE	CLOSE

A2ED- C9 A0 CMP ##A0 NO FILENAME?  
 A2EF- F0 25 BEQ \$A316 THEN CLOSE ALL FILES  
 A2F1- 20 64 A7 JSR \$A764 OTHERWISE, FIND OPEN FILE BUFFER  
 A2F4- B0 3A BCS \$A330 NO SUCH FILE OPEN? IGNORE IT.  
 A2F6- 20 FC A2 JSR \$A2FC CLOSE FILE /FREE BUFFER  
 A2F9- 4C EA A2 JMP \$A2EA NOW MAKE SURE THERE AREN'T TWO

A2FC- 20 AF A7 JSR \$A7AF IS THIS BUFFER EXEC?  
 A2FF- D0 05 BNE \$A306 NO  
 A301- A9 00 LDA ##00 } YES, TURN EXEC OFF FIRST  
 A303- 8D B3 AA STA \$AAB3 }  
 A306- A0 00 LDY ##00 }  
 A308- 98 TYA { RELEASE BUFFER  
 A309- 91 40 STA (\$40),Y }  
 A30B- 20 4E A7 JSR \$A74E FILE PTRS TO FIO PARPLANEA  
 A30E- A9 02 LDA ##02 } CLOSE/FIO OPCODE  
 A310- 8D BB B5 STA \$B5BB }  
 A313- 4C AB A6 JMP \$A6AB EXIT THRU FIO

CLOSE FILE

A316- 20 92 A7 JSR \$A792 GET FIRST FILE BUFFER  
 A319- D0 05 BNE \$A320 ALWAYS THERE  
 A31B- 20 9A A7 JSR \$A79A YES, GET NEXT  
 A31E- F0 10 BEQ \$A330 END OF CHAIN?  
 A320- 20 AF A7 JSR \$A7AF IS IT EXEC?  
 A323- F0 F6 BEQ \$A31B YES, SKIP IT  
 A325- 20 AA A7 JSR \$A7AA GET FIRST FILENAME CHAR  
 A328- F0 F1 BEQ \$A31B CLOSED, SKIP IT  
 A32A- 20 FC A2 JSR \$A2FC OPEN, SO CLOSE IT  
 A32D- 4C 16 A3 JMP \$A316 START ALL OVER  
 A330- 60 RTS

CLOSE ALL FILES

A331- A9 09 LDA ##09  
 A333- 2D 65 AA AND \$AA65 } "A" and "L" operands given?  
 A336- C9 09 CMP ##09 }  
 A338- F0 03 BEQ \$A33D }  
 A33A- 4C 00 A0 JMP \$A000 NO, SYNTAX ERROR  
 A33D- A9 04 LDA ##04 } OPEN & CHECK B TYPE FILE  
 A33F- 20 D5 A3 JSR \$A3D5 }  
 A342- AD 73 AA LDA \$AA73 } GET "A" VALUE  
 A345- AC 72 AA LDY \$AA72 }  
 A348- 20 E0 A3 JSR \$A3E0 WRITE IT  
 A34B- AD 6D AA LDA \$AA6D } GET "L" VALUE  
 A34E- AC 6C AA LDY \$AA6C }  
 A351- 20 E0 A3 JSR \$A3E0 WRITE IT  
 A354- AD 73 AA LDA \$AA73 } GET "A" AGAIN  
 A357- AC 72 AA LDY \$AA72 }  
 A35A- 4C FF A3 JMP \$A3FF GO WRITE A RANGE

BSAVE

A35D- 20 A8 A2 JSR \$A2A8 OPEN FILE  
 A360- A9 7F LDA #\$7F } GET FILE TYPE  
 A362- 2D C2 B5 AND \$B5C2 }  
 A365- C9 04 CMP ##04 } B TYPE FILE?  
 A367- F0 03 BEQ \$A36C }  
 A369- 4C D0 A6 JMP \$A6D0 NO, "FILE TYPE MISMATCH"  
 A36C- A9 04 LDA ##04 } OPEN & TEST FILE TYPE  
 A36E- 20 D5 A3 JSR \$A3D5 }  
 A371- 20 7A A4 JSR \$A47A READ ADDRESS  
 A374- AA TAX SAVE A HIGH  
 A375- AD 65 AA LDA \$AA65 }  
 A378- 29 01 AND ##01 } "A" KEYWORD GIVEN?  
 A37A- D0 06 BNE \$A382 }  
 A37C- 8E 72 AA STX \$AA72 } NO, USE VALUE JUST READ  
 A37F- 8C 73 AA STY \$AA73 }  
 A382- 20 7A A4 JSR \$A47A READ LENGTH  
 A385- AE 72 AA LDX \$AA72 } GET ADDRESS  
 A388- AC 73 AA LDY \$AA73 }  
 A38B- 4C 71 A4 JMP \$A471 GO READ RANGE OF DATA

BLOAD



A38E-	20 5D A3	JSR	\$A35D	BLOAD FILE	BRUN
A391-	20 51 A8	JSR	\$A851	SET DOS INTERCEPTS	
A394-	6C 72 AA	JMP	(#AA72)	THEN JUMP TO [A]	
A397-	AD B6 AA	LDA	\$AAB6	GET BASIC TYPE	SAVE
A39A-	F0 20	BEQ	\$A3BC	INT?	
A39C-	A5 D6	LDA	\$D6	FP, TOO BIG?	
A39E-	10 03	BPL	\$A3A3		
A3A0-	4C CC A6	JMP	\$A6CC	"PROGRAM TOO LARGE"	
A3A3-	A9 02	LDA	##02	} OPEN AND TEST FOR "A" TYPE FILE	
A3A5-	20 D5 A3	JSR	\$A3D5		
A3A8-	38	SEC			
A3A9-	A5 AF	LDA	\$AF	} COMPUTE PROGRAM LENGTH (PGMEND - LOMEM)	
A3AB-	E5 67	SBC	\$67		
A3AD-	A8	TAY			
A3AE-	A5 B0	LDA	\$B0		
A3B0-	E5 68	SBC	\$68		
A3B2-	20 E0 A3	JSR	\$A3E0	WRITE LENGTH	
A3B5-	A5 68	LDA	\$68	} WRITE A RANGE FROM LOMEM/EXIT	
A3B7-	A4 67	LDY	\$67		
A3B9-	4C FF A3	JMP	\$A3FF		
A3BC-	A9 01	LDA	##01	} OPEN AND TEST FOR "I" TYPE FILE	
A3BE-	20 D5 A3	JSR	\$A3D5		
A3C1-	38	SEC			
A3C2-	A5 4C	LDA	\$4C	} COMPUTE PROGRAM LENGTH (HIMEM - PP)	
A3C4-	E5 CA	SBC	\$CA		
A3C6-	A8	TAY			
A3C7-	A5 4D	LDA	\$4D		
A3C9-	E5 CB	SBC	\$CB		
A3CB-	20 E0 A3	JSR	\$A3E0	WRITE LENGTH	
A3CE-	A5 CB	LDA	\$CB	} WRITE RANGE FROM PP/EXIT	
A3D0-	A4 CA	LDY	\$CA		
A3D2-	4C FF A3	JMP	\$A3FF		
A3D5-	8D C2 B5	STA	\$B5C2	SET FILE TYPE WANTED	OPEN & TEST FILE TYPE
A3D8-	48	PHA			
A3D9-	20 A8 A2	JSR	\$A2A8	OPEN FILE	
A3DC-	68	PLA			
A3DD-	4C C4 A7	JMP	\$A7C4	GO CHECK IT	
A3E0-	8C C1 B5	STY	\$B5C1	} SAVE VALUE IN LENGTH PARAM WRITE LOW BYTE FIRST	WRITE A 2 BYTE VALU
A3E3-	8C C3 B5	STY	\$B5C3		
A3E6-	8D C2 B5	STA	\$B5C2		
A3E9-	A9 04	LDA	##04	} WRITE OPCODE	
A3EB-	8D BB B5	STA	\$B5BB		
A3EE-	A9 01	LDA	##01	} DO ONE BYTE	
A3F0-	8D BC B5	STA	\$B5BC		
A3F3-	20 A8 A6	JSR	\$A6A8	CALL FIO	
A3F6-	AD C2 B5	LDA	\$B5C2	} NOW DO HIGH BYTE	
A3F9-	8D C3 B5	STA	\$B5C3		
A3FC-	4C A8 A6	JMP	\$A6A8	WRITE IT AND EXIT	
A3FF-	8C C3 B5	STY	\$B5C3	} SET ADDRESS	I/O A RANG
A402-	8D C4 B5	STA	\$B5C4		
A405-	A9 02	LDA	##02	} SET SUBCODE TO I/O A RANGE OF BYTES	
A407-	8D BC B5	STA	\$B5BC		
A40A-	20 A8 A6	JSR	\$A6A8	CALL FIO	
A40D-	4C EA A2	JMP	\$A2EA	THEN CLOSE	
A410-	4C D0 A6	JMP	\$A6D0	"FILE TYPE MISMATCH"	
A413-	20 16 A3	JSR	\$A316	CLOSE ALL FILES	LOAD
A416-	20 A8 A2	JSR	\$A2A8	OPEN THE FILE	
A419-	A9 23	LDA	##23	} VALID FILE TYPE?	
A41B-	2D C2 B5	AND	\$B5C2		
A41E-	F0 F0	BEQ	\$A410	NO, "FILE TYPE MISMATCH"	
A420-	8D C2 B5	STA	\$B5C2		
A423-	AD B6 AA	LDA	\$AAB6	WHICH BASIC IS ACTIVE	
A426-	F0 28	BEQ	\$A450	INT	



A428-	A9 02	LDA	##02	WE WANT FP BASIC
A42A-	20 B1 A4	JSR	##A4B1	GO SET IT UP
A42D-	20 7A A4	JSR	##A47A	READ LENGTH OF PGM
A430-	18	CLC		
A431-	65 67	ADC	##67	} ADD LENGTH TO LOMEM/PGM START
A433-	AA	TAX		
A434-	98	TYA		
A435-	65 68	ADC	##68	
A437-	C5 74	CMP	##74	} BEYOND HIMEM?
A439-	B0 70	BCS	##A4AB	
A43B-	85 B0	STA	##B0	← SET PGM HIGH
A43D-	85 6A	STA	##6A	← AND BEGINNING OF VARIABLES
A43F-	86 AF	STX	##AF	
A441-	86 69	STX	##69	
A443-	A6 67	LDX	##67	} SET STARTING ADDRESS
A445-	A4 68	LDY	##68	
A447-	20 71 A4	JSR	##A471	READ RANGE INTO MEMORY
A44A-	20 51 A8	JSR	##A851	SET DOS INTERCEPTS
* A44D-	6C 60 9D	JMP	(##9D60)	RELOCATE PROGRAM TO THIS BASIC

A450-	A9 01	→ LDA	##01	} SET INT BASIC
A452-	20 B1 A4	JSR	##A4B1	
A455-	20 7A A4	JSR	##A47A	READ LENGTH
A458-	38	SEC		
A459-	A5 4C	LDA	##4C	} HIMEM - LENGTH GIVES PGM START
A45B-	ED 60 AA	SBC	##AA60	
A45E-	AA	TAX		
A45F-	A5 4D	LDA	##4D	
A461-	ED 61 AA	SBC	##AA61	
A464-	90 45	BCC	##A4AB	TOO LARGE
A466-	A8	TAY		
A467-	C4 4B	CPY	##4B	} PAST LOMEM?
A469-	90 40	BCC	##A4AB	
A46B-	F0 3E	BEQ	##A4AB	
A46D-	84 CB	STY	##CB	} SET PGM START
A46F-	86 CA	STX	##CA	

A471-	8E C3 B5	STX	##B5C3	} SET STARTING ADDRESS
A474-	8C C4 B5	STY	##B5C4	
A477-	4C 0A A4	JMP	##A40A	GO READ PGM INTO MEMORY

A47A-	AD 0A 9D	LDA	##9D0A	} READ INTO ##AA60 (RANGE LENGTH)
A47D-	8D C3 B5	STA	##B5C3	
A480-	AD 0B 9D	LDA	##9D0B	
A483-	8D C4 B5	STA	##B5C4	
A486-	A9 00	LDA	##00	} READ 2 BYTES
A488-	8D C2 B5	STA	##B5C2	
A48B-	A9 02	LDA	##02	
A48D-	8D C1 B5	STA	##B5C1	} READ A RANGE OF BYTES
A490-	A9 03	LDA	##03	
A492-	8D BB B5	STA	##B5BB	
A495-	A9 02	LDA	##02	
A497-	8D BC B5	STA	##B5BC	CALL FIO
A49A-	20 A8 A6	JSR	##A6A8	
A49D-	AD 61 AA	LDA	##AA61	} SET RANGE LENGTH IN FIO PARM LIST
A4A0-	8D C2 B5	STA	##B5C2	
A4A3-	A8	TAY		
A4A4-	AD 60 AA	LDA	##AA60	
A4A7-	8D C1 B5	STA	##B5C1	
A4AA-	60	RTS		

A4AB-	20 EA A2	JSR	##A2EA	CLOSE FILE
A4AE-	4C CC A6	JMP	##A6CC	"PROGRAM TO LARGE"

A4B1-	CD C2 B5	CMP	##B5C2	} ALREADY IN PROPER BASIC?
A4B4-	F0 1A	BEQ	##A4D0	
A4B6-	AE 5F AA	LDX	##AA5F	} REMEMBER CURRENT COMMAND
A4B9-	8E 62 AA	STX	##AA62	

FP LOAD

INT LOAD

READ LENGTH  
VALUE FROM  
FILE

SET PROPE  
BASIC

```

A4BC- 4A LSR
A4BD- F0 03 BEQ $A4C2 } INT?
A4BF- 4C 9E A5 JMP $A59E NO, GO SET IT
A4C2- A2 1D LDX #1D
A4C4- BD 75 AA LDA $AA75,X } COPY PRIMARY FILENAME
A4C7- 9D 93 AA STA $AA93,X } TO SECONDARY
A4CA- CA DEX
A4CB- 10 F7 BPL $A4C4
A4CD- 4C 7A A5 JMP $A57A GO SET FP
A4D0- 60 RTS

```

```

* A4D1- AD B6 AA LDA $AAB6 } FP? RUN
* A4D4- F0 03 BEQ $A4D9
* A4D6- 8D B7 AA STA $AAB7 IF SO SIGNAL SPECIAL EXIT FROM STATE = 0
A4D9- 20 13 A4 JSR $A413 LOAD PROGRAM
A4DC- 20 C8 9F JSR $9FC8 SKIP A LINE
A4DF- 20 51 A8 JSR $A851 DOS INTERCEPTS BACK
A4E2- 6C 58 9D JMP ($9D58) RUN BASIC P6M

```

```

A4E5- A5 4A LDA $4A } DELETE ALL
A4E7- 85 CC STA $CC } VARIABLES
A4E9- A5 4B LDA $4B
A4EB- 85 CD STA $CD
A4ED- 6C 56 9D JMP ($9D56) INT BASIC "CHAIN" EPA

```

```

A4F0- 20 16 A4 JSR $A416 LOAD IT CHAIN
A4F3- 20 C8 9F JSR $9FC8 NEW LINE
A4F6- 20 51 A8 JSR $A851 SET DOS INTERCEPTS
A4F9- 6C 56 9D JMP ($9D56) GO TO BASIC CHAIN EPA

```

```

A4FC- 20 65 D6 JSR $D665 CLEAR VARIABLES FP ROM RUN
A4FF- 85 33 STA $33 } SET PROMPT
A501- 85 D8 STA $D8 } AND ONERR
A503- 4C D2 D7 JMP $D7D2 GO RUN

```

```

A506- 20 65 0E JSR $0E65 } SAME AS ABOVE FP RAM RUN
A509- 85 33 STA $33
A50B- 85 D8 STA $D8
A50D- 4C D4 0F JMP $0FD4

```

```

A510- 20 26 A5 JSR $A526 WRITE
A513- A9 05 LDA #$05 } SET CSWL WRITE STATE
A515- 8D 52 AA STA $AA52
A518- 4C 83 9F JMP $9F83 EXIT DOS

```

```

A51B- 20 26 A5 JSR $A526 READ
A51E- A9 01 LDA #$01 } SET READ MODE
A520- 8D 51 AA STA $AA51
A523- 4C 83 9F JMP $9F83 EXIT DOS

```

```

A526- 20 64 A7 JSR $A764 FIND FILE BUFFER
A529- 90 06 BCC $A531 OK
A52B- 20 A3 A2 JSR $A2A3 OPEN IT IF NOT ALREADY
A52E- 4C 34 A5 JMP $A534

```

```

A531- 20 4E A7 JSR $A74E SET FIO BUFF PTRS
A534- AD 65 AA LDA $AA65
A537- 29 06 AND #$06 } R or B GIVEN?
A539- F0 13 BEQ $A54E

```

```

A53B- A2 03 LDX #$03
A53D- BD 6E AA LDA $AA6E,X } YES, COPY TO FIO PARMLIST
A540- 9D BD B5 STA $B5BD,X
A543- CA DEX

```

```

A544- 10 F7 BPL $A53D
A546- A9 0A LDA #$0A } POSITION CALL
A548- 8D BB B5 STA $B5BB
A54B- 20 A8 A6 JSR $A6A8 To FIO
A54E- 60 RTS

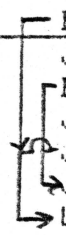
```

```

A54F- A9 40 LDA #$40 } V GIVEN? INIT
A551- 2D 65 AA AND $AA65
A554- F0 05 BEQ $A55B
A556- AD 66 AA LDA $AA66 YES, GET IT

```

A559-	D0 05	BNE	#\$A560	VALID?	
A55B-	A9 FE	LDA	##FE	IF NOT, USE 254	
A55D-	8D 66 AA	STA	##AA66	REPLACE V VALUE	
A560-	AD 0D 9D	LDA	##9D0D	} SUBCODE = FIRST PAGE OF DOS IMAGE	
A563-	8D BC B5	STA	##B5BC		
A566-	A9 0B	LDA	##0B	} INIT CALL TO FIO	
A568-	20 AA A2	JSR	##A2AA		
A56B-	4C 97 A3	JMP	##A397	EXIT BY SAVING GREETING PROG	
A56E-	A9 06	LDA	##06	} CATALOG CALL TO FIO	CATALOG
A570-	20 AA A2	JSR	##A2AA		
A573-	AD BF B5	LDA	##B5BF	} SET NEW V VALUE	
A576-	8D 66 AA	STA	##AA66		
A579-	60	RTS			
A57A-	A9 4C	LDA	##4C	} SET ROM CARD FOR FP	FP
A57C-	20 B2 A5	JSR	##A5B2		
A57F-	F0 2E	BEQ	##A5AF	GOT IT, DONE, COLDSTART DOS	
A581-	A9 00	LDA	##00	} WE ARE IN INT	
A583-	8D B6 AA	STA	##AAB6		
A586-	A0 1E	LDY	##1E	} BLANK PRIMARY FILENAME	
A588-	20 97 A0	JSR	##A097		
A58B-	A2 09	LDX	##09	} COPY "APPLESOFT" TO PRIMARY FILENAME	
A58D-	BD B7 AA	LDA	##AAB7,X		
A590-	9D 74 AA	STA	##AA74,X		
A593-	CA	DEX			
A594-	D0 F7	BNE	##A58D	} COLDSTART WITH RAM FP	
A596-	A9 C0	LDA	##C0		
A598-	8D 51 AA	STA	##AA51		
A59B-	4C D1 A4	JMP	##A4D1		RUN APPLESOFT
A59E-	A9 20	LDA	##20	} SET ROM FOR INT	INT
A5A0-	20 B2 A5	JSR	##A5B2		
A5A3-	F0 05	BEQ	##A5AA	GOT IT	
*A5A5-	A9 01	LDA	##01	} "LANGUAGE NOT AVAILABLE"	
*A5A7-	4C D2 A6	JMP	##A6D2		
*A5AA-	A9 00	LDA	##00	} RUN NOT INTERCEPTED	
*A5AC-	8D B7 AA	STA	##AAB7		
A5AF-	4C 84 9D	JMP	##9D84	COLDSTART DOS	
A5B2-	CD 00 E0	CMP	##E000	ALREADY SET?	SET ROM TO DESIRED BASIC
A5B5-	F0 0E	BEQ	##A5C5	YES	
A5B7-	8D 80 C0	STA	##C080	NO, TRY ROM CARD IN	NOTE: THIS ROUTINE WORKS REGARDLESS OF WHICH BASIC IS ON BOARD.
A5BA-	CD 00 E0	CMP	##E000	GOT IT?	
A5BD-	F0 06	BEQ	##A5C5	YES	
A5BF-	8D 81 C0	STA	##C081	NO, TRY ROM CARD OUT	
A5C2-	CD 00 E0	CMP	##E000	LAST CHANCE	
A5C5-	60	RTS			
A5C6-	20 A3 A2	JSR	##A2A3	OPEN THE FILE	EXEC
A5C9-	AD 4F AA	LDA	##AA4F	} COPY BUFFER ADDRESS TO EXEC BUFF PTR	
A5CC-	8D B4 AA	STA	##AAB4		
A5CF-	AD 50 AA	LDA	##AA50		
A5D2-	8D B5 AA	STA	##AAB5	} EXEC FILE ACTIVE	
A5D5-	AD 75 AA	LDA	##AA75		
A5D8-	8D B3 AA	STA	##AAB3	} GO SKIP "R" LINES	
A5DB-	D0 0E	BNE	##A5EB		
A5DD-	20 64 A7	JSR	##A764	LOCATE OPEN FILE	POSITION
A5E0-	90 06	BCC	##A5E8	OK?	
A5E2-	20 A3 A2	JSR	##A2A3	OPEN AS A TEXT FILE	
A5E5-	4C EB A5	JMP	##A5EB	} COPY BUFF PTRS TO FIO PARMLIST	
A5E8-	20 4E A7	JSR	##A74E		
A5EB-	AD 65 AA	LDA	##AA65	} R GIVEN AS KEYWORD? NO, DONE	
A5EE-	29 04	AND	##04		
A5F0-	F0 1B	BEQ	##A60D		
A5F2-	AD 6E AA	LDA	##AA6E		
A5F5-	D0 08	BNE	##A5FF		
A5F7-	AE 6F AA	LDX	##AA6F		



A5FA-	F0 11	BEQ	\$A60D	} DECREMENT R VALUE AND COMPARE	
A5FC-	CE 6F AA	DEC	\$AA6F		
A5FF-	CE 6E AA	DEC	\$AA6E		
A602-	20 8C A6	} JSR	\$A68C	READ NEXT BYTE	
A605-	F0 38		BEQ	\$A63F	END OF FILE? SAY SO
A607-	C9 8D	CMP	##8D	END OF LINE?	
A609-	D0 F7	BNE	\$A602		
A60B-	F0 E5	BEQ	\$A5F2	YES	
A60D-	60	RTS			
A60E-	20 5E A6	JSR	\$A65E	} BASIC EXECUTING? MUST BE	WRITE A BYTE TO FILE
A611-	B0 66	BCS	\$A679		
A613-	AD 5C AA	LDA	\$AA5C	GET BYTE	
A616-	8D C3 B5	STA	\$B5C3	PASS TO FIO	
A619-	A9 04	LDA	##04	} WRITE ONE BYTE	
A61B-	8D BB B5	STA	\$B5BB		
A61E-	A9 01	LDA	##01		
A620-	8D BC B5	STA	\$B5BC		
A623-	4C A8 A6	JMP	\$A6A8	CALL FIO AND EXIT	
A626-	20 5E A6	JSR	\$A65E	IS A BASIC PROGRAM RUNNING?	GET DISK INPUT BYTE
A629-	B0 4E	BCS	\$A679	NO, CLOSE THIS FILE/RESET	
A62B-	A9 06	LDA	##06	} STATE = 6	EXEC ENTRY
A62D-	8D 52 AA	STA	\$AA52		
A630-	20 8C A6	JSR	\$A68C	READ NEXT DISK BYTE	
A633-	D0 0F	BNE	\$A644	NOT END OF FILE...	
A635-	20 FC A2	JSR	\$A2FC	CLOSE FILE	
A638-	A9 03	LDA	##03	} STATE 3 (EXEC, not READ)?	
A63A-	CD 52 AA	CMP	\$AA52		
A63D-	F0 CE	BEQ	\$A60D	YES, NO MSG	
A63F-	A9 05	LDA	##05	} ELSE SAY "END OF DATA"	
A641-	4C D2 A6	JMP	\$A6D2		
* A644-	C9 E0	CMP	##E0	LOWER CASE?	
* A646-	90 02	BCC	\$A64A		
* A648-	29 7F	AND	##7F	IF SO, HIGH BIT OFF TO FOOL GETIN	
A64A-	8D 5C AA	STA	\$AA5C	SET HIS ACC VALUE	
* A64D-	AE 5A AA	LDX	\$AA5A	GET INPUT INDEX	
* A650-	F0 09	BEQ	\$A65B	START OF LINE	
* A652-	CA	DEX			
* A653-	BD 00 02	LDA	\$0200,X	} FORCE HIGH BIT ON IN PREVIOUS IN CASE IT WAS LOWER CASE	
* A656-	09 80	ORA	##80		
* A658-	9D 00 02	STA	\$0200,X		
A65B-	4C B3 9F	JMP	\$9FB3	DOS EXIT	
A65E-	48	PHA			
A65F-	AD B6 AA	LDA	\$AAB6		BASIC IN IMMEDIATE OR EXECUTION
A662-	F0 0E	BEQ	\$A672	INT BASIC?	
A664-	A6 76	LDX	\$76	NO, FP LINE NO. > 65280?	
* A666-	E8	INX		YES, NOT RUNNING	
A667-	F0 0D	BEQ	\$A676	PROMPT IS "J"?	
A669-	A6 33	LDX	\$33		
A66B-	E0 DD	CPX	##DD	YES	
A66D-	F0 07	BEQ	\$A676		
A66F-	68	PLA		BASIC IS "RUNNING" (CLC)	
A670-	18	CLC			
A671-	60	RTS			
A672-	A5 D9	LDA	\$D9	INTEGER BASIC RUNNING?	
A674-	30 F9	BMI	\$A66F		
A676-	68	PLA		BASIC IS IN IMMEDIATE MODE (SEC)	
A677-	38	SEC			
A678-	60	RTS			
A679-	20 FC A2	JSR	\$A2FC	CLOSE CURRENT FILE	
A67C-	20 5B A7	JSR	\$A75B	WARMSTART/STATE = 0	
A67E-	4C B3 9F	JMP	\$9FB3	EXIT DOS	
A682-	20 9D A6	JSR	\$A69D	SELECT EXEC FILE	EXEC READ
A685-	20 4E A7	JSR	\$A74E	COPY FILE BUFF PTRS	



```

A688- A9 03 LDA #03
A68A- D0 A1 BNE $A62D } SET STATE = 3 AND GO READ TEXT BYTE
A68C- A9 03 LDA #03
A68E- 8D BB B5 STA $B5BB } READ 1 BYTE
A691- A9 01 LDA #01
A693- 8D BC B5 STA $B5BC
A696- 20 A8 A6 JSR $A6A8 CALL FILE I/O
A699- AD C3 B5 LDA $B5C3 GET BYTE READ
A69C- 60 RTS

```

```

A69D- AD B5 AA LDA $AAB5 } POINT TO EXEC BUFF
A6A0- 85 41 STA $41
A6A2- AD B4 AA LDA $AAB4 } [040] -> EXEC FILE
A6A5- 85 40 STA $40
A6A7- 60 RTS

```

```

A6A8- 20 06 AB JSR $AB06 CALL FIO
A6AB- 90 16 BCC $A6C3 OK, EXIT
A6AD- 20 64 A7 JSR $A764 POINT [040] -> FILE BUFFER
A6B0- B0 05 BCS $A6B7
A6B2- A9 00 LDA #00
A6B4- A8 TAY
A6B5- 91 40 STA ($40),Y
A6B7- AD C5 B5 LDA $B5C5
A6BA- C9 05 CMP #05
A6BC- D0 14 BNE $A6D2 } FORCE FILE / RELEASE BUFFER
A6BE- A2 00 LDX #00 } CLOSED
A6C0- 8E C3 B5 STX $B5C3 } END OF FILE?
A6C3- 60 RTS } NO, PRINT ERROR MESSAGE

```

```

A6C4- A9 0B LDA #0B } "COMMAND SYNTAX ERROR"
A6C6- D0 0A BNE $A6D2 }
A6C8- A9 0C LDA #0C } "NO FILE BUFFS AVAIL"
A6CA- D0 06 BNE $A6D2 }
A6CC- A9 0E LDA #0E } "PROGRAM TOO LARGE"
A6CE- D0 02 BNE $A6D2 }
A6D0- A9 0D LDA #0D }
A6D2- 8D 5C AA STA $AA5C } "FILE TYPE MISMATCH"

```

```

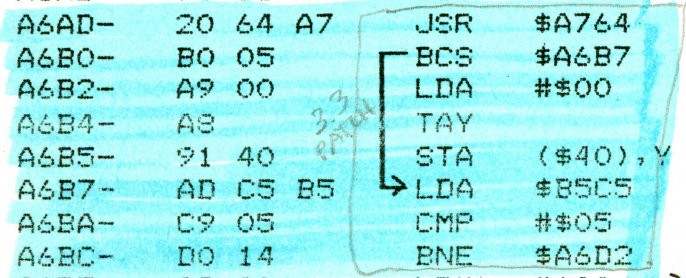
A6D5- 20 E6 BF JSR $BFE6 WARMSTART / CLEAR STATUS
A6D8- AD B6 AA LDA $AAB6 } INT BASIC?
A6DB- F0 04 BEQ $A6E1 } FP ON ERROR UNIT ACTIVE?
A6DD- A5 D8 LDA #D8 } YES
A6DF- 30 0E BMI $A6EF }
A6E1- A2 00 LDX #00 } (CR) (BELL) (CR)
A6E3- 20 02 A7 JSR $A702 }
A6E6- AE 5C AA LDX $AA5C } PRINT PARTICULAR MSG TEXT
A6E9- 20 02 A7 JSR $A702 } (CR)
A6EC- 20 C8 9F JSR $9FC8 }
A6EF- 20 51 A8 JSR $A851 SET DOS INTERCEPTS
A6F2- 20 5E A6 JSR $A65E BASIC RUNNING?
A6F5- AE 5C AA LDX $AA5C PASS ERROR CODE TO ON ERROR UNIT
A6F8- A9 03 LDA #03
A6FA- B0 03 BCS $A6FF NOT RUNNING, WARMSTART
A6FC- 6C 5A 9D JMP ($9D5A) BASIC ERROR ENTRY POINT
A6FF- 6C 5E 9D JMP ($9D5E) BASIC WARMSTART

```

```

A702- BD 3F AA LDA $AA3F,X } LOCATION OF TEXT
A705- AA TAX } BASED ON MSG #
A706- 8E 63 AA STX $AA63 }
A709- BD 71 A9 LDA $A971,X } GET A MSG CHARACTER
A70C- 48 PHA } SAVE IT
A70D- 09 80 ORA #80 } MSB ON FOR PRINTING
A70F- 20 C5 9F JSR $9FC5 } CSWL CALL TO PRINT IT
A712- AE 63 AA LDX $AA63 } NEXT CHAR
A715- E8 INX
A716- 68 PLA
A717- 10 ED BPL $A706 } LAST?

```



MISC. ERRORS

ERROR HANDLER

PRINT ERROR MS

```

A719- 60      RTS
A71A- AD 66 AA  LDA  $AA66 } VOL TO PARM2
A71D- 8D BF B5  STA  $B5BF }
A720- AD 68 AA  LDA  $AA68 } Drive to PARM2+1
A723- 8D C0 B5  STA  $B5C0 }
A726- AD 6A AA  LDA  $AA6A } Slot to PARM3
A729- 8D C1 B5  STA  $B5C1 }
A72C- AD 06 9D  LDA  $9D06 } PRIMARY
A72F- 8D C3 B5  STA  $B5C3 } FILENAME↑ TO PARM4
A732- AD 07 9D  LDA  $9D07 }
A735- 8D C4 B5  STA  $B5C4 }
A738- A5 40      LDA  $40      }
A73A- 8D 4F AA  STA  $AA4F } SAVE BUFFER POINTER
A73D- A5 41      LDA  $41      }
A73F- 8D 50 AA  STA  $AA50 }
A742- 60      RTS

```

BUILD FIO PARM LIST

```

A743- A0 1D      LDY  ##1D      30 CHARS
A745- B9 75 AA   LDA  $AA75,Y } COPY FILENAME
A748- 91 40      STA  ($40),Y }
A74A- 88         DEY
A74B- 10 F8     BPL  $A745
A74D- 60      RTS

```

COPY PRIMARY FILENAME TO BU [B40] → BUFFER FILENAME FIELD

```

A74E- A0 1E      LDY  ##1E      COPY FLOW↑
A750- B1 40      LDA  ($40),Y   T/S BUFF↑
A752- 99 A9 B5  STA  $B5A9,Y   DATA BUFF↑
A755- C8         INY
A756- C0 26     CPY  ##26     NEXT FILE↑
A758- D0 F6     BNE  $A750   To $B5C7
A75A- 60      RTS

```

COPY CURRENT BUFF PTRS TO FIO PARMS

```

A75B- A0 00      LDY  ##00
A75D- 8C 51 AA  STY  $AA51   WARMSTART STATUS
A760- 8C 52 AA  STY  $AA52   STATE = 0
A763- 60      RTS

```

RESET STATE TO 0

```

A764- A9 00      LDA  ##00 } ASSUME NO FREE BUFFS
A766- 85 45      STA  $45 }
A768- 20 92 A7   JSR  $A792   GET FIRST BUFFER
A76B- 4C 73 A7   JMP  $A773
A76E- 20 9A A7   JSR  $A79A   GET NEXT BUFFER
A771- F0 1D      BEQ  $A790   END OF CHAIN, FILE NOT OPEN
A773- 20 AA A7   JSR  $A7AA   GET 1ST BYTE
A776- D0 0A      BNE  $A782   OPEN FILE?
A778- A5 40      LDA  $40
A77A- 85 44      STA  $44 } NO, SAVE PTR TO FREE
A77C- A5 41      LDA  $41 } FILE BUFFER
A77E- 85 45      STA  $45 }
A780- D0 EC     BNE  $A76E
A782- A0 1D     LDY  ##1D
A784- B1 40     LDA  ($40),Y } PRIMARY FILENAME
A786- D9 75 AA  CMP  $AA75,Y } MATCHES THIS BUFFER?
A789- D0 E3     BNE  $A76E }
A78B- 88         DEY
A78C- 10 F6     BPL  $A784
A78E- 18         CLC
A78F- 60      RTS

```

LOCATE A FILE BUFF.

FILE LOCATED

```

A790- 38         SEC
A791- 60      RTS

```

FILE NOT OPEN (44,45 ↑ FREE BUFFER MAYBE)

```

A792- AD 00 9D  LDA  $9D00 } FIND FIRST FILE BUFFER
A795- AE 01 9D  LDX  $9D01 }
A798- D0 0A     BNE  $A7A4   USE IT FIRST
A79A- A0 25     LDY  ##25
A79C- B1 40     LDA  ($40),Y } GET LINK TO NEXT BUFFER
A79E- F0 09     BEQ  $A7A9   END OF CHAIN?
A7A0- AA         TAX

```

RUN DOWN FILE BUFF. CHAIN



A7A1- 88  
 A7A2- B1 40  
 A7A4- 86 41  
 A7A6- 85 40  
 A7A8- 8A  
 A7A9- 60  
 A7AA- A0 00  
 A7AC- B1 40  
 A7AE- 60  
 A7AF- AD B3 AA  
 A7B2- F0 0E  
 A7B4- AD B4 AA  
 A7B7- C5 40  
 A7B9- D0 08  
 A7BB- AD B5 AA  
 A7BE- C5 41  
 A7C0- F0 01  
 A7C2- CA  
 A7C3- 60  
 A7C4- 4D C2 B5  
 A7C7- F0 0A  
 A7C9- 29 7F  
 A7CB- F0 06  
 A7CD- 20 EA A2  
 A7D0- 4C D0 A6  
 A7D3- 60  
 A7D4- 38  
 A7D5- AD 00 9D  
 A7D8- 85 40  
 A7DA- AD 01 9D  
 A7DD- 85 41  
 A7DF- AD 57 AA  
 A7E2- 8D 63 AA  
 A7E5- A0 00  
 A7E7- 98  
 A7E8- 91 40  
 A7EA- A0 1E  
 A7EC- 38  
 A7ED- A5 40  
 A7EF- E9 2D  
 A7F1- 91 40  
 A7F3- 48  
 A7F4- A5 41  
 A7F6- E9 00  
 A7F8- C8  
 A7F9- 91 40  
 A7FB- AA  
 A7FC- CA  
 A7FD- 68  
 A7FE- 48  
 A7FF- C8  
 A800- 91 40  
 A802- 8A  
 A803- C8  
 A804- 91 40  
 A806- AA  
 A807- CA  
 A808- 68  
 A809- 48  
 A80A- C8  
 A80B- 91 40  
 A80D- C8  
 A80E- 8A

DEY  
 LDA (\$40),Y  
 → STX \$41 } SET BUFFER POINTER  
 STA \$40 }  
 TXA  
 INDICATE OPEN FILE FOUND  
 → RTS

GET FIRST BYTE OF FILENAME

LDY #\$00  
 LDA (\$40),Y } GET 1ST BYTE  
 RTS

TEST CURRENT BUFFER FOR EXEC

LDA \$AAB3 } EXEC ACTIVE?  
 BEQ \$A7C2 }  
 LDA \$AAB4 }  
 CMP \$40 } IF SO, IS THE CURRENT  
 BNE \$A7C3 } BUFFER EXEC'S?  
 LDA \$AAB5 }  
 CMP \$41 }  
 BEQ \$A7C3 }  
 DEX  
 RTS

EOR \$B5C2 FILE TYPE MATCH? CHECK FILE TYPE  
 BEQ \$A7D3 YES  
 AND #\$7F NO, LOCKED BIT OFF  
 ← BEQ \$A7D3 TRY ONCE MORE  
 JSR \$A2EA GO CLOSE IT  
 JMP \$A6D0 "FILE TYPE MISMATCH"  
 → RTS

INITIALIZE FILE BUFFERS

LDA \$9D00 }  
 STA \$40 } [40] → FIRST  
 LDA \$9D01 } BUFFER  
 STA \$41 } LINK

SET COUNTER TO MAXFILES

→ LDY #\$00 } MARK FILE CLOSED (FILENAME = #∅)  
 TYA }  
 STA (\$40),Y }  
 LDY #\$1E } INDEX PAST FILENAME FIELD TO LINK PTRS

LDA \$40 }  
 SBC #\$2D }  
 STA (\$40),Y } SET PTR TO FIO WORKAREA  
 PHA } (45 BYTES BEFORE FILENAME)  
 LDA \$41 }  
 SBC #\$00 }  
 STA (\$40),Y }

BACK UP 1 PAGE (256)

PLA }  
 PHA }  
 INY }  
 STA (\$40),Y } SET PTR TO TRACK/SECTOR LIST BUFF.  
 TXA } (256 BYTES BEFORE FIOWA)  
 INY }  
 STA (\$40),Y }

BACK UP 1 MORE PAGE

PLA }  
 PHA }  
 INY }  
 STA (\$40),Y } SET PTR TO DATA SECTOR BUFFER  
 INY } (256 BYTES BEFORE T/S LIST)  
 TXA }

```

A80F- 91 40 STA ($40),Y
A811- CE 63 AA DEC $AA63
A814- F0 17 BEQ $A82D
A816- AA TAX
A817- 68 PLA
A818- 38 SEC
A819- E9 26 SBC ##26
A81B- C8 INY
A81C- 91 40 STA ($40),Y
A81E- 48 PHA
A81F- 8A TXA
A820- E9 00 SBC #$00
A822- C8 INY
A823- 91 40 STA ($40),Y
A825- 85 41 STA $41
A827- 68 PLA
A828- 85 40 STA $40
A82A- 4C E5 A7 JMP $A7E5
A82D- 48 PHA
A82E- A9 00 LDA #$00
A830- C8 INY
A831- 91 40 STA ($40),Y
A833- C8 INY
A834- 91 40 STA ($40),Y
A836- AD B6 AA LDA $AAB6
A839- F0 0B BEQ $A846
A83B- 68 PLA
A83C- 85 74 STA $74
A83E- 85 70 STA $70
A840- 68 PLA
A841- 85 73 STA $73
A843- 85 6F STA $6F
A845- 60 RTS
A846- 68 PLA
A847- 85 4D STA $4D
A849- 85 CB STA $CB
A84B- 68 PLA
A84C- 85 4C STA $4C
A84E- 85 CA STA $CA
A850- 60 RTS
A851- A5 39 LDA $39
A853- CD 03 9D CMP $9D03
A856- F0 12 BEQ $A86A
A858- 8D 56 AA STA $AA56
A85B- A5 38 LDA $38
A85D- 8D 55 AA STA $AA55
A860- AD 02 9D LDA $9D02
A863- 85 38 STA $38
A865- AD 03 9D LDA $9D03
A868- 85 39 STA $39
A86A- A5 37 LDA $37
A86C- CD 05 9D CMP $9D05
A86F- F0 12 BEQ $A883
A871- 8D 54 AA STA $AA54
A874- A5 36 LDA $36
A876- 8D 53 AA STA $AA53
A879- AD 04 9D LDA $9D04
A87C- 85 36 STA $36
A87E- AD 05 9D LDA $9D05
A881- 85 37 STA $37
A883- 60 RTS
A884- 49 4E EOR #$4E
A886- 49 D4 EOR #$D4

```

DECREMENT FILES COUNTER DONE?

SET LINK FROM THIS FILE BUFF TO NEXT (POINTS TO FILENAME FIELD)

POSITION TO NEW BUFFER AND GO SET IT UP

LINK LAST BUFFER TO Ø

INT BASIC? YES...

SET FP'S HIMEM & STRING START PTRS TO JUST BELOW LAST BUFFER

SET INT'S HIMEM & PGM PTR TO JUST BELOW LAST BUFFER

SET DOS KBD/VID I/O INTERCEPTS

SAVE KBD HANDLER EPA

INSERT DOS KBD INTERCEPT

CSWL STILL TDOS?

YES - EXIT

SAVE VID HANDLER EPA

INSERT DOS VID INTERCEPT

COMMAND TEXT TABLE

A888-	4C 4F 41	JMP	\$414F
A88B-	C4 53	CPY	\$53
A88D-	41 56	EOR	(\$56, X)
A88F-	C5 52	CMP	\$52
A891-	55 CE	EOR	\$CE, X
A893-	43	???	
A894-	48	PHA	
A895-	41 49	EOR	(\$49, X)
A897-	CE 44 45	DEC	\$4544
A89A-	4C 45 54	JMP	\$5445
A89D-	C5 4C	CMP	\$4C
A89F-	4F	???	
A8A0-	43	???	
A8A1-	CB	???	
A8A2-	55 4E	EOR	\$4E, X
A8A4-	4C 4F 43	JMP	\$434F
A8A7-	CB	???	
A8A8-	43	???	
A8A9-	4C 4F 53	JMP	\$534F
A8AC-	C5 52	CMP	\$52
A8AE-	45 41	EOR	\$41
A8B0-	C4 45	CPY	\$45
A8B2-	58	CLI	
A8B3-	45 C3	EOR	\$C3
A8B5-	57	???	
A8B6-	52	???	
A8B7-	49 54	EOR	##54
A8B9-	C5 50	CMP	\$50
A8BB-	4F	???	
A8BC-	53	???	
A8BD-	49 54	EOR	##54
A8BF-	49 4F	EOR	##4F
A8C1-	CE 4F 50	DEC	\$504F
A8C4-	45 CE	EOR	\$CE
A8C6-	41 50	EOR	(\$50, X)
A8C8-	50 45	BVC	\$A90F
A8CA-	4E C4 52	LSR	\$52C4
A8CD-	45 4E	EOR	\$4E
A8CF-	41 4D	EOR	(\$4D, X)
A8D1-	C5 43	CMP	\$43
A8D3-	41 54	EOR	(\$54, X)
A8D5-	41 4C	EOR	(\$4C, X)
A8D7-	4F	???	
A8D8-	C7	???	
A8D9-	4D 4F CE	EOR	\$CE4F
A8DC-	4E 4F 4D	LSR	\$4D4F
A8DF-	4F	???	
A8E0-	CE 50 52	DEC	\$5250
A8E3-	A3	???	
A8E4-	49 4E	EOR	##4E
A8E6-	A3	???	
A8E7-	4D 41 58	EOR	\$5841
A8EA-	46 49	LSR	\$49
A8EC-	4C 45 D3	JMP	\$D345
A8EF-	46 D0	LSR	\$D0
A8F1-	49 4E	EOR	##4E
A8F3-	D4	???	
A8F4-	42	???	
A8F5-	53	???	
A8F6-	41 56	EOR	(\$56, X)
A8F8-	C5 42	CMP	\$42
A8FA-	4C 4F 41	JMP	\$414F
A8FD-	C4 42	CPY	\$42

MSB ON  
 49 4E 49 D4  
 I N I T  
 etc.

1  
 COMMAND  
 TEXT  
 TABLE  
 ↓

A8D2

A8FF-	52	???	
A900-	55 CE	EOR	\$CE, X
A902-	56 45	LSR	\$45, X
A904-	52	???	
A905-	49 46	EOR	##46
A907-	D9 00	21 CMP	\$2100, Y
A90A-	70 A0	BVS	\$A8AC
A90C-	70 A1	BVS	\$A8AF
A90E-	70 A0	BVS	\$A8B0
A910-	70 20	BVS	\$A932
A912-	70 20	BVS	\$A934
A914-	70 20	BVS	\$A936
A916-	70 20	BVS	\$A938
A918-	70 60	BVS	\$A97A
A91A-	00	BRK	
A91B-	22	???	
A91C-	06 20	ASL	\$20
A91E-	74	???	
A91F-	22	???	
A920-	06 22	ASL	\$22
A922-	04	???	
A923-	23	???	
A924-	78	SEI	
A925-	22	???	
A926-	70 30	BVS	\$A958
A928-	70 40	BVS	\$A96A
A92A-	70 40	BVS	\$A96C
A92C-	80	???	
A92D-	40	RTI	
A92E-	80	???	
A92F-	08	PHP	
A930-	00	BRK	
A931-	08	PHP	
A932-	00	BRK	
A933-	04	???	
A934-	00	BRK	
A935-	40	RTI	
A936-	70 40	BVS	\$A978
A938-	00	BRK	
A939-	21 79	AND	(\$79, X)
A93B-	20 71 20	JSR	\$2071
A93E-	71 20	ADC	(\$20), Y
A940-	70 D6	BVS	\$A918
A942-	04 D3	CPY	\$D3
A944-	0C D2 C2	CPY	\$C2D2
A947-	01 C3	CMP	(\$C3, X)
A949-	09 CF	CMP	##CF
A94B-	40	RTI	
A94C-	20 10 08	JSR	\$0810
A94F-	04	???	
A950-	02	???	
A951-	01 C0	ORA	(\$C0, X)
A953-	A0 90	LDY	##90
A955-	00	BRK	
A956-	00	BRK	
A957-	FE 00 01	INC	\$0100, X
A95A-	00	BRK	
A95B-	02	???	
A95C-	00	BRK	
A95D-	01 00	ORA	(\$00, X)
A95F-	07	???	
A960-	00	BRK	
A961-	01 00	ORA	(\$00, X)

COMMAND VALID PARMS TABLE

0	INIT	2170
2	LOAD	A070
4	SAVE	A170
6	RUN	A070
8	CHAIN	2070
10	DELETE	2070
12	LOCK	2070
14	UNLOCK	2070
16	CLOSE	6000
18	READ	2206
20	EXEC	2074
22	WRITE	2206
24	POSITION	2204
26	OPEN	2378
28	APPEND	2270
30	RENAME	3070
32	CATALOG	4070
34	MON	4080
36	NOMON	4080
38	PR#	0800
40	IN#	0800
42	MAXFILES	0400
44	FP	4070
46	INT	4000
48	BSAVE	2179
50	BLOAD	2071
52	BRUN	2071
54	VERIFY	2070

BIT	OPERAND
0	FILENAME OPT
1	NO POSITIONAL O
2	FILENAME #1
3	FILENAME #2
4	slot #
5	Maxfiles #
*6	deferred cmd
*7	create new file
8	C, I, O
9	V - Volume #
10	D - Drive
11	S - Slot
12	L - Length
13	R - Record #
14	B - Byte
15	A - Address

"V, D, S, L, R, B, A,"  
"C, I, O"

KEYWORD NAME TABLE

V-40	B-02
D-20	A-01
S-10	C-C0
L-08	I-A0
R-04	O-90

NO NUMERIC OPERAND

KEYWORD BIT POSITION TABLE

	MIN	MAX
V	0	254
D	1	2
S	1	7
L	1	32767
R	0	32767

KEYWORD VALUE RANGE TABLE

A963-	FF	???
A964-	7F	???
A965-	00	BRK
A966-	00	BRK
A967-	FF	???
A968-	7F	???
A969-	00	BRK
A96A-	00	BRK
A96B-	FF	???
A96C-	7F	???
A96D-	00	BRK
A96E-	00	BRK
A96F-	FF	???
A970-	FF	???

B | Ø | 32767  
 A | Ø | 65535

C, I, O not in this table  
 Since they do not have  
 numeric values.

**MESSAGE TEXT TABLE**

A971-	0D 07 8D	ORA	\$8D07
A974-	4C 41 4E	JMP	\$4E41
A977-	47	???	
A978-	55 41	EOR	\$41, X
A97A-	47	???	
A97B-	45 20	EOR	\$20
A97D-	4E 4F 54	LSR	\$544F
A980-	20 41 56	JSR	\$5641
A983-	41 49	EOR	(\$49, X)
A985-	4C 41 42	JMP	\$4241
A988-	4C C5 52	JMP	\$52C5
A98B-	41 4E	EOR	(\$4E, X)
A98D-	47	???	
A98E-	45 20	EOR	\$20
A990-	45 52	EOR	\$52
A992-	52	???	
A993-	4F	???	
A994-	D2	???	
A995-	57	???	
A996-	52	???	
A997-	49 54	EOR	##54
A999-	45 20	EOR	\$20
A99B-	50 52	BVC	\$A9EF
A99D-	4F	???	
A99E-	54	???	
A99F-	45 43	EOR	\$43
A9A1-	54	???	
A9A2-	45 C4	EOR	\$C4
A9A4-	45 4E	EOR	\$4E
A9A6-	44	???	
A9A7-	20 4F 46	JSR	\$464F
A9AA-	20 44 41	JSR	\$4144
A9AD-	54	???	
A9AE-	C1 46	CMP	(\$46, X)
A9B0-	49 4C	EOR	##4C
A9B2-	45 20	EOR	\$20
A9B4-	4E 4F 54	LSR	\$544F
A9B7-	20 46 4F	JSR	\$4F46
A9BA-	55 4E	EOR	\$4E, X
A9BC-	C4 56	CPY	\$56
A9BE-	4F	???	
A9BF-	4C 55 4D	JMP	\$4D55
A9C2-	45 20	EOR	\$20
A9C4-	4D 49 53	EOR	\$5349
A9C7-	4D 41 54	EOR	\$5441
A9CA-	43	???	
A9CB-	C8	INY	
A9CC-	49 2F	EOR	##2F
A9CE-	4F	???	

MSG #  
 Ø  
 1  
 2 \*  
 3 \*  
 4  
 5  
 6  
 7  
 8  
 9  
 10  
 11  
 12  
 13  
 14  
 15

TEXT  
 " (CR) (BEL) (CR) "  
 " LANGUAGE NOT AVAILABLE "  
 " RANGE ERROR "  
 " RANGE ERROR "  
 " WRITE PROTECTED "  
 " END OF DATA "  
 " FILE NOT FOUND "  
 " VOLUME MISMATCH "  
 " I/O ERROR "  
 " DISK FULL "  
 " FILE LOCKED "  
 " SYNTAX ERROR "  
 " NO BUFFERS AVAILABLE "  
 " FILE TYPE MISMATCH "  
 " PROGRAM TOO LARGE "  
 " NOT DIRECT COMMAND "

\* 2 - BAD FIO OPCODE  
 \* 3 - BAD FIO SUBCODE

A9CF-	20 45 52	JSR	\$5245
A9D2-	52	???	
A9D3-	4F	???	
A9D4-	D2	???	
A9D5-	44	???	
A9D6-	49 53	EOR	##53
A9D8-	4B	???	
A9D9-	20 46 55	JSR	\$5546
A9DC-	4C CC 46	JMP	\$46CC
A9DF-	49 4C	EOR	##4C
A9E1-	45 20	EOR	\$20
A9E3-	4C 4F 43	JMP	\$434F
A9E6-	4B	???	
A9E7-	45 C4	EOR	\$C4
A9E9-	53	???	
A9EA-	59 4E 54	EOR	\$544E, Y
A9ED-	41 58	EOR	(#58, X)
A9EF-	20 45 52	JSR	\$5245
A9F2-	52	???	
A9F3-	4F	???	
A9F4-	D2	???	
A9F5-	4E 4F 20	LSR	\$204F
A9F8-	42	???	
A9F9-	55 46	EOR	\$46, X
A9FB-	46 45	LSR	\$45
A9FD-	52	???	
A9FE-	53	???	
A9FF-	20 41 56	JSR	\$5641
AA02-	41 49	EOR	(#49, X)
AA04-	4C 41 42	JMP	\$4241
AA07-	4C C5 46	JMP	\$46C5
AA0A-	49 4C	EOR	##4C
AA0C-	45 20	EOR	\$20
AA0E-	54	???	
AA0F-	59 50 45	EOR	\$4550, Y
AA12-	20 4D 49	JSR	\$494D
AA15-	53	???	
AA16-	4D 41 54	EOR	\$5441
AA19-	43	???	
AA1A-	C8	INY	
AA1B-	50 52	BVC	\$AA6F
AA1D-	4F	???	
AA1E-	47	???	
AA1F-	52	???	
AA20-	41 4D	EOR	(#4D, X)
AA22-	20 54 4F	JSR	\$4F54
AA25-	4F	???	
AA26-	20 4C 41	JSR	\$414C
AA29-	52	???	
AA2A-	47	???	
AA2B-	C5 4E	CMP	\$4E
AA2D-	4F	???	
AA2E-	54	???	
AA2F-	20 44 49	JSR	\$4944
AA32-	52	???	
AA33-	45 43	EOR	\$43
AA35-	54	???	
AA36-	20 43 4F	JSR	\$4F43
AA39-	4D 4D 41	EOR	\$414D
AA3C-	4E C4 8D	LSR	\$8DC4
AA3F-	00	BRK	
AA40-	03	???	
AA41-	19 19 24	ORA	\$2419, Y

|  
MESSAGE  
TEXT  
TABLE  
↓

MSG #	TEXT INDEX
	TABLE



AA44-	33	???	
AA45-	3E 4C 5B	ROL	#5B4C, X
AA48-	64	???	
AA49-	6D 78 84	ADC	#8478
AA4C-	98	TYA	
AA4D-	AA	TAX	
AA4E-	BB	???	
AA4F-	2D 98 00	AND	#0098
AA52-	03	???	
AA53-	F0 FD	BEQ	#AA52
AA55-	1B	???	
AA56-	FD 03 03	SBC	#0303, X
AA59-	F7	???	
AA5A-	00	BRK	
AA5B-	01 AD	ORA	(#A0, X)
AA5D-	04	???	
AA5E-	00	BRK	
AA5F-	1A	???	
AA60-	9F	???	
AA61-	00	BRK	
AA62-	00	BRK	
AA63-	CC 00 00	CPY	#0000
AA66-	00 V	BRK	
AA67-	00	BRK	
AA68-	01 00 D	ORA	(#00, X)
AA6A-	07 S	???	
AA6B-	00	BRK	
AA6C-	01 00 L	ORA	(#00, X)
AA6E-	00 R	BRK	
AA6F-	00	BRK	
AA70-	00	BRK	
AA71-	00 B	BRK	
AA72-	00 A	BRK	
AA73-	00	BRK	
AA74-	00 MON	BRK	

AA4F: CURRENT BUFFER ADDRESS  
AA51: STATUS FLAGS - { 01: READ STATE  
00: WARMSTART  
80: COLDSTART  
40: FP RAM  
AA52: CSWL STATE  
AA53: TRUE VID HANDLER EP  
AA55: TRUE KBD HANDLER EP  
AA57: MAXFILES  
AA59: { S, X, Y, A REGISTER SAVE DURING  
to AA5C: { KBD/VID INTERCEPT  
AA5D: LINE INDEX  
AA5E: "MON" FLAGS (C, I, O: 40, 20, 10)  
AA5F: LAST COMMAND INDEX #2  
AA60: RANGE LENGTH FOR LOADS  
AA62: PENDING CMD #  
AA64: KEYWORD INDEX #  
AA63: BUFFER INIT COUNTER / etc. MSG INDE  
AA65: BITS FOR KEYWORDS PARSED

- VOLUME #
  - DRIVE #
  - SLOT #
  - LENGTH
  - RECORD #
  - BYTE
  - ADDRESS
- MON C, I and/or O (40=C, 20=I, 10=O)

KEYWORD VALUES

AA75-	C8	INY	
AA76-	C5 CC	CMP	#CC
AA78-	CC CF A0	CPY	#A0CF
AA7B-	A0 A0	LDY	##A0
AA7D-	A0 A0	LDY	##A0
AA7F-	A0 A0	LDY	##A0
AA81-	A0 A0	LDY	##A0
AA83-	A0 A0	LDY	##A0
AA85-	A0 A0	LDY	##A0
AA87-	A0 A0	LDY	##A0
AA89-	A0 A0	LDY	##A0
AA8B-	A0 A0	LDY	##A0
AA8D-	A0 A0	LDY	##A0
AA8F-	A0 A0	LDY	##A0
AA91-	A0 A0	LDY	##A0
AA93-	A0 A0	LDY	##A0
AA95-	A0 A0	LDY	##A0
AA97-	A0 A0	LDY	##A0
AA99-	A0 A0	LDY	##A0
AA9B-	A0 A0	LDY	##A0
AA9D-	A0 A0	LDY	##A0
AA9F-	A0 A0	LDY	##A0
AAA1-	A0 A0	LDY	##A0
AAA3-	A0 A0	LDY	##A0
AAA5-	A0 A0	LDY	##A0
AAA7-	A0 A0	LDY	##A0
AAA9-	A0 A0	LDY	##A0
AAAB-	A0 A0	LDY	##A0

← 9D06

PRIMARY FILENAME  
(INITIALY CONTAINS GREETING  
FILENAME)

SECONDARY (RENAME) FILENAME

AAD-	A0 A0	LDY	##A0
AAF-	A0 A0	LDY	##A0
AAB1-	03	???	
AAB2-	B4 00	STY	\$00
AAB4-	00	BRK	
AAB5-	00	BRK	
AAB6-	00	BRK	
AAB7-	00	BRK	
AAB8-	C1 D0	CMP	(#D0, X)
AABA-	D0 CC	BNE	##A88
AABC-	C5 D3	CMP	#D3
AABE-	CF	???	
AABF-	C6 D4	DEC	#D4
AAC1-	E8	INX	
AAC2-	B7	???	
AAC3-	BB	???	
AAC4-	B3	???	
AAC5-	BB	???	
AAC6-	B4 00	LDY	\$00, X
AAC8-	D0 7E	CPY	##7E
AACA-	B3	???	
AACB-	21 AB	AND	(#AB, X)
AACD-	05 AC	ORA	#AC
AACF-	57	???	
AAD0-	AC 6F AC	LDY	#AC6F
AAD3-	2A	ROL	
AAD4-	AD 97 AD	LDA	#AD97
AAD7-	EE AC F5	INC	#F5AC
AADA-	AC 39 AC	LDY	#AC39
AADD-	11 AD	ORA	(#AD), Y
AADF-	8D AE 17	STA	#17AE
AAE2-	AD 7E B3	LDA	#B37E
AAE5-	7E B3 89	ROR	##B3, X
AAE8-	AC 95 AC	LDY	#AC95
AAEB-	86 AC	STX	#AC
AAED-	92	???	
AAEE-	AC 7E B3	LDY	#B37E
AAF1-	7E B3 BD	ROR	##BD, X
AAF4-	AC C9 AC	LDY	#ACC9
AAF7-	BA	TSX	
AAF8-	AC C6 AC	LDY	#ACC6
* AAFB-	7E B3 E0		
* AAFE-	00	CPX	##00 X REG = 0?
* AAFF-	F0 02	BEQ	##A03 IF SO, ALLOW NEW FILES
* AB01-	A2 02	LDX	##02 IF NOT, DISALLOW
* AB03-	8E 5F AA	STX	##A5F BY SETTING LAST CMD INDEX ACCORDINGLY
AB06-	BA	TSX	
AB07-	8E 9B B3	STX	##B39B } SAVE STACK PTR
AB0A-	20 6A AE	JSR	##AE6A RESTORE FIOWA FROM BUFF
AB0D-	AD BB B5	LDA	##B5BB GET DESIRED OPERATION
AB10-	C9 0D	CMP	##0D } <13? NO RC=2
AB12-	B0 0B	BCS	##AB1F
AB14-	0A	ASL	
AB15-	AA	TAX	
AB16-	BD CA AA	LDA	##ACA, X } FIND OPERATION HANDLER ROUTINE
AB19-	48	PHA	
AB1A-	BD C9 AA	LDA	##AC9, X
AB1D-	48	PHA	
AB1E-	60	RTS	GO TO IT
AB1F-	4C 63 B3	JMP	##B363 RC=2 / BAD OPCODE
AB22-	20 28 AB	JSR	##AB28 OPEN FILE
AB25-	4C 7F B3	JMP	##B37F EXIT
AB28-	20 DC AB	JSR	##ABDC INITIALIZE FIOWA

MAXFILES DEFAULT  
 "CTL-D" / EXEC FILE ACTIVE FLAG (70)  
 } ↑ EXEC BUFF  
 BASIC ACTIVE { 00 - INT  
 40 - FP ROM  
 RUN FLAG { 80 - FP RAM

"APPLESOFT"

} ADDRESS OF RWTS PARMLIST  
 } VTOC SECTOR BUFFER ↑  
 } DIRECTORY SECTOR BUFFER ↑  
 } END OF DOS ↑

FIO FUNCTION ROUTINE TABLE

- 0 - B37F NOP
- 1 - AB22 OPEN
- 2 - AC06 CLOSE
- 3 - AC58 READ
- 4 - AC70 WRITE
- 5 - AD20 DELETE
- 6 - AD98 CATALOG
- 7 - ACEF LOCK
- 8 - ACFG UNLOCK
- 9 - AC3A RENAME
- 10 - AD12 POSITION
- 11 - AE8E INIT
- 12 - AD18 VERIFY
- 13 - B37F NOP

READ SUBCODE HANDLER TAB

- 0 - B37F NOP
- 1 - AC8A READ 1 BYTE
- 2 - AC96 READ RANGE
- 3 - AC87 POSITION/READ 1
- 4 - AC93 POSITION/READ RANGE
- 5 - B37F NOP

WRITE SUBCODE HANDLER TAB

- 0 - B37F NOP
- 1 - ACBE WRITE 1 BYTE
- 2 - ACCA WRITE RANGE
- 3 - ACBB POSITION/WRITE 1
- 4 - ACC7 POSITION/WRITE RANGE
- 5 - B37F NOP

FIO EXTERNAL ENTRY

FIO DOS FILE MANAGER

} SAVE STACK PTR  
 RESTORE FIOWA FROM BUFF  
 GET DESIRED OPERATION  
 } <13? NO RC=2  
 } FIND OPERATION HANDLER ROUTINE

GO TO IT

OPEN

AB2B-	A9 01	LDA	##01	} SET SECTOR LENGTH TO 256
AB2D-	8D E3 B5	STA	#B5E3	
AB30-	AE BE B5	LDX	#B5BE	} GET RECORD LENGTH
AB33-	AD BD B5	LDA	#B5BD	
AB36-	DO 05	BNE	#AB3D	} IS IT NON-ZERO?
AB38-	EO 00	CPX	##00	
AB3A-	DO 01	BNE	#AB3D	
AB3C-	E8	INX		IF NOT, FORCE IT TO 1
AB3D-	8D E8 B5	STA	#B5E8	} SET IT IN FIOWA
AB40-	8E E9 B5	STX	#B5E9	
AB43-	20 C9 B1	JSR	#B1C9	LOCATE/ALLOCATE DIRECTORY ENTRY
AB46-	90 5E	BCC	#ABA6	FILE ALREADY EXISTS?
AB48-	8E 9C B3	STX	#B39C	SAVE DIRECTORY INDEX
* AB4B-	AE 5F AA	LDX	#AA5F	GET LAST CMD ENTERED INDEX
* AB4E-	BD 09 A9	LDA	#A909, X	VALID KEYWORDS
* AB51-	AE 9C B3	LDX	#B39C	RESTORE DIRECTORY INDEX
* AB54-	4A	LSR		} INIT, SAVE, OPEN, or BSAVE /IF SO, OK
* AB55-	B0 0D	BCS	#AB64	
* AB57-	AD 51 AA	LDA	#AA51	} RUNNING "APPLESOFT"?
* AB5A-	C9 C0	CMP	##C0	
* AB5C-	DO 03	BNE	#AB61	} YES, "LANGUAGE NOT AVAILABLE"
* AB5E-	4C 5F B3	JMP	#B35F	
* AB61-	4C 73 B3	JMP	#B373	NO, "FILE NOT FOUND"
AB64-	A9 00	LDA	##00	} SET SECTOR COUNT TO ONE (T/S LIST SECTOR ONLY)
AB66-	9D E8 B4	STA	#B4E8, X	
AB69-	A9 01	LDA	##01	} SAVE DIRECTORY INDEX
AB6B-	9D E7 B4	STA	#B4E7, X	
AB6E-	8E 9C B3	STX	#B39C	SAVE DIRECTORY INDEX
AB71-	20 44 B2	JSR	#B244	ALLOCATE A SECTOR FOR T/S LIST
AB74-	AE 9C B3	LDX	#B39C	RESTORE INDEX
AB77-	9D C7 B4	STA	#B4C7, X	SET SECTOR NO. IN DIRECTORY ENTRY
AB7A-	8D D2 B5	STA	#B5D2	} AND IN FIOWA (1ST & CURR)
AB7D-	8D D4 B5	STA	#B5D4	
AB80-	AD F1 B5	LDA	#B5F1	} TRACK TO DIRECTORY
AB83-	9D C6 B4	STA	#B4C6, X	
AB86-	8D D1 B5	STA	#B5D1	} AND FIOWA
AB89-	8D D3 B5	STA	#B5D3	
AB8C-	AD C2 B5	LDA	#B5C2	} FILETYPE TO DIRECTORY ENTRY
AB8F-	9D C8 B4	STA	#B4C8, X	
AB92-	20 37 B0	JSR	#B037	WRITE BACK DIRECTORY SECTOR
AB95-	20 0C AF	JSR	#AF0C	POINT TO TRACK/SECTOR LIST BUFFER
AB98-	20 D6 B7	JSR	#B7D6	ZERO IT
AB9B-	20 3A AF	JSR	#AF3A	WRITE IT OUT
AB9E-	AE 9C B3	LDX	#B39C	DIRECTORY INDEX AGAIN
ABA1-	A9 06	LDA	##06	} "FILE NOT FOUND" (WAS CREATED) RC=6
ABA3-	8D C5 B5	STA	#B5C5	
ABA6-	BD C6 B4	LDA	#B4C6, X	} T/S OF TRACK/SECTOR LIST TO 1ST T/S LIST ↑ IN FIOWA
ABA9-	8D D1 B5	STA	#B5D1	
ABAC-	BD C7 B4	LDA	#B4C7, X	} PASS BACK FILETYPE IN PARM LIST AND IN FIOWA
ABAF-	8D D2 B5	STA	#B5D2	
ABB2-	BD C8 B4	LDA	#B4C8, X	} COPY NO. OF SECTORS IN FILE FROM DIRECTORY TO FIOWA
ABB5-	8D C2 B5	STA	#B5C2	
ABB8-	8D F6 B5	STA	#B5F6	} DIRECTORY OFFSET TO FIOWA
ABBB-	BD E7 B4	LDA	#B4E7, X	
ABBE-	8D EE B5	STA	#B5EE	} END OF DATA PTR TO INFINITY (ALMOST!)
ABC1-	BD E8 B4	LDA	#B4E8, X	
ABC4-	8D EF B5	STA	#B5EF	} NUMBER OF DATA BYTES REPRESENTED & ONE T/S LIST SECTOR (122*256) 30.5K
ABC7-	8E D9 B5	STX	#B5D9	
ABCA-	A9 FF	LDA	##FF	} END OF DATA PTR TO INFINITY (ALMOST!)
ABCC-	8D E0 B5	STA	#B5E0	
ABCF-	8D E1 B5	STA	#B5E1	} NUMBER OF DATA BYTES REPRESENTED & ONE T/S LIST SECTOR (122*256) 30.5K
ABD2-	AD E2 B3	LDA	#B3E2	
ABD5-	8D DA B5	STA	#B5DA	

NEW FILE

OLD FILE

ABD8-	18		CLC		} INITIALIZE TO FIRST T/S LIST		
ABD9-	4C	5E	AF	JMP	AF5E	} IN FILE	
ABDC-	A9	00		LDA	#00	} ZERO 45 BYTE WORKAREA	
ABDE-	AA			TAX			
ABDF-	9D	D1	B5	STA	B5D1, X		
ABE2-	E8			INX			
ABE3-	E0	2D		CPX	#2D		
ABE5-	D0	F8		BNE	ABDF	GET VOLUME #	
ABE7-	AD	BF	B5	LDA	B5BF	COMPLIMENT	
ABEA-	49	FF		EOR	#FF	SAVE FOR COMPARISONS	
ABEC-	8D	F9	B5	STA	B5F9	} DRIVE	
ABEF-	AD	C0	B5	LDA	B5C0		
ABF2-	8D	F8	B5	STA	B5F8	} SLOT*16	
ABF5-	AD	C1	B5	LDA	B5C1		
ABF8-	0A			ASL			
ABF9-	0A			ASL			
ABFA-	0A			ASL			
ABFB-	0A			ASL			
ABFC-	AA			TAX			
ABFD-	8E	F7	B5	STX	B5F7	} TRACK = 17 (CATALOG TRACK)	
AC00-	A9	11		LDA	#11		
AC02-	8D	FA	B5	STA	B5FA		
AC05-	60			RTS			
AC06-	20	1D	AF	JSR	AF1D	} CHECKPOINT DATA BUFFER	} CLOSE
AC09-	20	34	AF	JSR	AF34	} AND T/S LIST BUFFER	
AC0C-	20	C3	B2	JSR	B2C3	} RELEASE PREALLOCATED SECTORS	
AC0F-	A9	02		LDA	#02	} VTDC NEEDS REREADING?	
AC11-	2D	D5	B5	AND	B5D5		} NO
AC14-	F0	21		BEQ	AC37	} YES, GO DO IT	
AC16-	20	F7	AF	JSR	AFF7	} FLUSH THRU DIRECTORY SECTORS TO THE ONE DESCRIBING THIS FILE	
AC19-	A9	00		LDA	#00		
AC1B-	18			CLC			
AC1C-	20	11	B0	JSR	B011		
AC1F-	38			SEC			
AC20-	CE	D8	B5	DEC	B5D8	} GET OFFSET IN DIRECTORY SECTOR	
AC23-	D0	F7		BNE	AC1C		
AC25-	AE	D9	B5	LDX	B5D9	} UPDATE SECTOR COUNT	
AC28-	AD	EE	B5	LDA	B5EE		
AC2B-	9D	E7	B4	STA	B4E7, X		
AC2E-	AD	EF	B5	LDA	B5EF	} WRITE IT BACK TO DISK	
AC31-	9D	E8	B4	STA	B4E8, X		
AC34-	20	37	B0	JSR	B037	} EXIT	
AC37-	4C	7F	B3	JMP	B37F		
AC3A-	20	28	AB	JSR	AB28	} LOCATE/OPEN FILE	} RENAME
AC3D-	AD	F6	B5	LDA	B5F6	} FILE LOCKED?	
AC40-	30	2B		BMI	AC6D	} [\$42] -> NEW NAME	
AC42-	AD	BD	B5	LDA	B5BD		
AC45-	85	42		STA	#42		
AC47-	AD	BE	B5	LDA	B5BE	} DIRECTORY INDEX	
AC4A-	85	43		STA	#43		
AC4C-	AE	9C	B3	LDX	B39C	} COPY NEW NAME TO DIRECTORY	
AC4F-	20	1C	B2	JSR	B21C		
AC52-	20	37	B0	JSR	B037	} WRITE DIRECTORY	
AC55-	4C	7F	B3	JMP	B37F		
AC58-	AD	BC	B5	LDA	B5BC	} SUBCODE ≥ 5? INVALID IF SO	} READ
AC5B-	C9	05		CMP	#05		
AC5D-	B0	0B		BCS	AC6A	} *2 FOR INDEX	
AC5F-	0A			ASL			
AC60-	AA			TAX		} DETERMINE SUB-HANDLER ROUTINE	
AC61-	BD	E6	AA	LDA	AAE6, X		
AC64-	48			PHA			
AC65-	BD	E5	AA	LDA	AAE5, X		
AC68-	48			PHA			



AC69-	60		RTS		GO TO IT	
AC6A-	4C 67 B3		JMP	\$B367	RC=3 SUBCODE BAD	
AC6D-	4C 7B B3		JMP	\$B37B	"FILE LOCKED"	
AC70-	AD F6 B5		LDA	\$B5F6	} LOCKED?	WRITE
AC73-	30 F8		BMI	\$AC6D		
AC75-	AD BC B5		LDA	\$B5BC	} SUBCODE VALID? (NOT Z5)	
AC78-	C9 05		CMP	##05		
AC7A-	B0 EE		BCS	\$AC6A		
AC7C-	0A		ASL		*2 FOR INDEX	
AC7D-	AA		TAX			
AC7E-	BD F2 AA		LDA	\$AAF2,X	} DETERMINE SUB-HANDLER ROUTINE	
AC81-	48		PHA			
AC82-	BD F1 AA		LDA	\$AAF1,X		
AC85-	48		PHA			
AC86-	60		RTS		GO TO IT	
AC87-	20 00 B3		JSR	\$B300	POSITION	READ ONE BYTE
AC8A-	20 A8 AC		JSR	\$ACA8	READ NEXT BYTE	
AC8D-	8D C3 B5		STA	\$B5C3	PASS BACK IN PARM LIST	
AC90-	4C 7F B3		JMP	\$B37F	EXIT	
AC93-	20 00 B3		JSR	\$B300	POSITION	READ A RANGE OF BYTES
AC96-	20 B5 B1		JSR	\$B1B5	DEC & CHECK LENGTH	
AC99-	20 A8 AC		JSR	\$ACA8	READ A BYTE	
AC9C-	48		PHA		SAVE IT	
AC9D-	20 A2 B1		JSR	\$B1A2	POINT TO DATA AREA	
ACA0-	A0 00		LDY	##00	} STORE BYTE READ	
ACA2-	68		PLA			
ACA3-	91 42		STA	(\$42),Y		
ACA5-	4C 96 AC		JMP	\$AC96	CONTINUE	
ACAB-	20 B6 B0		JSR	\$B0B6	READ A BYTE	
ACAB-	B0 0B		BCS	\$ACB8	END OF DATA?	
ACAD-	B1 42		LDA	(\$42),Y	} SAVE IT	
ACAF-	48		PHA			
ACB0-	20 5B B1		JSR	\$B15B	INCREMENT RECORD #/BYTE OFFSET	
ACB3-	20 94 B1		JSR	\$B194	INCREMENT FILE OFFSET	
ACB6-	68		PLA		} RETURN WITH DATA READ	
ACB7-	60		RTS			
ACB8-	4C 6F B3		JMP	\$B36F	"END OF DATA"	
ACBB-	20 00 B3		JSR	\$B300	POSITION	WRITE ONE BYTE
ACBE-	AD C3 B5		LDA	\$B5C3	GET DATA TO WRITE	
ACC1-	20 DA AC		JSR	\$ACDA	GO DO IT	
ACC4-	4C 7F B3		JMP	\$B37F	AND EXIT	
ACC7-	20 00 B3		JSR	\$B300	POSITION	WRITE A RANGE OF BYTES
ACCA-	20 A2 B1		JSR	\$B1A2	COPY AND ADVANCE ADDR	
ACCD-	A0 00		LDY	##00	} GET BYTE TO WRITE	
ACCF-	B1 42		LDA	(\$42),Y		
ACD1-	20 DA AC		JSR	\$ACDA	GO WRITE IT	
ACD4-	20 B5 B1		JSR	\$B1B5	CHECK & DECREMENT LENGTH	
ACD7-	4C CA AC		JMP	\$ACCA	CONTINUE	
ACDA-	48		PHA		SAVE DATA	
ACDB-	20 B6 B0		JSR	\$B0B6	GO READ PROPER SECTOR	
ACDE-	68		PLA		} STORE BYTE TO BE WRITTEN	
ACDF-	91 42		STA	(\$42),Y		
ACE1-	A9 40		LDA	##40	} FLAG DATA BUFF AS NEEDING CHECKPOINT	
ACE3-	0D D5 B5		ORA	\$B5D5		
ACE6-	8D D5 B5		STA	\$B5D5		
ACE9-	20 5B B1		JSR	\$B15B	INCREMENT RECORD #/BYTE OFFSET	
ACEC-	4C 94 B1		JMP	\$B194	EXIT VIA FILE OFFSET INCREMENT	
ACEF-	A9 80		LDA	##80	} SET MASK TO LOCK	LOCK
ACF1-	8D 9E B3		STA	\$B39E		
ACF4-	D0 05		BNE	\$ACFB	COMMON CODE	
ACF6-	A9 00		LDA	##00	} SET MASK TO UNLOCK	UNLOCK
ACF8-	8D 9E B3		STA	\$B39E		
ACFB-	20 28 AB		JSR	\$AB28	OPEN FILE	

ACFE-	AE 9C B3	LDX	\$B39C	GET DIRECTORY INDEX		
AD01-	BD C8 B4	LDA	\$B4C8, X	UPDATE FILE TYPE TO LOCK - 80 OR UNLOCK - 780		
AD04-	29 7F	AND	##7F			
AD06-	0D 9E B3	ORA	\$B39E			
AD09-	9D C8 B4	STA	\$B4C8, X			
AD0C-	20 37 B0	JSR	\$B037	WRITE DIRECTORY SECTOR		
AD0F-	4C 7F B3	JMP	\$B37F	EXIT		
AD12-	20 00 B3	JSR	\$B300	POSITION	POSITION	
AD15-	4C 7F B3	JMP	\$B37F	EXIT		
AD18-	20 28 AB	JSR	\$AB28	OPEN FILE	VERIFY	
AD1B-	20 B6 B0	JSR	\$B0B6	READ NEXT DATA SECTOR		
AD1E-	B0 EF	BCS	\$AD0F	END OF FILE		
AD20-	EE E4 B5	INC	\$B5E4	} INC SECTOR POSITION		
AD23-	D0 F6	BNE	\$AD1B			
AD25-	EE E5 B5	INC	\$B5E5			
AD28-	4C 1B AD	JMP	\$AD1B	CONTINUE		
AD2B-	20 28 AB	JSR	\$AB28	OPEN FILE	DELETE	
AD2E-	AE 9C B3	LDX	\$B39C	DIRECTORY INDEX		
AD31-	BD C8 B4	LDA	\$B4C8, X	} LOCKED?		
AD34-	10 03	BPL	\$AD39			
AD36-	4C 7B B3	JMP	\$B37B	"FILE LOCKED"		
AD39-	AE 9C B3	LDX	\$B39C	DIRECTORY INDEX		
AD3C-	BD C6 B4	LDA	\$B4C6, X	GET TRACK FOR 1ST T/S LIST		
AD3F-	8D D1 B5	STA	\$B5D1	SET IT IN FIOWA		
AD42-	9D E6 B4	STA	\$B4E6, X	SAVE AT END OF FILENAME		
AD45-	A9 FF	LDA	##FF	} MARK FILE DELETED IN DIRECTORY		
AD47-	9D C6 B4	STA	\$B4C6, X			
AD4A-	BC C7 B4	LDY	\$B4C7, X	} SECTOR OF 1ST T/S LIST TO FIOWA		
AD4D-	8C D2 B5	STY	\$B5D2			
AD50-	20 37 B0	JSR	\$B037	WRITE DIRECTORY SECTOR BACK		
AD53-	18	CLC		} FIND FIRST T/S LIST SECTOR		
AD54-	20 5E AF	JSR	\$AF5E			
AD57-	B0 2A	BCS	\$AD83	SELECT T/S LIST BUFFER		
AD59-	20 0C AF	JSR	\$AF0C	} BUFFER INDEX TO FIRST ENTRY		
AD5C-	A0 0C	LDY	##0C			
AD5E-	8C 9C B3	STY	\$B39C			
AD61-	B1 42	LDA	(\$42), Y	GET TRACK NO. OF A DATA SECTOR		
AD63-	30 0B	BMI	\$AD70	} NONE, SKIP IT		
AD65-	F0 09	BEQ	\$AD70			
AD67-	48	PHA		} GET SECTOR NO. TOO		
AD68-	C8	INY				
AD69-	B1 42	LDA	(\$42), Y			
AD6B-	A8	TAY				
AD6C-	68	PLA		} FREE THE DATA SECTOR		
AD6D-	20 89 AD	JSR	\$AD89			
AD70-	AC 9C B3	LDY	\$B39C	} NEXT T/S LIST ENTRY		
AD73-	C8	INY				
AD74-	C8	INY		} GO ON IF MORE IN THIS SECTOR		
AD75-	D0 E7	BNE	\$AD5E			
AD77-	AD D3 B5	LDA	\$B5D3		} T/S OF THIS T/S LIST SECTOR	
AD7A-	AC D4 B5	LDY	\$B5D4			
AD7D-	20 89 AD	JSR	\$AD89	FREE IT		
AD80-	38	SEC		} AND GO GET NEXT ONE IF ANY		
AD81-	B0 D1	BCS	\$AD54			
AD83-	20 FB AF	JSR	\$AFFB			
AD86-	4C 7F B3	JMP	\$B37F	EXIT		
AD89-	38	SEC		} DEALLOCATE SECTOR IN YTOC		
AD8A-	20 DD B2	JSR	\$B2DD			
AD8D-	A9 00	LDA	##00		} (DOS 3.3 #105)	
AD8F-	A2 03	LDX	##03			
AD91-	9D F0 B5	STA	\$B5F0, X			
AD94-	CA	DEX		ZERO SECTOR ALLOCATION		
AD95-	10 FA	BPL	\$AD91	AREA		



CATALOG

HEADER

AD97-	60	RTS	EXIT TO CALLER
AD98-	20 DC AB	JSR	#ABDC INITIALIZE FLOWA
AD9B-	A9 FF	LDA	##FF } ANY VOL # WILL DO
AD9D-	8D F9 B5	STA	\$B5F9
ADA0-	20 F7 AF	JSR	\$AFF7 READ VTOC
ADA3-	A9 16	LDA	##16 } COUNT 22 LINES BEFORE WAITING
ADA5-	8D 9D B3	STA	\$B39D
ADAB-	20 2F AE	JSR	\$AE2F } SKIP 2 LINES
ADAB-	20 2F AE	JSR	\$AE2F
ADAE-	A2 0B	LDX	##0B
ADB0-	BD AF B3	LDA	\$B3AF, X } PRINT "DISK VOLUME "
ADB3-	20 ED FD	JSR	\$FDED
ADB4-	CA	DEX	
ADB7-	10 F7	BPL	\$ADB0
ADB9-	86 45	STX	\$45
ADBB-	AD F6 B7	LDA	\$B7F6 } CONVERT AND PRINT VOL #
ADBE-	85 44	STA	\$44
ADCO-	20 42 AE	JSR	\$AE42
ADC3-	20 2F AE	JSR	\$AE2F } SKIP 2 LINES
ADC6-	20 2F AE	JSR	\$AE2F
ADC9-	18	CLC	
ADCA-	20 11 B0	JSR	\$B011 } READ FIRST DIRECTORY SECTOR
ADCD-	B0 5D	BCS	\$AE2C NO MORE, EXIT
ADCF-	A2 00	LDX	##00
ADD1-	8E 9C B3	STX	\$B39C } SET INDEX
ADD4-	BD C6 B4	LDA	\$B4C6, X GET TRACK
ADD7-	F0 53	BEG	\$AE2C IF 0, DONE
ADD9-	30 4A	BMI	\$AE25 IF FF, DELETED ENTRY
ADDB-	A0 A0	LDY	##A0 ASSUME UNLOCKED
ADDD-	BD C8 B4	LDA	\$B4C8, X GET FILE TYPE
ADE0-	10 02	BPL	\$ADE4 OK
ADE2-	A0 AA	LDY	##AA LOCKED, USE A "*"
ADE4-	98	TYA	
ADE5-	20 ED FD	JSR	\$FDED } PRINT LOCK/UNLOCK FLAG
ADE8-	BD C8 B4	LDA	\$B4C8, X GET FILE TYPE AGAIN
ADEB-	29 7F	AND	##7F
ADED-	A0 07	LDY	##07
* ADEF-	0A	ASL	
* ADF0-	0A	ASL	
ADF1-	B0 03	BCS	\$ADF6 } GET ALPHABETIC REPRESENTATION OF FILE TYPE FROM TABLE
ADF3-	88	DEY	
ADF4-	D0 FA	BNE	\$ADFO
ADF6-	B9 A7 B3	LDA	\$B3A7, Y
ADF9-	20 ED FD	JSR	\$FDED PRINT IT
ADFC-	A9 A0	LDA	##A0 } A BLANK
ADFE-	20 ED FD	JSR	\$FDED
AE01-	BD E7 B4	LDA	\$B4E7, X
AE04-	85 44	STA	\$44
AE06-	BD E8 B4	LDA	\$B4E8, X } CONVERT AND PRINT NUMBER OF SECTORS
AE09-	85 45	STA	\$45
AE0B-	20 42 AE	JSR	\$AE42
AE0E-	A9 A0	LDA	##A0 } A BLANK
AE10-	20 ED FD	JSR	\$FDED
AE13-	E8	INX	
AE14-	E8	INX	
AE15-	E8	INX	
AE16-	A0 1D	LDY	##1D
AE18-	BD C6 B4	LDA	\$B4C6, X } PRINT IT
AE1B-	20 ED FD	JSR	\$FDED
AE1E-	E8	INX	
AE1F-	88	DEY	
AE20-	10 F6	BPL	\$AE18
AE22-	20 2F AE	JSR	\$AE2F NEW LINE

AE25-	20 30 B2	JSR	\$B230	ADVANCE TO NEXT DIRECTORY ENTRY	
AE28-	90 A7	BCC	\$ADD1	DO NEXT	
AE2A-	B0 9E	BCS	\$ADCA	NO MORE THIS SECTOR, READ NEXT	
AE2C-	4C 7F B3	JMP	\$B37F	EXIT WHEN FINISHED	
AE2F-	A9 8D	LDA	#\$8D	} OUTPUT A (CR)	SKIP A LINE
AE31-	20 ED FD	JSR	\$FDED		
AE34-	CE 9D B3	DEC	\$B39D	DECREMENT LINE COUNTER	
AE37-	D0 08	BNE	\$AE41	OK	
AE39-	20 0C FD	JSR	\$FDOC	ELSE, WAIT FOR KEYBOARD (TO DELAY)	
AE3C-	A9 15	LDA	##15	} THEN COUNT 21 LINES	
AE3E-	8D 9D B3	STA	\$B39D		
AE41-	60	RTS			
AE42-	A0 02	LDY	##02	THREE DIGITS	CONVERT #44 VALUE TO 3 DIGIT DEC NUMBER & PRINT IT
AE44-	A9 00	LDA	##00	} INIT ACCUMULATOR	
AE46-	48	PHA			
AE47-	A5 44	LDA	\$44		
AE49-	D9 A4 B3	CMP	\$B3A4,Y	100's, 10's, or UNIT'S	
AE4C-	90 12	BCC	\$AE60	DIGIT REQUIRED?	
AE4E-	F9 A4 B3	SBC	\$B3A4,Y	} YES, SUBTRACT 100, 10, 1	
AE51-	85 44	STA	\$44		
AE53-	A5 45	LDA	\$45		
AE55-	E9 00	SBC	##00	} INCREMENT ACCUMULATOR	
AE57-	85 45	STA	\$45		
AE59-	68	PLA			
AE5A-	69 00	ADC	##00		
AE5C-	48	PHA			
AE5D-	4C 47 AE	JMP	\$AE47	TRY AGAIN.	
AE60-	68	PLA			
AE61-	09 B0	ORA	##B0	} CONVERT ACCUMULATOR	
AE63-	20 ED FD	JSR	\$FDED	PRINT DIGIT	
AE66-	88	DEY			
AE67-	10 DB	BPL	\$AE44	} NEXT DIGIT	
AE69-	60	RTS			
AE6A-	20 08 AF	JSR	\$AF08	PT TO BUFFER #1	RESTORE FIOWA
AE6D-	A0 00	LDY	##00	} START RETURN CODE AT 0	
AE6F-	8C C5 B5	STY	\$B5C5		
AE72-	B1 42	LDA	(\$42),Y	} COPY 45 BYTE SAVED IMAGE OF FIOWORKAREA FROM BUFFER	
AE74-	99 D1 B5	STA	\$B5D1,Y		
AE77-	C8	INY			
AE78-	C0 2D	CPY	##2D		
AE7A-	D0 F6	BNE	\$AE72		
AE7C-	18	CLC			
AE7D-	60	RTS			
AE7E-	20 08 AF	JSR	\$AF08	SELECT FIOWA BUFFER	SAVE FIOWA
AE81-	A0 00	LDY	##00		
AE83-	B9 D1 B5	LDA	\$B5D1,Y	} SAVE 45 BYTE FIOWA INTO BUFFER	
AE86-	91 42	STA	(\$42),Y		
AE88-	C8	INY			
AE89-	C0 2D	CPY	##2D		
AE8B-	D0 F6	BNE	\$AE83		
AE8D-	60	RTS			
AE8E-	20 DC AB	JSR	\$ABDC	INITIALIZE FIOWA	INIT
AE91-	A9 04	LDA	##04	} FORMAT DISKETTE w/RWTS	
AE93-	20 58 B0	JSR	\$B058		
AE96-	AD F9 B5	LDA	\$B5F9	} PUT VOL # INTO VTOC	
AE99-	49 FF	EOR	##FF		
AE9B-	8D C1 B3	STA	\$B3C1		
AE9E-	A9 11	LDA	##11	} AND TRACK TO ALLOCATE NEXT	
AEA0-	8D EB B3	STA	\$B3EB		
AEA3-	A9 01	LDA	##01	} DIRECTION OF ALLOCATION (FORWARD)	
AEA5-	8D EC B3	STA	\$B3EC		
AEA8-	A2 38	LDX	##38		
AEAA-	A9 00	LDA	##00	OFFSET TO VOLUME BIT MAP	



AEAC-	9D BB B3	STA	#B3BB, X	} ZERO VOLUME SPACE (ALL IN USE)
AEAF-	E8	INX		
AEBO-	D0 FA	BNE	\$AEAC	
AEB2-	A2 0C	LDX	##0C	} SKIP 1ST 3 TRACKS (DOS BOOT SPACE)
AEB4-	E0 8C	CPX	##8C	
AEB6-	F0 14	BEQ	\$AECC	} END OF MAP?
AEB8-	A0 03	LDY	##03	
AEBA-	B9 A0 B3	LDA	\$B3A0, Y	} COPY 4 BYTE BIT MASK TO ENTRY
AEBD-	9D F3 B3	STA	\$B3F3, X	
AECO-	E8	INX		
AEC1-	88	DEY		
AEC2-	10 F6	BPL	\$AEBA	} SKIP ALSO DIRECTORY TRACK (17)
AEC4-	E0 44	CPX	##44	
AEC6-	D0 EC	BNE	\$AEB4	
AEC8-	A2 48	LDX	##48	
AECA-	D0 E8	BNE	\$AEB4	} ZERO DIRECTORY SECTOR BUFFER
AECB-	20 FB AF	JSR	\$AFFB	
AECF-	A2 00	LDX	##00	} ZERO DIRECTORY SECTOR BUFFER
AED1-	8A	TXA		
AED2-	9D BB B4	STA	\$B4BB, X	} ZERO DIRECTORY SECTOR BUFFER
AED5-	E8	INX		
AED6-	D0 FA	BNE	\$AED2	} POINT RWTS TO DIRECTORY BUFFER
AED8-	20 45 B0	JSR	\$B045	
AEDB-	A9 11	LDA	##11	} TRACK 17
AEDD-	AC F0 B3	LDY	\$B3F0	
AEE0-	88	DEY		} LAST SECTOR ON TRACK
AEE1-	88	DEY		
AEE2-	8D EC B7	STA	\$B7EC	} PASS TRACK NO. TO RWTS
AEE5-	8D BC B4	STA	\$B4BC	
AEE8-	8C BD B4	STY	\$B4BD	} POINT THIS DIR. SECTOR TO NEXT (S-1)
AEEB-	C8	INY		
AEEC-	8C ED B7	STY	\$B7ED	} RWTS SECTOR NO. FOR THIS DIR. SECT.
AEEF-	A9 02	LDA	##02	
AEF1-	20 58 B0	JSR	\$B058	} GO TO RWTS TO WRITE SECTOR
AEF4-	AC BD B4	LDY	\$B4BD	
AEF7-	88	DEY		} GET NEXT SECTOR AND NEXT
AEF8-	30 05	BMI	\$AEFF	
AEFA-	D0 EC	BNE	\$AEE8	} INITIALIZE ALL BUT SECTOR 0
AEFC-	98	TYA		
AEFD-	F0 E6	BEQ	\$AEE5	} ON LAST ONE, ZERO TRACK PTR
AEFF-	20 C2 B7	JSR	\$B7C2	
AF02-	20 4A B7	JSR	\$B74A	} POINT RWTS PARMS AT DOS LOAD POINT
AF05-	4C 7F B3	JMP	\$B37F	
AF08-	A2 00	LDX	##00	} SELECT BUFFER #1 (FIOWA)
AF0A-	F0 06	BEQ	\$AF12	
AF0C-	A2 02	LDX	##02	} SELECT BUFFER #2 (T/S LIST)
AF0E-	D0 02	BNE	\$AF12	
AF10-	A2 04	LDX	##04	} SELECT BUFFER #3 (DATA)
AF12-	BD C7 B5	LDA	\$B5C7, X	
AF15-	85 42	STA	\$42	} SET \$42, \$43 FROM FIO PARMLIST
AF17-	BD C8 B5	LDA	\$B5C8, X	
AF1A-	85 43	STA	\$43	
AF1C-	60	RTS		
AF1D-	2C D5 B5	BIT	\$B5D5	} DATA BUFFER CHANGED?
AF20-	70 01	BVS	\$AF23	
AF22-	60	RTS		} NO, EXIT
AF23-	20 E4 AF	JSR	\$AFE4	
AF26-	A9 02	LDA	##02	} GO WRITE USING RWTS
AF28-	20 52 B0	JSR	\$B052	
AF2B-	A9 BF	LDA	##BF	} DATA BUFFER NO LONGER NEEDS CHECKPOINT
AF2D-	2D D5 B5	AND	\$B5D5	
AF30-	8D D5 B5	STA	\$B5D5	
AF33-	60	RTS		



POINT TO A BUFFER
-------------------

CHECKPOINT DATA SECTOR BUFFER (#3)
------------------------------------

AF34-	AD D5 B5	LDA	#B5D5	} T/S LIST CHANGED? NO	CHECKPOINT TRACK/SECTOR LIST BUFFER (#2)
AF37-	30 01	BMI	#AF3A		
AF39-	60	RTS			
AF3A-	20 4B AF	→ JSR	#AF4B	SET UP RWTS PTR	
AF3D-	A9 02	LDA	##02	} WRITE SECTOR	
AF3F-	20 52 B0	JSR	#B052		
AF42-	A9 7F	LDA	##7F		
AF44-	2D D5 B5	AND	#B5D5	} T/S LIST NO LONGER NEEDS CHECKPOINTING	
AF47-	8D D5 B5	STA	#B5D5		
AF4A-	60	RTS			
AF4B-	AD C9 B5	LDA	#B5C9	} COPY ADDRESS OF T/S LIST BUFF TO RWTS PARAMS	PREPARE FOR RWTS WITH T/S LIST
AF4E-	8D F0 B7	STA	#B7F0		
AF51-	AD CA B5	LDA	#B5CA		
AF54-	8D F1 B7	STA	#B7F1		
AF57-	AE D3 B5	LDX	#B5D3	} GET T/S FOR IT	
AF5A-	AC D4 B5	LDY	#B5D4		
AF5D-	60	RTS			
AF5E-	08	PHP		MEMORIZE ENTRY CODE	GET A T/S LIST SECTOR  C=0: FIRST C=1: NEX
AF5F-	20 34 AF	JSR	#AF34	CHECKPOINT CURR. T/S LIST	
AF62-	20 4B AF	JSR	#AF4B	SET UP FOR RWTS	
AF65-	20 0C AF	JSR	#AF0C	SELECT T/S LIST BUFF.	
AF68-	28	PLP		} FIRST OR NEXT?	
AF69-	B0 09	BCS	#AF74		
AF6B-	AE D1 B5	LDX	#B5D1	} FIRST, WHERE IS IT?	
AF6E-	AC D2 B5	LDY	#B5D2		
AF71-	4C B5 AF	JMP	#AFB5		
AF74-	A0 01	→ LDY	##01	NEXT, GET TRK	
AF76-	B1 42	LDA	(#42), Y	} ARE THERE ANY MORE AVAILABLE?	
AF78-	F0 08	BEQ	#AF82		
AF7A-	AA	TAX		} YES, GET SECTOR OF NEXT AND GO!	
AF7B-	C8	INY			
AF7C-	B1 42	LDA	(#42), Y		
AF7E-	A8	TAY		} NO MORE T/S LISTS, ARE WE WRITING TO FILE?	
AF7F-	4C B5 AF	← JMP	#AFB5		
AF82-	AD BB B5	→ LDA	#B5BB	} NO, EXIT WITH END-OF-FILE ERROR	
AF85-	C9 04	CMP	##04		
AF87-	F0 02	BEQ	#AF8B		
AF89-	38	SEC		} YES, ALLOCATE A NEW SECTOR	
AF8A-	60	RTS			
AF8B-	20 44 B2	→ JSR	#B244	} POINT OLD T/S LIST TO NEW	
AF8E-	A0 02	LDY	##02		
AF90-	91 42	STA	(#42), Y		
AF92-	48	PHA			
AF93-	88	DEY			
AF94-	AD F1 B5	LDA	#B5F1		
AF97-	91 42	STA	(#42), Y	} WRITE T/S LIST BUFFER ZERO BUFFER	
AF99-	48	PHA			
AF9A-	20 3A AF	JSR	#AF3A		
AF9D-	20 D6 B7	JSR	#B7D6	} +5, +6 CONTAIN RELATIVE SECTOR # OF FIRST SECTOR IN THIS T/S LIST	
AFA0-	A0 05	LDY	##05		
AFA2-	AD DE B5	LDA	#B5DE		
AFA5-	91 42	STA	(#42), Y		
AFA7-	C8	INY			
AFA8-	AD DF B5	LDA	#B5DF		
AFAB-	91 42	STA	(#42), Y	} T/S OF THIS LIST	
AFAD-	68	FLA			
AFAE-	AA	TAX			
AFAF-	68	FLA		} WRITE IT (READ IT IF OLD)	
AFB0-	A8	TAY			
AFB1-	A9 02	LDA	##02		
AFB3-	D0 02	BNE	#AFB7	} SET TRACK	
AFB5-	A9 01	→ LDA	##01		
AFB7-	8E D3 B5	STX	#B5D3		



AFBA-	8C D4 B5	STY	#B5D4	SET SECTOR
AFBD-	20 52 B0	JSR	#B052	CALL RWTS
AFCO-	A0 05	LDY	#05	
AFC2-	B1 42	LDA	(\$42),Y	
AFC4-	8D DC B5	STA	#B5DC	} BUFF+5,+6 → REL. SECT. # FIRST SECT + NUMBER OF SECTORS PER LIST
AFC7-	18	CLC		
AFC8-	6D DA B5	ADC	#B5DA	} = REL. SECT. # OF LAST+1 SECTOR IN LIST
AFCB-	8D DE B5	STA	#B5DE	
AFCE-	C8	INY		
AFCF-	B1 42	LDA	(\$42),Y	
AFD1-	8D DD B5	STA	#B5DD	
AFD4-	6D DB B5	ADC	#B5DB	
AFD7-	8D DF B5	STA	#B5DF	
AFDA-	18	CLC		} EXIT AND SAY WE FOUND IT
AFDB-	60	RTS		

AFDC-	20 E4 AF	JSR	#AFE4	SET UP FOR RWTS	} READ A DATA SECT
AFDF-	A9 01	LDA	#01	READ OPCODE	
AFE1-	4C 52 B0	JMP	#B052	GO DO IT	
AFE4-	AC CB B5	LDY	#B5CB	} COPY ADDRESS OF DATA SECTOR BUFF TO RWTS PARAMS	} PREPARE FOR RWTS WITH DATA
AFE7-	AD CC B5	LDA	#B5CC		
AFEA-	8C F0 B7	STY	#B7F0		
AFED-	8D F1 B7	STA	#B7F1		
AFF0-	AE D6 B5	LDX	#B5D6	} GET T/S FOR IT	
AFF3-	AC D7 B5	LDY	#B5D7		
AFF6-	60	RTS			

AFF7-	A9 01	LDA	#01	READ	} READ / WRITE VTOC
AFF9-	D0 02	BNE	#AFFD	OR	
AFFB-	A9 02	LDA	#02	WRITE	
AFFD-	AC C3 AA	LDY	#AAC3	} COPY VTOC SECTOR BUFFER ADDRESS TO RWTS PARAMS	
B000-	8C F0 B7	STY	#B7F0		
B003-	AC C4 AA	LDY	#AAC4		
B006-	8C F1 B7	STY	#B7F1		
B009-	AE FA B5	LDX	#B5FA	GET TRACK NO. AND SECTOR	
B00C-	A0 00	LDY	#00		
B00E-	4C 52 B0	JMP	#B052	AND GO	

B011-	08	PHP		MEMORIZE ENTRY CODE	} READ A DIRECTORY SECTOR
B012-	20 45 B0	JSR	#B045	SET BUFFER POINTERS	
B015-	28	PLP		} FIRST OR NEXT?	} C=0 FIRST C=1 NEXT
B016-	B0 08	BCS	#B020		
B018-	AC BD B3	LDY	#B3BD	} FIRST, GET T/S FROM VTOC+1	
B01B-	AE BC B3	LDX	#B3BC		
B01E-	D0 0A	BNE	#B02A	} NEXT, GET TRACK FROM DIRECTORY. IF 0...	
B020-	AE BC B4	LDX	#B4BC		
B023-	D0 02	BNE	#B027	} WE ARE AT END OF DIRECTORY	
B025-	38	SEC			
B026-	60	RTS			
B027-	AC BD B4	LDY	#B4BD	GET SECTOR IF NEXT	
B02A-	8E 97 B3	STX	#B397	} SAVE T/S OF THIS SECTOR	
B02D-	8C 98 B3	STY	#B398		
B030-	A9 01	LDA	#01	} READ IT	
B032-	20 52 B0	JSR	#B052		
B035-	18	CLC		} AND EXIT	
B036-	60	RTS			

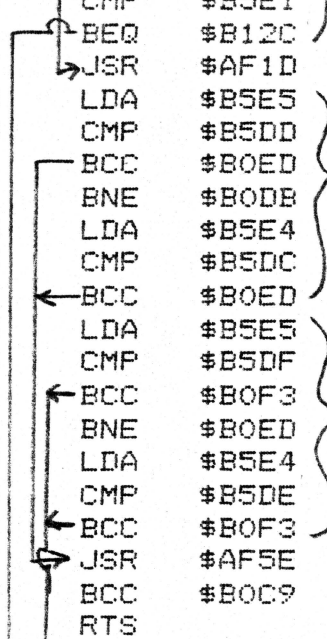
B037-	20 45 B0	JSR	#B045	SET BUFFER PTRS	} WRITE DIRECTOR SECTOR
B03A-	AE 97 B3	LDX	#B397		
B03D-	AC 98 B3	LDY	#B398	} FIND ITS T/S	
B040-	A9 02	LDA	#02	} GO WRITE IT AND EXIT	
B042-	4C 52 B0	JMP	#B052		

B045-	AD C5 AA	LDA	#AAC5	} COPY DIRECTORY BUFFER ADDRESS TO RWTS PARAMS	} PREPARE FOR RWTS FOR DIRECTORY
B048-	8D F0 B7	STA	#B7F0		
B04B-	AD C6 AA	LDA	#AAC6		
B04E-	8D F1 B7	STA	#B7F1		
B051-	60	RTS			



RWTS DRIVER

B052-	8E EC B7	STX	\$B7EC	} SET TRACK/SECTOR	RWTS DRIVER
B055-	8C ED B7	STY	\$B7ED		
B058-	8D F4 B7	STA	\$B7F4	} AND COMMAND CODE WRITE?	RWTS DRIVER
B05B-	C9 02	CMF	##02		
B05D-	D0 06	BNE	\$B065	} NO	RWTS DRIVER
B05F-	0D D5 B5	ORA	\$B5D5		
B062-	8D D5 B5	STA	\$B5D5	} SET FLAG	RWTS DRIVER
B065-	AD F9 B5	LDA	\$B5F9		
B068-	49 FF	EOR	##FF	} SET VOL # EXPECTED	RWTS DRIVER
B06A-	8D EB B7	STA	\$B7EB		
B06D-	AD F7 B5	LDA	\$B5F7	} SLOT #16	RWTS DRIVER
B070-	8D E9 B7	STA	\$B7E9		
B073-	AD F8 B5	LDA	\$B5F8	} DRIVE	RWTS DRIVER
B076-	8D EA B7	STA	\$B7EA		
B079-	AD E2 B5	LDA	\$B5E2	} SECTOR SIZE	RWTS DRIVER
B07C-	8D F2 B7	STA	\$B7F2		
B07F-	AD E3 B5	LDA	\$B5E3	} SECTOR SIZE	RWTS DRIVER
B082-	8D F3 B7	STA	\$B7F3		
B085-	A9 01	LDA	##01	} JOB TYPE	RWTS DRIVER
B087-	8D E8 B7	STA	\$B7E8		
B08A-	AC C1 AA	LDY	\$AAC1	} PASS ↑ TO RWTS PARMS	RWTS DRIVER
B08D-	AD C2 AA	LDA	\$AAC2		
B090-	20 B5 B7	JSR	\$B7B5	} GO TO RWTS	RWTS DRIVER
B093-	AD F6 B7	LDA	\$B7F6		
B096-	8D BF B5	STA	\$B5BF	} RETURN TRUE VOL# TO FIO PARMS	RWTS DRIVER
B099-	A9 FF	LDA	##FF		
B09B-	8D EB B7	STA	\$B7EB	} WATCH FOR CHANGES	RWTS DRIVER
B09E-	B0 01	BCS	\$B0A1		
B0A0-	60	RTS		} NO ERROR, EXIT NOW	RWTS DRIVER
B0A1-	AD F5 B7	LDA	\$B7F5		
B0A4-	A0 07	LDY	##07	} ERROR, GET RETURN CODE	RWTS DRIVER
B0A6-	C9 20	CMF	##20		
B0A8-	F0 08	BEQ	\$B0B2	} IF VOL MISMATCH, RC=7	RWTS DRIVER
B0AA-	A0 04	LDY	##04		
B0AC-	C9 10	CMF	##10	} IF WRITE PROTECTED, RC=4	RWTS DRIVER
B0AE-	F0 02	BEQ	\$B0B2		
B0B0-	A0 08	LDY	##08	} ALL OTHERS ARE RC=8 "I/O ERROR"	RWTS DRIVER
B0B2-	98	TYA			
B0B3-	4C 85 B3	JMP	\$B385	} CLEAR OUT OF FIO NOW!	RWTS DRIVER
B0B6-	AD E4 B5	LDA	\$B5E4		
B0B9-	CD E0 B5	CMF	\$B5E0	} CURRENT POSITION IN CURRENT SECTOR?	READ NEXT DATA SECTOR
B0BC-	D0 08	BNE	\$B0C6		
B0BE-	AD E5 B5	LDA	\$B5E5	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0C1-	CD E1 B5	CMF	\$B5E1		
B0C4-	F0 66	BEQ	\$B12C	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0C6-	20 1D AF	JSR	\$AF1D		
B0C9-	AD E5 B5	LDA	\$B5E5	} CURRENT POSITION PRIOR TO THIS T/S LIST'S DOMAIN?	RWTS DRIVER
B0CC-	CD DD B5	CMF	\$B5DD		
B0CF-	90 1C	BCC	\$B0ED	} CURRENT POSITION PRIOR TO THIS T/S LIST'S DOMAIN?	RWTS DRIVER
B0D1-	D0 08	BNE	\$B0DB		
B0D3-	AD E4 B5	LDA	\$B5E4	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0D6-	CD DC B5	CMF	\$B5DC		
B0D9-	90 12	BCC	\$B0ED	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0DB-	AD E5 B5	LDA	\$B5E5		
B0DE-	CD DF B5	CMF	\$B5DF	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0E1-	90 10	BCC	\$B0F3		
B0E3-	D0 08	BNE	\$B0ED	} CURRENT POSITION PAST THIS T/S LIST'S DOMAIN?	RWTS DRIVER
B0E5-	AD E4 B5	LDA	\$B5E4		
B0E8-	CD DE B5	CMF	\$B5DE	} NO, CHECKPOINT DATA BUFFER	RWTS DRIVER
B0EB-	90 06	BCC	\$B0F3		
B0ED-	20 5E AF	JSR	\$AF5E	} GET {NEXT} T/S LIST AND TRY AGAIN	RWTS DRIVER
B0F0-	90 D7	BCC	\$B0C9		
B0F2-	60	RTS		} RAN OFF END OF FILE READING	RWTS DRIVER



B0F3-	38	→ SEC		DATA IS IN THIS T/S LIST
B0F4-	AD E4 B5	LDA	\$B5E4	} COMPUTE DISPLACEMENT TO PROPER ENTRY IN T/S LIST
B0F7-	ED DC B5	SBC	\$B5DC	
B0FA-	0A	ASL		
B0FB-	69 0C	ADC	##0C	
B0FD-	A8	TAY		
B0FE-	20 0C AF	JSR	\$AF0C	SELECT BUFFER
B101-	B1 42	LDA	(#42),Y	GET TRACK.
B103-	D0 0F	BNE	\$B114	OK?
B105-	AD BB B5	LDA	\$B5BB	} NO SECTOR HERE, WRITING?
B108-	C9 04	CMF	##04	
B10A-	F0 02	BEQ	\$B10E	YES, GO MAKE ONE
B10C-	38	SEC		} NO READ INTO EMPTY AREA
B10D-	60	RTS		
B10E-	20 34 B1	→ JSR	\$B134	ADD NEW SECTOR
B111-	4C 20 B1	→ JMP	\$B120	AND GO ON
B114-	8D D6 B5	→ STA	\$B5D6	OLD SECTOR, SET ITS TRACK
B117-	C8	INY		} AND SECTOR #
B118-	B1 42	LDA	(#42),Y	
B11A-	8D D7 B5	STA	\$B5D7	} READ IT IN
B11D-	20 DC AF	JSR	\$AFDC	
B120-	AD E4 B5	→ LDA	\$B5E4	} SET SECTOR LAST READ #
B123-	8D E0 B5	STA	\$B5E0	
B126-	AD E5 B5	LDA	\$B5E5	
B129-	8D E1 B5	STA	\$B5E1	} SELECT DATA BUFFER
B12C-	20 10 AF	→ JSR	\$AF10	
B12F-	AC E6 B5	LDY	\$B5E6	GET BYTE OFFSET
B132-	18	CLC		} EXIT NORMALLY
B133-	60	RTS		

B134-	8C 9D B3	STY	\$B39D	SAVE YREG
B137-	20 44 B2	JSR	\$B244	GO ALLOCATE SECTOR
B13A-	AC 9D B3	LDY	\$B39D	RESTORE YREG
B13D-	C8	INY		
B13E-	91 42	STA	(#42),Y	} PUT SECTOR # IN LIST
B140-	8D D7 B5	STA	\$B5D7	
B143-	88	DEY		} AND TRACK #
B144-	AD F1 B5	LDA	\$B5F1	
B147-	91 42	STA	(#42),Y	
B149-	8D D6 B5	STA	\$B5D6	} SELECT DATA BUFFER
B14C-	20 10 AF	JSR	\$AF10	
B14F-	20 D6 B7	JSR	\$B7D6	ZERO IT
B152-	A9 C0	LDA	##C0	} T/S LIST & DATA SECTORS NEED CHECKPOINT
B154-	0D D5 B5	ORA	\$B5D5	
B157-	8D D5 B5	STA	\$B5D5	
B15A-	60	RTS		

ADD A NEW DATA SECTOR

B15B-	AE EA B5	LDX	\$B5EA	} RETURN RECORD # TO FIO CALLER
B15E-	8E BD B5	STX	\$B5BD	
B161-	AE EB B5	LDX	\$B5EB	} PASS BYTE OFFSET ALSO
B164-	8E BE B5	STX	\$B5BE	
B167-	AE EC B5	LDX	\$B5EC	} INC BYTE #
B16A-	AC ED B5	LDY	\$B5ED	
B16D-	8E BF B5	STX	\$B5BF	} AT RECORD LENGTH?
B170-	8C C0 B5	STY	\$B5C0	
B173-	E8	INX		} YES, BYTE # 0
B174-	D0 01	BNE	\$B177	
B176-	C8	INY		} AT RECORD LENGTH?
B177-	CC E9 B5	→ CPY	\$B5E9	
B17A-	D0 11	BNE	\$B18D	} YES, BYTE # 0
B17C-	EC E8 B5	CPX	\$B5E8	
B17F-	D0 0C	BNE	\$B18D	} YES, BYTE # 0
B181-	A2 00	LDX	##00	
B183-	A0 00	LDY	##00	} YES, BYTE # 0
B185-	EE EA B5	INC	\$B5EA	

INCREMENT RECORD # & BYTE #

```

B188- DO 03 BNE $B18D } NEXT RECORD #
B18A- EE EB B5 INC $B5EB }
B18D- 8E EC B5 STX $B5EC } AND BYTE #
B190- 8C ED B5 STY $B5ED }
B193- 60 RTS
B194- EE E6 B5 INC $B5E6 BUMP SECTOR BYTE OFFSET
B197- DO 08 BNE $B1A1 IF AT END...
B199- EE E4 B5 INC $B5E4 }
B19C- DO 03 BNE $B1A1 } NEXT SECTOR #
B19E- EE E5 B5 INC $B5E5 }
B1A1- 60 RTS
B1A2- AC C3 B5 LDY $B5C3 }
B1A5- AE C4 B5 LDX $B5C4 } COPY ADDRESS TO $42
B1A8- 84 42 STY $42 }
B1AA- 86 43 STX $43 }
B1AC- EE C3 B5 INC $B5C3 }
B1AF- DO 03 BNE $B1B4 } ADVANCE FOR NEXT TIME
B1B1- EE C4 B5 INC $B5C4 }
B1B4- 60 RTS
B1B5- AC C1 B5 LDY $B5C1 }
B1B8- DO 08 BNE $B1C2 }
B1BA- AE C2 B5 LDX $B5C2 }
B1BD- F0 07 BEQ $B1C6 } DECREMENT LENGTH
R1BF- CE C2 B5 DEC $B5C2 } IN FIO PARAM LIST
B1C2- CE C1 B5 DEC $B5C1 } BY ONE
B1C5- 60 RTS
B1C6- 4C 7F B3 JMP $B37F IF ZERO, EXIT FIO
B1C9- 20 F7 AF JSR $AFF7 READ VTOC
B1CC- AD C3 B5 LDA $B5C3 }
B1CF- 85 42 STA $42 } SET UP POINTER TO
B1D1- AD C4 B5 LDA $B5C4 } FILENAME PASSED
B1D4- 85 43 STA $43 }
B1D6- A9 01 LDA #$01 } 1ST PASS, LOCATE FILE
B1D8- 8D 9D B3 STA $B39D }
B1DB- A9 00 LDA #$00 } START DIRECTORY SECTOR
B1DD- 8D D8 B5 STA $B5D8 } OFFSET
B1E0- 18 CLC GET FIRST DIRECTORY SECTOR
B1E1- EE D8 B5 INC $B5D8 BUMP SECTOR OFFSET
B1E4- 20 11 B0 JSR $B011 GET DIRECTORY SECTOR
B1E7- B0 51 BCS $B23A END OF DIRECTORY? NEW PASS
B1E9- A2 00 LDX #$00 } START ENTRY INDEX
B1EB- 8E 9C B3 STX $B39C }
B1EE- BD C6 B4 LDA $B4C6,X GET TRACK
B1F1- F0 1F BEQ $B212 EMPTY ENTRY/END OF DIRECTORY?
B1F3- 30 22 BMI $B217 DELETED ENTRY?
B1F5- A0 00 LDY #$00 }
B1F7- E8 INX }
B1F8- E8 INX } BUMP TO FILENAME
B1F9- E8 INX }
B1FA- B1 42 LDA ($42),Y }
B1FC- DD C6 B4 CMP $B4C6,X } IS THIS THE NAME
B1FF- DO 0A BNE $B20B } WE ARE LOOKING FOR?
B201- C8 INY }
B202- C0 1E CPY #$1E }
B204- DO F3 BNE $B1F9 }
B206- AE 9C B3 LDX $B39C }
B209- 18 CLC } YES, RETURN INDEX AND EXIT
B20A- 60 RTS }
B20B- 20 30 B2 JSR $B230 NO, NEXT ENTRY
B20E- 90 DB BCC $B1EB CHECK IT OUT
B210- B0 CF BCS $B1E1 END OF SECTOR, GO GET NEXT
B212- AC 9D B3 LDY $B39D } 1ST PASS? YES, GO TO SECOND
B215- DO C1 BNE $B1D8 }

```

INCREMENT  
FILE OFFSET

COPY AND  
ADVANCE  
RANGE  
ADDRESS

DECREMENT  
RANGE  
LENGTH

LOCATE OR  
ALLOCATE  
A DIRECTOR  
ENTRY



B217-	AC 9D B3	→ LDY	\$B39D	} 1ST PASS? YES, SKIP ENTRY	
B21A-	D0 EF	BNE	\$B20B		
B21C-	A0 00	LDY	##00	} 2ND PASS, ALLOCATE ENTRY	COPY FILE NAME TO DIRECTORY
B21E-	E8	INX			
B21F-	E8	INX		} BUMP PAST T/S & TYPE	
B220-	E8	→ INX			
B221-	B1 42	LDA	(#42),Y	} COPY NAME TO DIRECTORY	
B223-	9D C6 B4	STA	\$B4C6,X		
B226-	C8	INX			
B227-	C0 1E	CPY	##1E		
B229-	D0 F5	BNE	\$B220		
B22B-	AE 9C B3	LDX	\$B39C		RESTORE DIRECTORY INDEX
B22E-	38	SEC			
B22F-	60	RTS			
B230-	18	CLC		} ADD 35 (LENGTH OF ENTRY) TO... INDEX AT END OF SECTOR?	ADVANCE TO NEXT DIRECTORY ENTRY
B231-	AD 9C B3	LDA	\$B39C		
B234-	69 23	ADC	##23		
B236-	AA	TAX			
B237-	E0 F5	CPX	##F5		
B239-	60	RTS			
B23A-	A9 00	LDA	##00	} ON PASS 1? THEN GO DO 2	SWITCH TO 2ND PASS IN DIRECTORY
B23C-	AC 9D B3	LDY	\$B39D		
B23F-	D0 97	BNE	\$B1D8		
B241-	4C 77 B3	JMP	\$B377		
B244-	AD F1 B5	LDA	\$B5F1	} WORKING ON A TRACK NOW?	ALLOCATE A DISK SECTOR
B247-	F0 21	BEQ	\$B26A		
B249-	CE F0 B5	→ DEC	\$B5F0	} YES, WHAT IS NEXT SECT. # TRACK USED UP?	
B24C-	30 17	BMI	\$B265		
B24E-	18	CLC			
B24F-	A2 04	LDX	##04	} ROTATE TRACK MASK LEFT BY ONE	
B251-	3E F1 B5	ROL	\$B5F1,X		
B254-	CA	DEX			
B255-	D0 FA	BNE	\$B251	} THIS TRACK IN USE?	
B257-	90 F0	BCC	\$B249		
B259-	EE EE B5	INC	\$B5EE	} NO, FILE HAS ONE MORE SECTOR NOW	
B25C-	D0 03	BNE	\$B261		
B25E-	EE EF B5	INC	\$B5EF		
B261-	AD F0 B5	LDA	\$B5F0		PASS BACK SECTOR # (TRACK IS @ B5F1)
B264-	60	RTS			
B265-	A9 00	→ LDA	##00	} NO TRACK BEING USED	FIND A TRACK WITH FREE SECTORS
B267-	8D F1 B5	STA	\$B5F1		
B26A-	A9 00	→ LDA	##00	} ALLOW ONE CYCLE AT LEAST	
B26C-	8D 9E B3	STA	\$B39E		
B26F-	20 F7 AF	JSR	\$AFF7		READ VTOC
B272-	18	CLC			
B273-	AD EB B3	LDA	\$B3EB	} GET LAST TRK TO ALLOCATED FROM GO PROPER DIRECTION (-1 or +1) BACK TO TRK 0?	
B276-	6D EC B3	ADC	\$B3EC		
B279-	F0 09	BEQ	\$B284	} OUT PAST TRK 34?	
B27B-	CD EF B3	CMP	\$B3EF		
B27E-	90 14	BCC	\$B294		
B280-	A9 FF	LDA	##FF	} YES, REVERSE ALLOCATION DIRECTION	
B282-	D0 0A	BNE	\$B28E		
B284-	AD 9E B3	→ LDA	\$B39E	} 2ND TIME AT TRK 0?	
B287-	D0 37	BNE	\$B2C0		
B289-	A9 01	LDA	##01	} NO, BUT REMEMBER WE WERE HERE	
B28B-	8D 9E B3	STA	\$B39E		
B28E-	8D EC B3	STA	\$B3EC		NEW DIRECTION (+1 or -1)
B291-	18	CLC			
B292-	69 11	ADC	##11	} BEGIN AT DIRECTORY TRACK (17 ± 1)	
B294-	8D EB B3	STA	\$B3EB		
B297-	8D F1 B5	STA	\$B5F1		
B29A-	A8	TAY			
B29B-	0A	ASL			



B29C-	0A	ASL		} COMPUTE BIT MAP INDEX (TRK*4)		
B29D-	A8	TAY				
B29E-	A2 04	LDX	##04	4 BYTES TO CHECK		
B2A0-	18	CLC		ASSUME TRK FULL		
B2A1-	B9 F6 B3	LDA	\$B3F6,Y	} COPY TRK MASK TO FLOWA		
B2A4-	9D F1 B5	STA	\$B5F1,X		8 SECTORS IN USE?	
B2A7-	F0 06	BEQ	\$B2AF		TRK NOT FULL	
B2A9-	38	SEC				
B2AA-	A9 00	LDA	##00	} PREALLOCATE ALL SECTORS ON TRK		
B2AC-	99 F6 B3	STA	\$B3F6,Y		NEXT BYTE	
B2AF-	88	DEY				
B2B0-	CA	DEX				
B2B1-	D0 EE	BNE	\$B2A1			
B2B3-	90 BD	BCC	\$B272	NO FREE SECTORS FOUND THIS TRK?	TRY NEXT	
B2B5-	20 FB AF	JSR	\$AFFB	FOUND SOME, WRITE VTOC		
B2B8-	AD F0 B3	LDA	\$B3F0	} START WITH LAST SECTOR IN TRK		
B2BB-	8D F0 B5	STA	\$B5F0			
B2BE-	D0 89	BNE	\$B249	GO GET A FREE SECTOR		
B2C0-	4C 77 B3	JMP	\$B377	NO SPACE, EXIT		
B2C3-	AD F1 B5	LDA	\$B5F1	} PREALLOCATED TRK?	RELEASE PRE-ALLOCATED TRK AND CHECKPOINT VTOC	
B2C6-	D0 01	BNE	\$B2C9			NO
B2C8-	60	RTS		YES, SAVE IT		
B2C9-	48	PHA		READ VTOC		
B2CA-	20 F7 AF	JSR	\$AFF7	SECTOR # (SHIFT COUNT)		
B2CD-	AC F0 B5	LDY	\$B5F0	AND TRK #		
B2D0-	68	PLA		DON'T FREE ANYTHING		
B2D1-	18	CLC		GO UPDATE VTOC BIT MAP		
B2D2-	20 DD B2	JSR	\$B2DD			
B2D5-	A9 00	LDA	##00	} NO TRACK PREALLOCATED NOW		
B2D7-	8D F1 B5	STA	\$B5F1		EXIT BY WRITING VTOC	
B2DA-	4C FB AF	JMP	\$AFFB			
B2DD-	A2 FC	LDX	##FC	DO 4 BYTES FORWARD	FREE A SECTOR OR SHIFT MASK TO VTOC	
B2DF-	7E F6 B4	ROR	\$B4F6,X	} ROTATE TRK MASK IN FLOWA RIGHT, ADDING CARRY		
B2E2-	E8	INX				NEXT SECTOR
B2E3-	D0 FA	BNE	\$B2DF	SHIFTED BACK TO END OF TRK?		
B2E5-	C8	INY		NO, CONTINUE		
B2E6-	CC F0 B3	CPY	\$B3F0	} TRK * 4 INDEX TO MAP		
B2E9-	D0 F2	BNE	\$B2DD			
B2EB-	0A	ASL				
B2EC-	0A	ASL				
B2ED-	A8	TAY				
B2EE-	F0 0F	BEQ	\$B2FF	DON'T FOOL WITH TRK 0		
B2F0-	A2 04	LDX	##04			
B2F2-	BD F1 B5	LDA	\$B5F1,X	} "OR" 4 BYTE MASK IN FLOWA INTO MAP		
B2F5-	19 F6 B3	ORA	\$B3F6,Y			
B2F8-	99 F6 B3	STA	\$B3F6,Y			
B2FB-	88	DEY				
B2FC-	CA	DEX				
B2FD-	D0 F3	BNE	\$B2F2			
B2FF-	60	RTS				
B300-	AD BD B5	LDA	\$B5BD	} SET RECORD # PASSED IN FLOWA AND IN SECTOR OFFSETS	CALCULATE FILE POSITION	
B303-	8D E6 B5	STA	\$B5E6			
B306-	8D EA B5	STA	\$B5EA			
B309-	AD BE B5	LDA	\$B5BE			
B30C-	8D E4 B5	STA	\$B5E4			
B30F-	8D EB B5	STA	\$B5EB			
B312-	A9 00	LDA	##00	} CLEAR SECTOR OFFSET HIGH		
B314-	8D E5 B5	STA	\$B5E5			
B317-	A0 10	LDY	##10	16 BIT MULTIPLY		
B319-	AA	TAX				
B31A-	AD E6 B5	LDA	\$B5E6	BYTE OFFS		
B31D-	4A	LSR				
B31E-	B0 03	BCS	\$B323			



B320- 8A  
 B321- 90 0E  
 B323- 18  
 B324- AD E5 B5  
 B327- 6D E8 B5  
 B32A- 8D E5 B5  
 B32D- 8A  
 B32E- 6D E9 B5  
 B331- 6A  
 B332- 6E E5 B5  
 B335- 6E E4 B5  
 B338- 6E E6 B5  
 B33B- 88  
 B33C- D0 DB  
 B33E- AD BF B5  
 B341- 8D EC B5  
 B344- 6D E6 B5  
 B347- 8D E6 B5  
 B34A- AD C0 B5  
 B34D- 8D ED B5  
 B350- 6D E4 B5  
 B353- 8D E4 B5  
 B356- A9 00  
 B358- 6D E5 B5  
 B35B- 8D E5 B5  
 B35E- 60

TXA  
 BCC \$B331  
 CLC  
 LDA \$B5E5  
 ADC \$B5E8  
 STA \$B5E5  
 TXA  
 ADC \$B5E9  
 ROR  
 ROR \$B5E5  
 ROR \$B5E4  
 ROR \$B5E6  
 DEY  
 BNE \$B319  
 LDA \$B5BF  
 STA \$B5EC  
 ADC \$B5E6  
 STA \$B5E6  
 LDA \$B5C0  
 STA \$B5ED  
 ADC \$B5E4  
 STA \$B5E4  
 LDA #\$00  
 ADC \$B5E5  
 STA \$B5E5  
 RTS

SECTOR OFFS  
 + RECD LEN

SHIFT  
 3 BYTE  
 POSITION

COMPUTE  
 RECORD #  
 \*  
 RECORD LEN  
 =  
 FILE POSITION  
 (3 BYTE OFFSET)

ADD BYTE # FROM  
 PARMS INTO  
 3 BYTE FILE  
 POSITION

B35F- A9 01 LDA #\$01  
 B361- D0 22 BNE \$B385  
 B363- A9 02 LDA #\$02  
 B365- D0 1E BNE \$B385  
 B367- A9 03 LDA #\$03  
 B369- D0 1A BNE \$B385  
 B36B- A9 04 LDA #\$04  
 B36D- D0 16 BNE \$B385  
 B36F- A9 05 LDA #\$05  
 B371- D0 12 BNE \$B385  
 B373- A9 06 LDA #\$06  
 B375- D0 0E BNE \$B385  
 \* B377- 4C ED BF JMP \$BFED  
 \* B37A- EA NOP  
 B37B- A9 0A LDA #\$0A  
 B37D- D0 06 BNE \$B385

"LANGUAGE NOT AVAILABLE" ERROR  
 "RANGE ERROR" (BAD OPCODE)  
 "RANGE ERROR" (BAD SUBCODE)  
 "WRITE PROTECTED"  
 "END OF DATA"  
 "FILE NOT FOUND"  
 GO CLOSE ALL DOS FILES, RC=9  
 "DISK FULL"  
 "FILE LOCKED"

B37F- AD C5 B5 LDA \$B5C5  
 B382- 18 CLC  
 B383- 90 01 BCC \$B386  
 B385- 38 SEC  
 B386- 08 PHP  
 B387- 8D C5 B5 STA \$B5C5  
 \* B38A- A9 00 LDA #\$00  
 \* B38C- 85 48 STA \$48  
 B38E- 20 7E AE JSR \$AE7E  
 B391- 28 PLP  
 B392- AE 9B B3 LDX \$B39B  
 B395- 9A TXS  
 B396- 60 RTS

GET FIO RETURN CODE = 0 EXIT FIO  
 NO ERROR  
 ERROR  
 SAVE INDICATOR  
 AND RC  
 CLEAR MONITOR STATUS REG (AFTER RWTS)  
 SAVE FLOWA  
 RESTORE INDICATOR  
 AND STACK PTR  
 EXIT FROM FIO

B397- 11 00  
 B399- 00  
 B39A- 00  
 B39B- F5 00  
 B39D- 01 00  
 B39F- 00  
 B3A0- 00  
 B3A1- 00

ORA (\$00),Y B397/8: CURRENT DIRECTORY T/S FIO SCRATCH SPAC  
 BRK  
 BRK  
 SBC \$00,X B39B: STACK REG SAVE B39C: DIRECTORY IN  
 ORA (\$00,X) B39D: CATALOG LINE CTR. B39E: LOCK/UNLOCK PARM  
 BRK { DIRECTORY LOOKUP FLG } { ALLOC. FLG  
 BRK etc. }  
 BRK

B3A0-B3A3: SECTOR MASK TO FREE ENTIRE TRACK (USED BY INIT) <sup>68</sup>

B3A2-	F8	SED		
B3A3-	FF	???		
B3A4-	01 0A	ORA	(#0A, X)	} 1, 10, 100
B3A6-	64	???		
B3A7-	D4	???		
B3A8-	C9 C1	CMP	##C1	T, I, A, B, S, R, A, B
B3AA-	C2	???		00 01 02 04 08 10 20 40
*B3AB-	D3	???		FILE TYPES (80 = LOCKED)
*B3AC-	D2	???		
*B3AD-	C1 C2	CMP	(#C2, X)	
B3AF-	A0 C5	LDY	##C5	
B3B1-	CD D5 CC	CMP	##CCD5	
B3B4-	CF	???		"DISK VOLUME #"
B3B5-	D6 A0	DEC	##A0, X	(BACKWARDS FOR PRINTING)
B3B7-	CB	???		
B3B8-	D3	???		
B3B9-	C9 C4	CMP	##C4	
B3BB-	02	???		
B3BC-	11 0C	ORA	(#0C), Y	B3BC/D: ↑ 1ST DIRECTORY SECTOR
B3BE-	02	???		
B3BF-	00	BRK		B3BE: DOS RELEASE #
B3C0-	00	BRK		
B3C1-	FE 00 00	INC	##0000, X	B3C1: VOL #
B3C4-	00	BRK		
B3C5-	00	BRK		
B3C6-	00	BRK		
B3C7-	00	BRK		
B3C8-	00	BRK		
B3C9-	00	BRK		
B3CA-	00	BRK		
B3CB-	00	BRK		
B3CC-	00	BRK		
B3CD-	00	BRK		
B3CE-	00	BRK		
B3CF-	00	BRK		
B3D0-	00	BRK		
B3D1-	00	BRK		
B3D2-	00	BRK		
B3D3-	00	BRK		
B3D4-	00	BRK		
B3D5-	00	BRK		
B3D6-	00	BRK		
B3D7-	00	BRK		
B3D8-	00	BRK		
B3D9-	00	BRK		
B3DA-	00	BRK		
B3DB-	00	BRK		
B3DC-	00	BRK		
B3DD-	00	BRK		
B3DE-	00	BRK		
B3DF-	00	BRK		
B3E0-	00	BRK		
B3E1-	00	BRK		
B3E2-	7A	???		B3E2: NUMBER OF ENTRIES IN EACH T/S LIST SECTOR (122)
B3E3-	00	BRK		
B3E4-	00	BRK		
B3E5-	00	BRK		
B3E6-	00	BRK		
B3E7-	00	BRK		
B3E8-	00	BRK		
B3E9-	00	BRK		
B3EA-	00	BRK		
B3EB-	18	CLC		B3EB: TRACK TO ALLOCATE NEXT

VOLUME TABLE OF CONTENTS BUFFER

(ALL OTHERS NOT USED)

B3EC-	01	00	ORA	(\$00,X)
B3EE-	00		BRK	
B3EF-	23		???	
B3F0-	0D	00 01	ORA	\$0100
B3F3-	00		BRK	
B3F4-	00	Ø	TRK	
B3F5-	00		BRK	
B3F6-	00		↓	
B3F7-	00		BRK	
B3F8-	00	1	BRK	
B3F9-	00		BRK	
B3FA-	00		BRK	
B3FB-	00		BRK	
B3FC-	00	2	BRK	
B3FD-	00		BRK	
B3FE-	00		BRK	
B3FF-	00		BRK	
B400-	00	3	BRK	
B401-	00		BRK	
B402-	00		BRK	
B403-	00		BRK	
B404-	00	4	BRK	
B405-	00		BRK	
B406-	00		BRK	
B407-	3F		???	
B408-	F8	5	SED	
B409-	00		BRK	
B40A-	00		BRK	
B40B-	00		BRK	
B40C-	00	6	BRK	
B40D-	00		BRK	
B40E-	00		BRK	
B40F-	00		BRK	
B410-	00	7	BRK	
B411-	00		BRK	
B412-	00		BRK	
B413-	00		BRK	
B414-	78	8	SEI	
B415-	00		BRK	
B416-	00		BRK	
B417-	00		BRK	
B418-	78	9	SEI	
B419-	00		BRK	
B41A-	00		BRK	
B41B-	FF		???	
B41C-	F8	10	SED	
B41D-	00		BRK	
B41E-	00		BRK	
B41F-	07		???	
B420-	F8	11	SED	
B421-	00		BRK	
B422-	00		BRK	
B423-	00		BRK	
B424-	00		BRK	
B425-	00	12	BRK	
B426-	00		BRK	
B427-	1F		???	
B428-	F8	13	SED	
B429-	00		BRK	
B42A-	00		BRK	
B42B-	00		BRK	
B42C-	38	14	SEC	
B42D-	00		BRK	

B3EC: DIRECTION OF TRK ALLOC.  
 B3EF: NO. OF TRACKS ON A DISK  
 B3F0: NO. OF SECTORS ON A TRACK  
 B3F1/2: SECTOR SIZE

FREE SECTOR BIT MAP  
 (4 BYTES PER TRACK)

↓  
 VTOC SECTOR  
 BUFFER  
 ↓

B42E-	00		BRK
B42F-	FF		???
B430-	F8	15	SED
B431-	00		BRK
B432-	00		BRK
B433-	FF		???
B434-	F8	16	SED
B435-	00		BRK
B436-	00		BRK
B437-	00		BRK
B438-	00	17	BRK
B439-	00		BRK
B43A-	00		BRK
B43B-	00		BRK
B43C-	00	18	BRK
B43D-	00		BRK
B43E-	00		BRK
B43F-	00		BRK
B440-	00	19	BRK
B441-	00		BRK
B442-	00		BRK
B443-	00		BRK
B444-	00	20	BRK
B445-	00		BRK
B446-	00		BRK
B447-	00		BRK
B448-	00	21	BRK
B449-	00		BRK
B44A-	00		BRK
B44B-	03		???
B44C-	F8	22	SED
B44D-	00		BRK
B44E-	00		BRK
B44F-	00		BRK
B450-	00	23	BRK
B451-	00		BRK
B452-	00		BRK
B453-	00		BRK
B454-	38	24	SEC
B455-	00		BRK
B456-	00		BRK
B457-	00		BRK
B458-	00	25	BRK
B459-	00		BRK
B45A-	00		BRK
B45B-	7F		???
B45C-	F8	26	SED
B45D-	00		BRK
B45E-	00		BRK
B45F-	FF		???
B460-	F8	27	SED
B461-	00		BRK
B462-	00		BRK
B463-	1F		???
B464-	F8	28	SED
B465-	00		BRK
B466-	00		BRK
B467-	FF		???
B468-	F8	29	SED
B469-	00		BRK
B46A-	00		BRK
B46B-	FF		???
B46C-	F8	30	SED

|  
VTOC SECTOR  
BUFFER  
↓

B46D-	00		BRK
B46E-	00		BRK
B46F-	FF		???
B470-	F8	31	SED
B471-	00		BRK
B472-	00		BRK
B473-	1F		???
B474-	F8	32	SED
B475-	00		BRK
B476-	00		BRK
B477-	1F		???
B478-	F8	33	SED
B479-	00		BRK
B47A-	00		BRK
B47B-	00		BRK
B47C-	38	34	SEC
B47D-	00		BRK
B47E-	00		BRK
B47F-	00		BRK
B480-	00		BRK
B481-	00		BRK
B482-	00		BRK
B483-	00		BRK
B484-	00		BRK
B485-	00		BRK
B486-	00		BRK
B487-	00		BRK
B488-	00		BRK
B489-	00		BRK
B48A-	00		BRK
B48B-	00		BRK
B48C-	00		BRK
B48D-	00		BRK
B48E-	00		BRK
B48F-	00		BRK
B490-	00		BRK
B491-	00		BRK
B492-	00		BRK
B493-	00		BRK
B494-	00		BRK
B495-	00		BRK
B496-	00		BRK
B497-	00		BRK
B498-	00		BRK
B499-	00		BRK
B49A-	00		BRK
B49B-	00		BRK
B49C-	00		BRK
B49D-	00		BRK
B49E-	00		BRK
B49F-	00		BRK
B4A0-	00		BRK
B4A1-	00		BRK
B4A2-	00		BRK
B4A3-	00		BRK
B4A4-	00		BRK
B4A5-	00		BRK
B4A6-	00		BRK
B4A7-	00		BRK
B4A8-	00		BRK
B4A9-	00		BRK
B4AA-	00		BRK
B4AB-	00		BRK

|  
VTOC SECTOR  
BUFFER  
↓



B4AC- 00 BRK  
 B4AD- 00 BRK  
 B4AE- 00 BRK  
 B4AF- 00 BRK  
 B4B0- 00 BRK  
 B4B1- 00 BRK  
 B4B2- 00 BRK  
 B4B3- 00 BRK  
 B4B4- 00 BRK  
 B4B5- 00 BRK  
 B4B6- 00 BRK  
 B4B7- 00 BRK  
 B4B8- 00 BRK  
 B4B9- 00 BRK  
 B4BA- 00 BRK

1  
 VTOC SECTOR  
 BUFFER  
 ↓

B4BB- 00 BRK  
 B4BC- 11 0B ORA  
 B4BE- 00 BRK  
 B4BF- 00 BRK  
 B4C0- 00 BRK  
 B4C1- 00 BRK  
 B4C2- 00 BRK  
 B4C3- 00 BRK  
 B4C4- 00 BRK  
 B4C5- 00 BRK

(#0B), Y } NEXT DIRECTORY SECTOR ↑ (T/S)

DIRECTORY SECTOR BUFFER  
 (CATALOG)

B4C6- 12 ???  
 B4C7- 0C ???  
 B4C8- B1 C8 STA (#C8, X)  
 B4CA- D5 CC CMP #CC  
 B4CC- CC CF AO CPY #AOCF  
 B4CF- A0 A0 LDY #A0  
 B4D1- A0 A0 LDY #A0  
 B4D3- A0 A0 LDY #A0  
 B4D5- A0 A0 LDY #A0  
 B4D7- A0 A0 LDY #A0  
 B4D9- A0 A0 LDY #A0  
 B4DB- A0 A0 LDY #A0  
 B4DD- A0 A0 LDY #A0  
 B4DF- A0 A0 LDY #A0  
 B4E1- A0 A0 LDY #A0  
 B4E3- A0 A0 LDY #A0  
 B4E5- A0 A0 LDY #A0

DIRECTORY ENTRY

+0 TRACK } T/S LIST ↑  
 +1 SECTOR }  
 +2 FILE TYPE / +3 FILENAME

B4E7- 02 ???  
 B4E8- 00 BRK

+33 } SECTORS IN USE  
 +34 }

B4E9- 13 ???  
 B4EA- 0C ???  
 B4EB- B1 C1 STA (#C1, X)  
 B4ED- D0 D0 BNE #B4BF  
 B4EF- CC C5 D3 CPY #D3C5  
 B4F2- CF ???  
 B4F3- C6 D4 DEC #D4  
 B4F5- A0 A0 LDY #A0  
 B4F7- A0 A0 LDY #A0  
 B4F9- A0 A0 LDY #A0  
 B4FB- A0 A0 LDY #A0  
 B4FD- A0 A0 LDY #A0  
 B4FF- A0 A0 LDY #A0  
 B501- A0 A0 LDY #A0  
 B503- A0 A0 LDY #A0  
 B505- A0 A0 LDY #A0  
 B507- A0 A0 LDY #A0  
 B509- A0 2B LDY #2B  
 B50B- 00 BRK

B50C-	OC		???	
B50D-	OC		???	
B50E-	81	C1	STA	(\$C1, X)
B510-	CE	C9 CD	DEC	\$CDC9
B513-	C1	CC	CMP	(\$CC, X)
B515-	D3		???	
B516-	A0	A0	LDY	##A0
B518-	A0	A0	LDY	##A0
B51A-	A0	A0	LDY	##A0
B51C-	A0	A0	LDY	##A0
B51E-	A0	A0	LDY	##A0
B520-	A0	A0	LDY	##A0
B522-	A0	A0	LDY	##A0
B524-	A0	A0	LDY	##A0
B526-	A0	A0	LDY	##A0
B528-	A0	A0	LDY	##A0
B52A-	A0	A0	LDY	##A0
B52C-	A0	12	LDY	##12
B52E-	00		BRK	
B52F-	18		CLC	
B530-	OC		???	
B531-	84	D5	STY	\$D5
B533-	D0	C4	BNE	\$B4F9
B535-	C1	D4	CMP	(\$D4, X)
B537-	C5	A0	CMP	\$A0
B539-	B3		???	
B53A-	AE	B2 A0	LDX	\$A0B2
B53D-	A0	A0	LDY	##A0
B53F-	A0	A0	LDY	##A0
B541-	A0	A0	LDY	##A0
B543-	A0	A0	LDY	##A0
B545-	A0	A0	LDY	##A0
B547-	A0	A0	LDY	##A0
B549-	A0	A0	LDY	##A0
B54B-	A0	A0	LDY	##A0
B54D-	A0	A0	LDY	##A0
B54F-	A0	09	LDY	##09
B551-	00		BRK	
B552-	19	OC 81	ORA	\$81OC, Y
B555-	C3		???	
B556-	CF		???	
B557-	D0	D9	BNE	\$B532
B559-	A0	A0	LDY	##A0
B55B-	A0	A0	LDY	##A0
B55D-	A0	A0	LDY	##A0
B55F-	A0	A0	LDY	##A0
B561-	A0	A0	LDY	##A0
B563-	A0	A0	LDY	##A0
B565-	A0	A0	LDY	##A0
B567-	A0	A0	LDY	##A0
B569-	A0	A0	LDY	##A0
B56B-	A0	A0	LDY	##A0
B56D-	A0	A0	LDY	##A0
B56F-	A0	A0	LDY	##A0
B571-	A0	A0	LDY	##A0
B573-	0E	00 09	ASL	\$0900
B576-	OC		???	
B577-	81	C3	STA	(\$C3, X)
B579-	CF		???	
B57A-	CC	CF D2	CPY	\$D2CF
B57D-	A0	C4	LDY	##C4
B57F-	C5	CD	CMP	\$CD
B581-	CF		???	

↓  
 DIRECTORY  
 SECTOR  
 BUFFER



B582-	A0 A0	LDY	##A0
B584-	A0 A0	LDY	##A0
B586-	A0 A0	LDY	##A0
B588-	A0 A0	LDY	##A0
B58A-	A0 A0	LDY	##A0
B58C-	A0 A0	LDY	##A0
B58E-	A0 A0	LDY	##A0
B590-	A0 A0	LDY	##A0
B592-	A0 A0	LDY	##A0
B594-	A0 A0	LDY	##A0
B596-	09 00	ORA	##00
B598-	1C	???	
B599-	0C	???	
B59A-	84 03	STY	#03
B59C-	08	INY	
B59D-	C1 09	CMP	(#09, X)
B59F-	CE A0 A0	DEC	#A0A0
B5A2-	A0 A0	LDY	##A0
B5A4-	A0 A0	LDY	##A0
B5A6-	A0 A0	LDY	##A0
B5A8-	A0 A0	LDY	##A0
B5AA-	A0 A0	LDY	##A0
B5AC-	A0 A0	LDY	##A0
B5AE-	A0 A0	LDY	##A0
B5B0-	A0 A0	LDY	##A0
B5B2-	A0 A0	LDY	##A0
B5B4-	A0 A0	LDY	##A0
B5B6-	A0 A0	LDY	##A0
B5B8-	A0 03	LDY	##03
B5BA-	00	BRK	

DIRECTORY  
SECTOR  
BUFFER



B5BB-	02	???	
B5BC-	00	BRK	
B5BD-	01 00	ORA	(#00, X)
B5BF-	FE 01 07	INC	#0701, X
B5C2-	81 75	STA	(#75, X)
B5C4-	AA	TAX	
B5C5-	00	BRK	
B5C6-	00	BRK	
B5C7-	00	BRK	
B5C8-	98	TYA	
B5C9-	00	BRK	
B5CA-	97	???	
B5CB-	00	BRK	
B5CC-	96 00	STX	#00, Y
B5CE-	00	BRK	
B5CF-	00	BRK	
B5D0-	00	BRK	

+0 OPCODE  
+1 SUBCODE/INIT: DOS↑

+2 → +8 VARIABLE PARAMS DEPENDENT ON CALL TYPE OPCODE

+9 RETURN CODE

FIO WORKAREA ↑  
SAVED IMAGE

T/S LIST BUFF ↑

DATA BUFF ↑

NEXT FILE BUFF ↑

**FIO PARMLIST**

**CURRENT FILE BUFFER PTRS**

B5D1-	12	???	(T)
B5D2-	0C	???	(S)
B5D3-	12	???	(T)
B5D4-	0C	???	(S)
B5D5-	00	BRK	
B5D6-	12	???	
B5D7-	0B	???	
B5D8-	01 00	ORA	(#00, X)
B5DA-	7A	???	
B5DB-	00	BRK	
B5DC-	00	BRK	
B5DD-	00	BRK	
B5DE-	7A	???	
B5DF-	00	BRK	
B5E0-	00	BRK	
B5E1-	00	BRK	

B5D1: } 1st T/S LIST ↑

B5D2: }

B5D3: } CURR. T/S LIST ↑

B5D4: }

B5D5: } FLAGS - 80 = WRITE T/S LIST, 40 = WRITE DATA, 20 = WRITE VTOC, 02 = WRITE

B5D6: } CURR. DATA SECTOR

B5D7: }

B5D8: } DIRECT. SECTOR

B5D9: } DIRECTORY OFFSET

B5DA: } NUMBER OF SECTORS DESCRIBED BY ONE T/S LIST

B5DB: }

B5DC: } RELATIVE SECTOR # OF FIRST SECTOR IN LIST

B5DD: }

B5DE: } RELATIVE SECTOR # + 1 OF LAST SECTOR IN LIST

B5DF: }

B5E0: } SECTOR LAST READ

B5E1: }

**FIO WORK AREA**

B5E2-	00	BRK
B5E3-	01 01	ORA (\$01, X)
B5E5-	00	BRK
B5E6-	00	BRK
B5E7-	00	BRK
B5E8-	01 00	ORA (\$00, X)
B5EA-	00	BRK
B5EB-	00	BRK
B5EC-	00	BRK
B5ED-	00	BRK
B5EE-	02	???
B5EF-	00	BRK
B5F0-	00	BRK
B5F1-	00	BRK
B5F2-	00	BRK
B5F3-	00	BRK
B5F4-	00	BRK
B5F5-	00	BRK
B5F6-	81 70	STA (\$70, X)
B5F8-	01 FF	ORA (\$FF, X)
B5FA-	11 00	ORA (\$00), Y
B5FC-	00	BRK
B5FD-	00	BRK
B5FE-	00	BRK
B5FF-	00	BRK

FILE POSITION

SECTOR ALLOCATION AREA

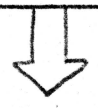
B5E2: } SECTOR LENGTH IN BYTES  
 B5E3: }  
 B5E4/5: FILE POSITION (SECTOR OFFSET)  
 B5E6: BYTE OFFSET IN CURRENT SECTOR  
 B5E7: NOT USED  
 B5E8/9: RECORD LENGTH (FROM OPEN)  
 B5EA: } RECORD NUMBER  
 B5EB: }  
 B5EC: } BYTE OFFSET INTO RECORD  
 B5ED: }  
 B5EE: } NO. OF SECTORS IN FILE  
 B5EF: }  
 B5F0: NEXT SECTOR TO ALLOCATE  
 B5F1: TRACK BEING ALLOCATED  
 B5F2: } BIT MAP OF SECTORS FREE  
 B5F3: } ON CURRENT TRACK, ROTATED  
 B5F4: } TO NEXT SECTOR TO ALLOCATE  
 B5F5: }  
 B5F6: FILE TYPE B5F7: SLOT # \*16  
 B5F8: DRIVE # B5F9: VOL # (COMPL.)  
 B5FA: TRACK #  
 B5FB - B5FD: UNUSED

B600-	F0 4A	BEQ	\$B64C
B602-	99 FF FF	STA	(\$FFF, Y)
B605-	03	???	
B606-	3C	???	
B607-	AD FF FF	LDA	(\$FFFF)
B60A-	FF	???	
B60B-	26 B3	ROL	(\$B3)
B60D-	FF	???	
B60E-	FF	???	
B60F-	4D 4A 10	EOR	(\$104A)
B612-	FF	???	
B613-	FF	???	
B614-	3D 4A CA	AND	(\$CA4A, X)
B617-	FF	???	
B618-	FF	???	
B619-	A5 4A	LDA	(\$4A)
B61B-	C8	INY	
B61C-	FF	???	
B61D-	FF	???	
B61E-	03	???	
B61F-	4A	LSR	
B620-	40	RTI	
B621-	FF	???	
B622-	FF	???	
B623-	46 08	LSR	(\$08)
B625-	91 FF	STA	(\$FF), Y
B627-	FF	???	
B628-	20 33 09	JSR	(\$0933)
B62B-	FF	???	
B62C-	FF	???	
B62D-	03	???	
B62E-	BD CC FF	LDA	(\$FFCC, X)
B631-	FF	???	
B632-	43	???	
B633-	C8	INY	
B634-	1D FF FF	ORA	(\$FFF, X)
B637-	20 40 07	JSR	(\$0740)
B63A-	FF	???	

PAGE 3  
RAM BOOT  
LOADER  
IMAGE

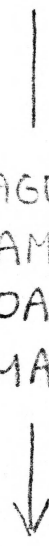
(IN NON STANDARD  
FORMAT.)

STAGE ONE  
BOOT FROM  
THIS POINT



B63B-	FF	???	
B63C-	3E 91 29	ROL	\$2991, X
B63F-	FF	???	
B640-	FF	???	
B641-	85 09	STA	\$09
B643-	3C	???	
B644-	FF	???	
B645-	FF	???	
B646-	5D 00 A5	EOR	\$A500, X
B649-	FF	???	
B64A-	FF	???	
B64B-	A9 1D	LDA	##1D
B64D-	C8	INY	
B64E-	FF	???	
B64F-	FF	???	
B650-	3F	???	
B651-	4A	LSR	
B652-	40	RTI	
B653-	FF	???	
B654-	FF	???	
B655-	85 2A	STA	\$2A
B657-	91 FF	STA	(\$FF), Y
B659-	FF	???	
B65A-	C0 85	CPY	##85
B65C-	09 FF	ORA	##FF
B65E-	FF	???	
B65F-	09 4A	ORA	##4A
B661-	99 FF FF	STA	\$FFFF, Y
B664-	4A	LSR	
B665-	3C	???	
B666-	1D FF FF	ORA	\$FFFF, X
B669-	4A	LSR	
B66A-	85 07	STA	\$07
B66C-	FF	???	
B66D-	FF	???	
B66E-	4A	LSR	
B66F-	4A	LSR	
B670-	29 FF	AND	##FF (DOS 3.3 PATCH)
B672-	FF	???	
B673-	4A	LSR	
B674-	4A	LSR	
B675-	2A	ROL	
B676-	FF	???	
B677-	FF	???	
B678-	8A	TXA	
B679-	4A	LSR	
B67A-	A5 FF	LDA	\$FF
B67C-	FF	???	
B67D-	40	RTI	
B67E-	08	PHP	
B67F-	C8	INY	
B680-	FF	???	
B681-	FF	???	
B682-	84 00	STY	\$00
B684-	40	RTI	
B685-	FF	???	
B686-	FF	???	(DOS 3.3 PATCH)
B687-	41 BD	EOR	(\$BD, X)
B689-	91 FF	STA	(\$FF), Y
B68B-	FF	???	
B68C-	85 00	STA	\$00
B68E-	09 FF	ORA	##FF
B690-	FF	???	

(DOS 3.3 PATCH) PAGE 3  
RAM BOOT  
LOADER  
IMAGE



(DOS 3.3 PATCH)

(DOS 3.3 PATCH)



B691-	03	???	
B692-	A0 66	LDY	##66
B694-	FF	???	
B695-	FF	???	
B696-	CC 32 1D	CPY	\$1D32
B699-	FF	???	
B69A-	FF	???	
B69B-	AD A2 2A	LDA	\$2AA2
B69E-	FF	???	
B69F-	FF	???	
B6A0-	27	???	
B6A1-	00	BRK	
B6A2-	26 FF	ROL	##FF
B6A4-	FF	???	
B6A5-	85 3E	STA	\$3E
B6A7-	4A	LSR	
B6A8-	FF	???	
B6A9-	FF	???	
B6AA-	09 6C	ORA	##6C
B6AC-	3C	???	
B6AD-	FF	???	
B6AE-	FF	???	
B6AF-	A9 3F	LDA	##3F
B6B1-	26 FF	ROL	##FF
B6B3-	FF	???	
B6B4-	2B	???	
B6B5-	E6 4A	INC	\$4A
B6B7-	FF	???	
B6B8-	FF	???	
B6B9-	A6 3F	LDX	\$3F
B6BB-	4A	LSR	
B6BC-	FF	???	
B6BD-	FF	???	
B6BE-	F4	???	
B6BF-	85 4A	STA	\$4A
B6C1-	FF	???	
B6C2-	FF	???	
B6C3-	D0 03	BNE	\$B6C3
B6C5-	4A	LSR	
B6C6-	60	RTS	
B6C7-	FF	???	
B6C8-	C8	INY	
B6C9-	CC 08 2B	CPY	\$2B08
B6CC-	FF	???	
B6CD-	08	PHP	
B6CE-	AD 66 A6	LDA	\$A666
B6D1-	FF	???	
B6D2-	00	BRK	
B6D3-	3E BD 40	ROL	\$40BD, X
B6D6-	FF	???	
B6D7-	99 85 C8	STA	\$C885, Y
B6DA-	91 FF	STA	(\$FF), Y
B6DC-	0A	ASL	
B6DD-	ED 40 09	SBC	\$0940
B6E0-	FF	???	
B6E1-	0A	ASL	
B6E2-	D0 91	BNE	\$B675
B6E4-	FF	???	
B6E5-	FF	???	
B6E6-	0A	ASL	
B6E7-	3D 09 0D	AND	\$0D09, X
B6EA-	FF	???	
B6EB-	08	PHP	

DOS 3.3 PATCH

PAGE 3  
RAM BOOT  
LOADER  
IMAGE



B6EC-	E6 33	INC	#33	
B6EE-	4A	LSR		
B6EF-	FF	???		
B6F0-	00	BRK		
B6F1-	41 1D	EOR	(#1D, X)	
B6F3-	4A	LSR		
B6F4-	FF	???		
B6F5-	B9 E6 2A	LDA	#2AE6, Y	
B6F8-	4A	LSR		
B6F9-	FF	???		
B6FA-	99 06 26	STA	#2606, Y	
B6FD-	08	PHP		
B6FE-	36 48	ROL	#48, X	←
B700-	8E E9 B7	STX	#B7E9	} SET SLOT #'S IN RWTS PARMS
B703-	8E F7 B7	STX	#B7F7	
B706-	A9 01	LDA	#01	} PREVIOUS DRIVE MUST BE 1
B708-	8D F8 B7	STA	#B7F8	
B70B-	8D EA B7	STA	#B7EA	THIS DRIVE TOO
B70E-	AD E0 B7	LDA	#B7E0	} # PAGES IN SECOND LOAD
B711-	8D E1 B7	STA	#B7E1	
B714-	A9 00	LDA	#00	} TRACK 0
B716-	8D EC B7	STA	#B7EC	
B719-	AD E2 B7	LDA	#B7E2	} SECTOR 10 (DOS 3.3 PATCH)
B71C-	8D ED B7	STA	#B7ED	
B71F-	AD E3 B7	LDA	#B7E3	} LOAD ADDRESS OF DOS
B722-	8D F1 B7	STA	#B7F1	
B725-	A9 01	LDA	#01	} READING (OPCODE)
B727-	8D F4 B7	STA	#B7F4	
B72A-	8A	TXA		} GET TRUE SLOT #
B72B-	4A	LSR		
B72C-	4A	LSR		
B72D-	4A	LSR		
B72E-	4A	LSR		
B72F-	AA	TAX		
B730-	A9 00	LDA	#00	} SET REMEMBERED CURR. TRKS
B732-	9D F8 04	STA	#04F8, X	
B735-	9D 78 04	STA	#0478, X	
B738-	20 93 B7	JSR	#B793	GO READ DOS
B73B-	A2 FF	LDX	#FF	} BRAND NEW STACK
B73D-	9A	TXS		
B73E-	8E EB B7	STX	#B7EB	ANY VOL WILL DO
B741-	20 93 FE	JSR	#FE93	SET VIDEO (DOS 3.3 PATCH)
B744-	20 89 FE	JSR	#FE89	SET KEYBOARD
B747-	4C 84 9D	JMP	#9D84	GO TO DOS COLDSTART (1BD3 @ BOOT-RELOC)
B74A-	AD F1 B7	LDA	#B7F1	} SET PAGE OF DOS LOAD POINT
B74D-	8D E3 B7	STA	#B7E3	
B750-	38	SEC		} COMPUTE NO. OF PAGES IN DOS
B751-	AD E7 B7	LDA	#B7E7	
B754-	ED E3 B7	SBC	#B7E3	} SECOND STAGE BOOT (25)
B757-	8D E0 B7	STA	#B7E0	
B75A-	A9 00	LDA	#00	} TRACK 0
B75C-	8D EC B7	STA	#B7EC	
B75F-	8D ED B7	STA	#B7ED	} SECTOR 0
B762-	8D F0 B7	STA	#B7F0	
B765-	AD E7 B7	LDA	#B7E7	} WRITE FROM 1ST STAGE BOOT (\$B600)
B768-	8D F1 B7	STA	#B7F1	
B76B-	8D FE B6	STA	#B6FE	} TELL RAM BOOTER WHERE IT GOES
B76E-	A9 0A	LDA	#0A	
B770-	8D E1 B7	STA	#B7E1	} 10 SECTORS TO WRITE
B773-	8D E2 B7	STA	#B7E2	
B776-	A9 48	LDA	#48	} SET # SECTORS IN LOAD ONE
B778-	8D FF B6	STA	#B6FF	
B77B-	A9 02	LDA	#02	SAME IN 3 PAGE

#3CC: FIRST PAGE OF FIRST STAGE DOS BOOT LOAD (\$B6)

END OF \$300 RAM BOOTSTRAP

DOS 2ND STAGE BOOT LOADER

DOS 3.3 PATCH

PUT DOS 0 TRKS 0-2



B77D- 8D F4 B7 STA #B7F4 } SET RWTS WRITE OPCODE  
 B780- 20 93 B7 JSR #B793 WRITE FIRST STAGE  
 B783- AD E3 B7 LDA #B7E3 } POINT TO DOS SECOND STAGE BOOT LOAD  
 B786- 8D F1 B7 STA #B7F1 }  
 B789- AD E0 B7 LDA #B7E0 } NUMBER OF SECTORS TO WRITE  
 B78C- 8D E1 B7 STA #B7E1 }  
 B78F- 20 93 B7 JSR #B793 WRITE SECOND STAGE  
 B792- 60 RTS

B793- AD E5 B7 LDA #B7E5 } POINT TO PARMLIST  
 B796- AC E4 B7 LDY #B7E4 }  
 B799- 20 B5 B7 JSR #B7B5 AND CALL RWTS  
 B79C- AC ED B7 LDY #B7ED } NEXT SECTOR  
 B79F- C8 INY  
 B7A0- C0 0D CPY ##0D END OF TRACK?  
 B7A2- D0 05 BNE #B7A9 NO  
 B7A4- A0 00 LDY ##00 YES, SECTOR 0  
 B7A6- EE EC B7 INC #B7EC NEXT TRACK  
 B7A9- 8C ED B7 STY #B7ED SET SECTOR  
 B7AC- EE F1 B7 INC #B7F1 NEXT DATA PAGE  
 B7AF- CE E1 B7 DEC #B7E1 } DEC & CHECK # OF SECTORS LEFT  
 B7B2- D0 DF BNE #B793 }  
 B7B4- 60 RTS

READ or WRITE  
A GROUP  
OF PAGES

\* B7B5- 08 PHP  
 \* B7B6- 78 SEI  
 \* B7B7- 20 00 BD JSR #BD00  
 \* B7BA- B0 03 BCS #B7BF  
 \* B7BC- 28 PLS  
 \* B7BD- 18 CLC  
 \* B7BE- 60 RTS  
 \* B7BF- 28 PLP  
 \* B7C0- 38 SEC  
 \* B7D1- 60 RTS

SAVE S REG  
DISABLE INTERRUPTS  
AND CALL RWTS  
RESTORE S REG (REENABLING INTERRUPTS  
AND PASS BACK CARRY AS ERROR  
INDICATOR.

DISABLE  
AND CALL  
RWTS  
EXTERNAL RWTS

\* B7C2- AD BC B5 LDA #B5BC  
 \* B7C5- 8D F1 B7 STA #B7F1  
 \* B7C8- A9 00 LDA ##00  
 \* B7CA- 8D F0 B7 STA #B7F0  
 \* B7CD- AD F9 B5 LDA #B5F9  
 \* B7D0- 49 FF EOR ##FF  
 \* B7D2- 8D EB B7 STA #B7EB  
 \* B7D5- 60 RTS

SET RWTS BUFFER  
TO DOS LOAD POINT  
SET PROPER RWTS VOL #

SET RWTS  
PARMS FOR  
WRITING  
DOS

\* B7D6- A9 00 LDA ##00  
 \* B7D8- A8 TAY  
 \* B7D9- 91 42 STA (#42),Y  
 \* B7DB- C8 INY  
 \* B7DC- D0 FB BNE #B7D9  
 \* B7DE- 60 RTS

ZERO 256  
BYTES

ZERO CURRENT  
BUFFER

B7DF- 00 BRK  
 B7E0- 1B ???  
 B7E1- 00 BRK  
 B7E2- 0A ASL  
 B7E3- 1B ???  
 B7E4- E8 INX  
 B7E5- B7 ???  
 B7E6- 00 BRK  
 B7E7- B6 01 LDX #01,Y  
 B7E9- 70 01 BVS #B7EC  
 B7EB- FF ???  
 B7EC- 12 ???  
 B7ED- 0B ???  
 B7EE- FB ???  
 B7EF- B7 ???  
 B7F0- 00 BRK  
 B7F1- 96 00 STX #00,Y

B7E0: NUMBER OF PAGES IN SECOND DOS LOAD  
 B7E1: NUMBER OF SECTORS TO WRITE  
 B7E2: NUMBER OF PAGES IN FIRST DOS LOAD  
 B7E3: INIT DOS PAGE COUNTER  
 B7E4: } ↑ RWTS PARMS  
 5: }  
 B7E6: ↑ FIRST STAGE BOOT LOCATION  
 B7E8: JOB TYPE (01)  
 B7E9: S\*16 B7EA: DRIVE  
 B7EB: VOL # EXPECTED  
 B7EC: TRACK TO USE  
 B7ED: SECTOR TO USE  
 B7EE: } ↑ DEVICE CHARACTERISTICS TABLE  
 F: }  
 B7F0: } ↑ DATA BUFFER (256 BYTES)  
 1: }

\* 25-SLA  
27-MAS

RWTS PARMLIST

B7F3-	01 01	ORA	(#01, X)	B7F2/3: SECTOR SIZE	B7F4: COMMAND CODE	
B7F5-	00	BRK		B7F5: RETURN CODE		
B7F6-	FE 70 01	INC	#0170, X	B7F6: VOL FOUND	B7F7: PREVIOUS SLOT ? DRIVE	
B7F9-	00	BRK				
B7FA-	00	BRK				
B7FB-	00	BRK		B7FB: DEVICE TYPE		DEVICE CHARAT: TABLE
B7FC-	01 EF	ORA	(#EF, X)	B7FC: PHASES PER TRK		
B7FE-	D8	CLD		B7FD/E: MOTOR ON TIME COUNT		
B7FF-	00	BRK				
B800-	A2 32	LDX	##32	OUTPUT INDEX		CONVERT DATA FOR WRITE 8 BIT BYTES TO 5 BIT NIBBLES
B802-	A0 00	LDY	##00	INPUT INDEX		
B804-	B1 3E	LDA	(#3E), Y	GET DATA BYTE		
B806-	85 26	STA	#26	A1A2A3A4 A5A6A7A8		
B808-	4A	LSR				
B809-	4A	LSR				
B80A-	4A	LSR				
B80B-	9D 00 BB	STA	##BB00, X	000A, A2A3A4A5 #1		
B80E-	C8	INY		NEXT DATA BYTE		
B80F-	B1 3E	LDA	(#3E), Y	B1B2B3B4 B5B6B7B8		
B811-	85 27	STA	#27			
B813-	4A	LSR				
B814-	4A	LSR				
B815-	4A	LSR				
B816-	9D 33 BB	STA	##BB33, X	000B, B2B3B4B5 #2		
B819-	C8	INY		NEXT DATA BYTE		
B81A-	B1 3E	LDA	(#3E), Y	C1C2C3C4 C5C6C7C8		
B81C-	85 2A	STA	#2A			
B81E-	4A	LSR				
B81F-	4A	LSR				
B820-	4A	LSR				
B821-	9D 66 BB	STA	##BB66, X	000C, C2C3C4C5 #3		
B824-	C8	INY		NEXT DATA BYTE		
B825-	B1 3E	LDA	(#3E), Y	D1D2D3D4 D5D6D7D8		
B827-	4A	LSR		0D1D2D3 D4D5D6D7	CARRY = D8	
B828-	26 2A	ROL	#2A	C2C3C4C5 C6C7C8D8		
B82A-	4A	LSR		00D1D2 D3D4D5D6	C = D7	
B82B-	26 27	ROL	#27	B2B3B4B5 B6B7B8D7		
B82D-	4A	LSR		000D1 D2D3D4D5	C = D6	
B82E-	26 26	ROL	#26	A2A3A4A5 A6A7A8D6		
B830-	9D 99 BB	STA	##BB99, X	000D1, D2D3D4D5 #4		
B833-	C8	INY		NEXT DATA BYTE		
B834-	B1 3E	LDA	(#3E), Y	E2E3E4 E5E6E7E8		
B836-	4A	LSR		0E, E2E3 E4E5E6E7	C = E8	
B837-	26 2A	ROL	#2A	C3C4C5C6 C7C8D8E8		
B839-	4A	LSR		00E1E2 E3E4E5E6	C = E7	
B83A-	26 27	ROL	#27	B3B4B5B6 B7B8D7E7		
B83C-	4A	LSR		000E1 E2E3E4E5	C = E6	
B83D-	9D CC BB	STA	##BBCC, X	000E1, E2E3E4E5 #5		
B840-	A5 26	LDA	#26	A2A3A4A5 A6A7A8D6		
B842-	2A	ROL		A3A4A5A6 A7A8D6E6		
B843-	29 1F	AND	##1F			
B845-	9D 00 BC	STA	##BC00, X	000A6, A7A8D6E6		
B848-	A5 27	LDA	#27	B3B4B5B6 B7B8D7E7		
B84A-	29 1F	AND	##1F			
B84C-	9D 33 BC	STA	##BC33, X	000B6, B7B8D7E7		
B84F-	A5 2A	LDA	#2A	C3C4C5C6 C7C8D8E8		
B851-	29 1F	AND	##1F			
B853-	9D 66 BC	STA	##BC66, X	000C6, C7C8D8E8		
B856-	C8	INY				
B857-	CA	DEX				
B858-	10 AA	BPL	##B804	DO 51 GROUPS OF 5 BYTES (255)		
B85A-	B1 3E	LDA	(#3E), Y	GET LAST DATA BYTE		
B85C-	AA	TAX				

BUILD  
PRIMARY  
BUFFER

BUILD  
SECONDARY  
BUFFER

B85D-	29 07	AND	##07		
B85F-	8D 99 BC	STA	\$BC99	STORE LAST 3 BITS IN SECONDARY BUFF	
B862-	8A	TXA			
B863-	4A	LSR			
B864-	4A	LSR			
B865-	4A	LSR			
B866-	8D FF BB	STA	\$BBFF	AND FIRST 5 BITS IN PRIMARY BUFF	
B869-	60	RTS		DONE	
B86A-	38	SEC		ASSUME ERROR	WRITE SECTOR DATA
B86B-	BD 8D C0	LDA	\$C08D, X	SENSE WRITE PROTECT	
B86E-	BD 8E C0	LDA	\$C08E, X	SET READ MODE	
B871-	30 7C	BMI	\$B8EF	WRITE PROTECTED!	
B873-	84 27	STX	\$27	SAVE SLOT/DRIVE	DATA IS IN 58
B875-	8E 78 06	STX	\$0678		FORM
B878-	AD 00 BC	LDA	\$BC00	INITIALIZE CHECKSUM	
B87B-	85 26	STA	\$26		
B87D-	A9 FF	LDA	##FF	WRITING SYNC'S	
B87F-	9D 8F C0	STA	\$C08F, X	SET WRITE MODE	
B882-	1D 8C C0	ORA	\$C08C, X	START WRITING	
B885-	48	PHA			
B886-	68	PLA		(TIMING)	
B887-	EA	NOP			
B888-	A0 0A	LDY	##0A	WRITE 10 MORE SYNC'S BEFORE DATA	
B88A-	05 26	ORA	\$26	(TIMING)	
B88C-	20 F4 B8	JSR	\$B8F4	WRITE NIBBLE	
B88F-	88	DEY		DO 10	
B890-	D0 F8	BNE	\$B88A		
B892-	A9 D5	LDA	##D5		
B894-	20 F3 B8	JSR	\$B8F3		
B897-	A9 AA	LDA	##AA		
B899-	20 F3 B8	JSR	\$B8F3		
B89C-	A9 AD	LDA	##AD		
B89E-	20 F3 B8	JSR	\$B8F3		
B8A1-	98	TYA			
B8A2-	A0 9A	LDY	##9A	START WITH 0 Acc	
B8A4-	D0 03	BNE	\$B8A9	DO SECONDARY 154 BYTE BUFFER FIRST	
B8A6-	B9 00 BC	LDA	\$BC00, Y	GET DATA BYTE	
B8A9-	59 FF BB	EOR	\$BBFF, Y4	XOR WITH PREVIOUS	
B8AC-	AA	TAX			
B8AD-	BD 9A BC	LDA	\$BC9A, X	TRANSLATE TO DISK CODE	WRITE 154 BYTE SECONDARY BUFFER
B8B0-	A6 27	LDX	\$27	GET SLOT #	
B8B2-	9D 8D C0	STA	\$C08D, X	WRITE IT	
B8B5-	BD 8C C0	LDA	\$C08C, X	DO 153	
B8B8-	88	DEY			
B8B9-	D0 EB	BNE	\$B8A6		
B8BB-	A5 26	LDA	\$26	START WITH FIRST BYTE OF SECONDARY	
B8BD-	EA	NOP			
B8BE-	59 00 BB	EOR	\$BB00, Y	MERGE WITH NEXT DATA	
B8C1-	AA	TAX			
B8C2-	BD 9A BC	LDA	\$BC9A, X	TRANSLATE	WRITE 256 BYTE PRIMARY BUFFER
B8C5-	AE 78 06	LDX	\$0678		
B8C8-	9D 8D C0	STA	\$C08D, X	AND WRITE IT	
B8CB-	BD 8C C0	LDA	\$C08C, X	PREVIOUS DATA	
B8CE-	B9 00 BB	LDA	\$BB00, Y	DO 256	
B8D1-	C8	INY			
B8D2-	D0 EA	BNE	\$B8BE		
B8D4-	AA	TAX			
B8D5-	BD 9A BC	LDA	\$BC9A, X	TRANSLATE CHECKSUM	
B8D8-	A6 27	LDX	\$27		
B8DA-	20 F6 B8	JSR	\$B8F6	WRITE IT	
B8DD-	A9 DE	LDA	##DE		
B8DF-	20 F3 B8	JSR	\$B8F3		
B8E2-	A9 AA	LDA	##AA		

WRITE SECTOR DATA

DATA IS IN 58 FORM

(D5)  
(AA)  
(AD)

WRITE SECTOR DATA PROLOGUE

WRITE 154 BYTE SECONDARY BUFFER

WRITE 256 BYTE PRIMARY BUFFER

(DE)  
(AA)

WRITE SECTOR DATA EPILOGUE



```

B8E4- 20 F3 B8 JSR $B8F3
B8E7- A9 EB LDA ##EB
B8E9- 20 F3 B8 JSR $B8F3
B8EC- BD 8E C0 LDA $C08E, X
B8EF- BD 8C C0 LDA $C08C, X
B8F2- 60 RTS

```

(EB)

SET READ MODE AGAIN

```

B8F3- 18 CLC
B8F4- 48 PHA
B8F5- 68 PLA
B8F6- 9D 8D C0 STA $C08D, X
B8F9- 1D 8C C0 ORA $C08C, X
B8FC- 60 RTS

```

(TIMING)

WRITE DATA IN LATCH  
CLEAR LATCH

WRITE ONE  
5 BIT NIBBL

```

B8FD- A0 20 LDY ##20
B8FF- 88 DEY
B900- F0 61 BEQ $B963
B902- BD 8C C0 LDA $C08C, X
B905- 10 FB BPL $B902
B907- 49 D5 EOR ##D5
B909- D0 F4 BNE $B8FF
B90B- EA NOP

```

GIVE IT 32 TRIES  
TO FIND (D5)  
IF NOT FOUND, SECTOR EMPTY

READ SECTOR  
DATA

READS 5 BIT  
NIBBLES

```

B90C- BD 8C C0 LDA $C08C, X
B90F- 10 FB BPL $B90C
B911- C9 AA CMP ##AA
B913- D0 F2 BNE $B907
B915- A0 9A LDY ##9A
B917- BD 8C C0 LDA $C08C, X
B91A- 10 FB BPL $B917
B91C- C9 AD CMP ##AD
B91E- D0 E7 BNE $B907
B920- A9 00 LDA ##00

```

GET DATA BYTE  
NOT YET  
(D5)?  
No

NEXT  
(AA)?  
BYTE CTR FOR LATER

FIND DATA  
PROLOGUE

```

B922- 88 DEY
B923- 84 26 STY $26
B925- BC 8C C0 LDY $C08C, X
B928- 10 FB BPL $B925
B92A- 59 00 BA EOR $BA00, Y
B92D- A4 26 LDY $26
B92F- 99 00 BC STA $BC00, Y
B932- D0 EE BNE $B922

```

START CHECKSUM

GET NEXT

READ SECOND.  
BUFFER

TRANSLATE TO 5 BIT XOR WITH PREVIOUS  
PUT IN BUFFER  
DO 154

```

B934- 84 26 STY $26
B936- BC 8C C0 LDY $C08C, X
B939- 10 FB BPL $B936
B93B- 59 00 BA EOR $BA00, Y
B93E- A4 26 LDY $26
B940- 99 00 BB STA $BB00, Y
B943- C8 INY
B944- D0 EE BNE $B934

```

NEXT  
XLATE + XOR

READ PRIMARY  
BUFFER

STORE  
DO 256

```

B946- BC 8C C0 LDY $C08C, X
B949- 10 FB BPL $B946
B94B- D9 00 BA CMP $BA00, Y
B94E- D0 13 BNE $B963
B950- BD 8C C0 LDA $C08C, X
B953- 10 FB BPL $B950
B955- C9 DE CMP ##DE
B957- D0 0A BNE $B963
B959- EA NOP

```

GET CHECKSUM

IS IT OK?  
No, ERROR

(DE)

VERIFY PART  
OF EPILOGUE

```

B95A- BD 8C C0 LDA $C08C, X
B95D- 10 FB BPL $B95A
B95F- C9 AA CMP ##AA
B961- F0 5C BEQ $B9BF
B963- 38 SEC
B964- 60 RTS

```

NO ERRORS, EXIT  
ERROR EXIT

```

B965- A0 F8 LDY ##F8
B967- 84 26 STY $26

```

GIVE IT 8\*256  
TRIES TO FIND A  
SECTOR

READ SECTOR  
ADDRESS HEADER

```

B969- C8 INY
B96A- D0 04 BNE $B970
B96C- E6 26 INC $26
B96E- F0 F3 BEQ $B963
B970- BD 8C C0 LDA $C08C, X
B973- 10 FB BPL $B970
B975- C9 D5 CMP ##D5
B977- D0 F0 BNE $B969
B979- EA NOP
B97A- BD 8C C0 LDA $C08C, X
B97D- 10 FB BPL $B97A
B97F- C9 AA CMP ##AA
B981- D0 F2 BNE $B975
B983- A0 03 LDY ##03
B985- BD 8C C0 LDA $C08C, X
B988- 10 FB BPL $B985
B98A- C9 B5 CMP ##B5
B98C- D0 E7 BNE $B975
B98E- A9 00 LDA ##00
B990- 85 27 STA $27
B992- BD 8C C0 LDA $C08C, X
B995- 10 FB BPL $B992
B997- 2A ROL
B998- 85 26 STA $26
B99A- BD 8C C0 LDA $C08C, X
B99D- 10 FB BPL $B99A
B99F- 25 26 AND $26
B9A1- 99 2C 00 STA $002C, Y
B9A4- 45 27 EOR $27
B9A6- 88 DEY
B9A7- 10 E7 BPL $B990
B9A9- A8 TAY
B9AA- D0 B7 BNE $B963
B9AC- BD 8C C0 LDA $C08C, X
B9AF- 10 FB BPL $B9AC
B9B1- C9 DE CMP ##DE
B9B3- D0 AE BNE $B963
B9B5- EA NOP
B9B6- BD 8C C0 LDA $C08C, X
B9B9- 10 FB BPL $B9B6
B9BB- C9 AA CMP ##AA
B9BD- D0 A4 BNE $B963
B9BF- 18 CLC
B9C0- 60 RTS

```

COUNT TRIES

CYCLE 8 TIMES THEN GIVE UP

LOOK FOR...

(D5)

(AA)

INIT Y=3

(B5)

SEARCH FOR A SECTOR ADDRESS HEADER PROLOGUE

GET 1ST NIBBLE SHIFT TO EVEN BITS

GET 2ND NIBBLE

COMBINE THEM AND SAVE KEEP CHECKSUM

DO 4 BYTES

CHECKSUM OK?

READ SECTOR ADDRESS INFORMATION

2C = CHECKSUM  
2D = SECTOR #  
2E = TRACK #  
2F = VOL ID #

(DE)

VERIFY PART OF EPILOGUE

(AA)

NO ERROR EXIT

```

B9C1- A2 32 LDX ##32
B9C3- A0 00 LDY ##00
B9C5- BD 00 BC LDA $BC00, X
B9C8- 4A LSR
B9C9- 4A LSR
B9CA- 4A LSR
B9CB- 85 27 STA $27
B9CD- 4A LSR
B9CE- 85 26 STA $26
B9D0- 4A LSR
B9D1- 1D 00 BB ORA $BB00, X
B9D4- 91 3E STA ($3E), Y
B9D6- C8 INY
B9D7- BD 33 BC LDA $BC33, X
B9DA- 4A LSR
B9DB- 4A LSR
B9DC- 4A LSR
B9DD- 4A LSR
B9DE- 26 27 ROL $27

```

INPUT INDEX OUTPUT INDEX  
A6 A7 A8 D6 E6 000

000A6 A7 A8 D6 E6

0000 A6 A7 A8 D6  
0000 0A6 A7 A8

A1 A2 A3 A4 A5 A6 A7 A8 #1  
STORE IN BUFFER

B6 B7 B8 D7 E7 000

0000 B6 B7 B8 D7 C = E7  
00 A6 A7 A8 D6 E6 E7

CONVERT DATA READ  
5 BIT NIBBLES  
8 BIT BYTES

B9E0- 4A  
 B9E1- 26 26  
 B9E3- 1D 33 BB  
 B9E6- 91 3E  
 B9E8- C8  
 B9E9- BD 66 BC  
 B9EC- 4A  
 B9ED- 4A  
 B9EE- 4A  
 B9EF- 4A  
 B9F0- 26 27  
 B9F2- 4A  
 B9F3- 26 26  
 B9F5- 1D 66 BB  
 B9F8- 91 3E  
 B9FA- C8  
 B9FB- A5 26  
 B9FD- 29 07  
 B9FF- 1D 99 BB  
 BA02- 91 3E  
 BA04- C8  
 BA05- A5 27  
 BA07- 29 07  
 BA09- 1D CC BB  
 BA0C- 91 3E  
 BA0E- C8  
 BA0F- CA  
 BA10- 10 B3  
 BA12- AD 99 BC  
 BA15- 4A  
 BA16- 4A  
 BA17- 4A  
 BA18- 0D FF BB  
 BA1B- 91 3E  
 BA1D- 60  
 BA1E- 85 2A  
 BA20- CD 78 04  
 BA23- F0 59  
 BA25- 86 2B  
 BA27- A9 00  
 BA29- 85 26  
 BA2B- AD 78 04  
 BA2E- 85 27  
 BA30- 38  
 BA31- E5 2A  
 BA33- F0 42  
 BA35- B0 07  
 BA37- 49 FF  
 BA39- EE 78 04  
 BA3C- 90 05  
 BA3E- 69 FE  
 BA40- CE 78 04  
 BA43- C5 26  
 BA45- 90 02  
 BA47- A5 26  
 BA49- C9 0C  
 BA4B- 90 02  
 BA4D- A9 0B  
 BA4F- A8  
 BA50- AD 78 04  
 BA53- 29 03  
 BA55- 0A  
 BA56- 05 2B

LSR  
 ROL  
 ORA  
 STA  
 INY  
 LDA  
 LSR  
 LSR  
 LSR  
 ROL  
 LSR  
 ROL  
 ORA  
 STA  
 INY  
 LDA  
 AND  
 ORA  
 STA  
 INY  
 DEX  
 BPL  
 LDA  
 LSR  
 LSR  
 ORA  
 STA  
 RTS  
 STA  
 CMP  
 BEQ  
 STX  
 LDA  
 STA  
 SEC  
 SBC  
 BEQ  
 BCS  
 EOR  
 INC  
 BCC  
 ADC  
 DEC  
 CMP  
 BCC  
 LDA  
 CMP  
 BCC  
 LDA  
 TAY  
 LDA  
 AND  
 ASL  
 ORA

0000 0B<sub>6</sub>B<sub>7</sub>B<sub>8</sub> C = D<sub>7</sub>  
 \$26 000A<sub>6</sub> A<sub>7</sub>A<sub>8</sub>D<sub>6</sub>D<sub>7</sub>  
 \$BB33, X [B<sub>1</sub>B<sub>2</sub>B<sub>3</sub>B<sub>4</sub> B<sub>5</sub>B<sub>6</sub>B<sub>7</sub>B<sub>8</sub>]  
 (\$3E), Y STORE IN BUFFER #2  
 \$BC66, X C<sub>6</sub>C<sub>7</sub>C<sub>8</sub>D<sub>8</sub> E<sub>8</sub>000  
 0000 C<sub>6</sub>C<sub>7</sub>C<sub>8</sub>D<sub>8</sub> C = E<sub>8</sub>  
 \$27 0A<sub>6</sub>A<sub>7</sub>A<sub>8</sub> D<sub>6</sub>E<sub>6</sub>E<sub>7</sub>E<sub>8</sub>  
 0000 0C<sub>6</sub>C<sub>7</sub>C<sub>8</sub> C = D<sub>8</sub>  
 \$26 00A<sub>6</sub>A<sub>7</sub> A<sub>8</sub>D<sub>6</sub>D<sub>7</sub>D<sub>8</sub>  
 \$BB66, X [C<sub>1</sub>C<sub>2</sub>C<sub>3</sub>C<sub>4</sub> C<sub>5</sub>C<sub>6</sub>C<sub>7</sub>C<sub>8</sub>]  
 (\$3E), Y STORE IN BUFFER #3  
 \$26 00A<sub>6</sub>A<sub>7</sub> A<sub>8</sub>D<sub>6</sub>D<sub>7</sub>D<sub>8</sub>  
 \$07 0000 0D<sub>6</sub>D<sub>7</sub>D<sub>8</sub>  
 \$BB99, X [D<sub>1</sub>D<sub>2</sub>D<sub>3</sub>D<sub>4</sub> D<sub>5</sub>D<sub>6</sub>D<sub>7</sub>D<sub>8</sub>]  
 (\$3E), Y STORE IN BUFFER #4  
 \$27 0A<sub>6</sub>A<sub>7</sub>A<sub>8</sub> D<sub>6</sub>E<sub>6</sub>E<sub>7</sub>E<sub>8</sub>  
 \$07 0000 0E<sub>6</sub>E<sub>7</sub>E<sub>8</sub>  
 \$BBCC, X [E<sub>1</sub>E<sub>2</sub>E<sub>3</sub>E<sub>4</sub> E<sub>5</sub>E<sub>6</sub>E<sub>7</sub>E<sub>8</sub>]  
 (\$3E), Y STORE IN BUFFER #5

DO 51 GROUPS OF 5 (255)  
 \$B9C5  
 \$BC99 GET LAST 3 BITS OF LAST BYTE  
 \$BBFF COMBINE WITH FIRST 5 BITS  
 (\$3E), Y AND STORE IN BUFFER  
 EXIT

DOS 3.2.1  
 DOS 3.2.1  
 DOS 3.2.1

\$2A SAVE WANTED TRK  
 \$0478 ALREADY THERE?  
 \$BA7E SAVE SLOT/DRIVE  
 \$2B INIT. LOOP COUNT  
 \$0478 GET OLD TRK  
 \$27  
 CURR\_TRK - NEW\_TRK  
 WE'RE THERE  
 \$BA77  
 \$BA3E CURR > NEW (MOVE ARM OUT)  
 \$FF CURR < NEW (MOVE ARM IN)  
 \$0478  
 \$BA43 } COMPUTE |CURR - NEW|  
 \$FE }  
 \$0478 }  
 \$26 }  
 \$BA49 } USE MINIMUM(CURR-NEW, LOOPCOUNT)  
 \$26 }  
 \$0C }  
 \$BA4F }  
 \$0B }  
 \$0478 CONSTRUCT PROPER PHASE SELECT  
 \$03  
 \$2B ADD SLOT/DRIVE

SEEK ARM  
 TO ABSOLUTE  
 TRACK IN Acc  
 Acc = phases  
 (usually two per track)  
 \$478 = current arm  
 position



DOS 3.2.1

BA58-	AA		TAX		
BA59-	BD 81	CO	LDA	#\$C081,X	TURN PHASES ON TO MOVE ARM
BA5C-	B9 90	BA	LDA	#\$BA90,Y	GET PROPER DELAY FACTOR
BA5F-	20 7F	BA	JSR	#\$BA7F	DELAY
BA62-	A5 27		LDA	#\$27	
BA64-	29 03		AND	##\$03	CONSTRUCT PHASE OFF SELECT
BA66-	0A		ASL		
BA67-	05 2B		ORA	#\$2B	
BA69-	AA		TAX		
BA6A-	BD 80	CO	LDA	#\$C080,X	TURN MOTOR PHASE OFF
BA6D-	B9 90	BA	LDA	#\$BA90,Y	AND DELAY
BA70-	20 7F	BA	JSR	#\$BA7F	
BA73-	E6 26		INC	#\$26	NEXT CYCLE
BA75-	D0 B4		BNE	#\$BA2B	
BA77-	A9 FF		LDA	##\$FF	DELAY (FUDGE FACTOR)
BA79-	20 7F	BA	JSR	#\$BA7F	
BA7C-	A6 2B		LDX	#\$2B	RETURN WITH SLOT/DRIVE STILL IN X
BA7E-	60		RTS		

BA7F-	A2 11		LDX	##\$11	
BA81-	CA		DEX		
BA82-	D0 FD		BNE	#\$BA81	
BA84-	E6 46		INC	#\$46	
BA86-	D0 02		BNE	#\$BAA8	
BA88-	E6 47		INC	#\$47	
BA8A-	38		SEC		
BA8B-	E9 01		SBC	##\$01	
BA8D-	D0 F0		BNE	#\$BA7F	
BA8F-	60		RTS		

\$46,\$47 COUNT TIME SPENT SEEKING

ARM MOVE  
DELAY SUBRT

BA90-	01 30		ORA	(\$30,X)	ACCELERATE
BA92-	28		PLP		
BA93-	24 20		BIT	#\$20	
BA95-	1E 1D 1C		ASL	#\$1C1D,X	
BA98-	1C		???		
BA99-	1C		???		
BA9A-	1C		???		
BA9B-	1C		???		
BA9C-	70 2C		BVS	#\$BACA	DECELERATE
BA9E-	26 22		ROL	#\$22	
BAA0-	1F		???		
BAA1-	1E 1D 1C		ASL	#\$1C1D,X	
BAA4-	1C		???		
BAA5-	1C		???		
BAA6-	1C		???		
BAA7-	1C		???		

ARM MOVEMENT  
DELAY TABLE

BAA8-	00		BRK		DISK CODE TO 5 BIT (READ) TRANSLATE TABLE  (SHIFTED 5 BIT) XXXX X000
BAA9-	00		BRK		
BAAA-	00		BRK		
BAAB-	00		BRK		
BAAC-	01 08		ORA	(\$08,X)	
BAAE-	10 18		BPL	#\$BAC8	
BAB0-	02		???		
BAB1-	03		???		
BAB2-	04		???		
BAB3-	05 06		ORA	#\$06	
BAB5-	20 28 30		JSR	#\$3028	
BAB8-	07		???		
BAB9-	09 38		ORA	##\$38	
BABB-	40		RTI		
BABC-	0A		ASL		
BABD-	48		PHA		
BABE-	50 58		BVC	##\$B18	
BAC0-	0B		???		
BAC1-	0C		???		



BAC2-	0D 0E 0F	ORA	#0F0E
BAC5-	11 12	ORA	(#12), Y
BAC7-	13	???	
BAC8-	14	???	
BAC9-	15 16	ORA	#16, X
BACB-	17	???	
BACC-	19 1A 1B	ORA	#1B1A, Y
BACF-	1C	???	
BAD0-	1D 1E 21	ORA	#211E, X
BAD3-	22	???	
BAD4-	23	???	
BAD5-	24 60	BIT	#60
BAD7-	68	PLA	
BAD8-	25 26	AND	#26
BADA-	70 78	BVS	#BB54
BADC-	27	???	
BADD-	80	???	
BADE-	88	DEY	
BADF-	90 29	BCC	#BB0A
BAE1-	2A	RCL	
BAE2-	2B	???	
BAE3-	2C 2D 2E	BIT	#2E2D
BAE4-	2F	???	
BAE7-	31 32	AND	(#32), Y
BAE9-	33	???	
BAEA-	98	TYA	
BAEB-	A0 34	LDY	##34
BAED-	A8	TAY	
BAEE-	B0 B8	BCS	#BAAS
BAF0-	35 36	AND	#36, X
BAF2-	37	???	
BAF3-	39 3A C0	AND	#C03A, Y
BAF6-	C8	INY	
BAF7-	D0 3B	BNE	#BB34
BAF9-	3C	???	
BAFA-	D8	CLD	
BAFB-	E0 3E	CPX	##3E
BAFD-	E8	INX	
BAFE-	F0 F8	BEQ	#BAF8

↑  
 READ TRANSLATE  
 TABLE  
 CONTINUED  
 ↓

BB00-	00	BRK	
BB01-	00	BRK	
BB02-	00	BRK	
BB03-	00	BRK	
BB04-	00	BRK	
BB05-	00	BRK	
BB06-	00	BRK	
BB07-	00	BRK	
BB08-	00	BRK	
BB09-	00	BRK	
BB0A-	00	BRK	
BB0B-	00	BRK	
BB0C-	00	BRK	
BB0D-	00	BRK	
BB0E-	00	BRK	
BB0F-	00	BRK	
BB10-	00	BRK	
BB11-	00	BRK	
BB12-	00	BRK	
BB13-	00	BRK	
BB14-	C8	INY	
BB15-	D0 A0	BNE	#BAB7
BB17-	C0 B0	CPY	##B0
BB19-	C8	INY	



256 BYTE  
 PRIMARY  
 DATA BUFFER  
 (5 BIT FORMAT)

CONTAINS FIRST 5 BITS  
 OF ALL BYTES



```

BB1A- DO 28 BNE $BB44
BB1C- 18 CLC
BB1D- B0 A8 BCS $BAC7
BB1F- 00 BRK
BB20- 00 BRK
BB21- 00 BRK
BB22- A0 D0 LDY #$D0
BB24- 00 BRK
BB25- 00 BRK
BB26- 28 PLP
BB27- C8 INY
BB28- D0 C8 BNE $BAF2
BB2A- 08 PHP
BB2B- 18 CLC
BB2C- C8 INY
BB2D- C0 00 CPY #$00
BB2F- 00 BRK
BB30- A8 TAY
BB31- 48 PHA
BB32- 98 TYA
BB33- 00 BRK
BB34- 00 BRK
BB35- 00 BRK
BB36- 00 BRK
BB37- 00 BRK
BB38- 00 BRK
BB39- 00 BRK
BB3A- 00 BRK
BB3B- 00 BRK
BB3C- 00 BRK
BB3D- 00 BRK
BB3E- 00 BRK
BB3F- 00 BRK
BB40- 00 BRK
BB41- 00 BRK
BB42- 00 BRK
BB43- 00 BRK
BB44- 00 BRK
BB45- 98 TYA
BB46- 00 BRK
BB47- C8 INY
BB48- D0 C0 BNE $BB0A
BB4A- D0 B8 BNE $BB04
BB4C- D0 D8 BNE $BB26
BB4E- A0 00 LDY #$00
BB50- B8 CLV
BB51- C0 60 CPY #$60
BB53- 50 10 BVC $BB65
BB55- B0 D0 BCS $BB27
BB57- 60 RTS
BB58- 50 40 BVC $BB9A
BB5A- C0 A0 CPY #$A0
BB5C- C0 00 CPY #$00
BB5E- 10 C8 BPL $BB28
BB60- C8 INY
BB61- 60 RTS
BB62- 00 BRK
BB63- 00 BRK
BB64- 00 BRK
BB65- 00 BRK
BB66- 00 BRK
BB67- 00 BRK
BB68- 00 BRK

```

  
 PRIMARY DATA  
 BUFFER CONTINUED  


BB69-	00	BRK	
BB6A-	00	BRK	
BB6B-	00	BRK	
BB6C-	00	BRK	
BB6D-	00	BRK	
BB6E-	00	BRK	
BB6F-	00	BRK	
BB70-	00	BRK	
BB71-	00	BRK	
BB72-	00	BRK	
BB73-	00	BRK	
BB74-	00	BRK	
BB75-	00	BRK	
BB76-	00	BRK	
BB77-	00	BRK	
BB78-	00	BRK	
BB79-	28	PLP	
BB7A-	C0 C0	CPY	##C0
BB7C-	C8	INY	
BB7D-	D0 A0	BNE	\$BB1F
BB7F-	A0 D0	LDY	##D0
BB81-	A0 60	LDY	##60
BB83-	28	PLP	
BB84-	C0 28	CPY	##28
BB86-	B0 18	BCS	\$BBA0
BB88-	A8	TAY	
BB89-	C8	INY	
BB8A-	28	PLP	
BB8B-	B0 00	BCS	\$BB8D
BB8D-	D0 C0	BNE	\$BB4F
BB8F-	D0 00	BNE	\$BB91
BB91-	00	BRK	
BB92-	28	PLP	
BB93-	D0 28	BNE	\$BBBD
BB95-	00	BRK	
BB96-	00	BRK	
BB97-	48	PHA	
BB98-	10 00	BPL	\$BB9A
BB9A-	00	BRK	
BB9B-	00	BRK	
BB9C-	00	BRK	
BB9D-	00	BRK	
BB9E-	00	BRK	
BB9F-	00	BRK	
BBA0-	00	BRK	
BBA1-	00	BRK	
BBA2-	00	BRK	
BBA3-	00	BRK	
BBA4-	00	BRK	
BBA5-	00	BRK	
BBA6-	00	BRK	
BBA7-	00	BRK	
BBA8-	00	BRK	
BBA9-	00	BRK	
BBAA-	00	BRK	
BBAB-	00	BRK	
BBAC-	00	BRK	
BBAD-	A8	TAY	
BBAE-	D0 C8	BNE	\$BB78
BBB0-	C8	INY	
BBB1-	A0 B0	LDY	##B0
BBB3-	C8	INY	
BBB4-	C0 00	CPY	##00

  
 PRIMARY DATA  
 BUFFER CONTINUED  


BBB6-	00	BRK	
BBB7-	C0 B0	CPY	##B0
BBB9-	18	CLC	
BBBA-	00	BRK	
BBBB-	B0 C8	BCS	##BB85
BBBD-	D0 18	BNE	##BBD7
BBBF-	10 D0	BPL	##BB91
BBC1-	C8	INY	
BBC2-	D0 60	BNE	##BC24
BBC4-	50 40	BVC	##BC06
BBC6-	C8	INY	
BBC7-	A0 10	LDY	##10
BBC9-	68	PLA	
BBCA-	30 08	BMI	##BBD4
BBCC-	00	BRK	
BBCD-	00	BRK	
BBCE-	00	BRK	
BBCF-	00	BRK	
BBD0-	00	BRK	
BBD1-	00	BRK	
BBD2-	00	BRK	
BBD3-	00	BRK	
BBD4-	00	BRK	
BBD5-	00	BRK	
BBD6-	00	BRK	
BBD7-	00	BRK	
BBD8-	00	BRK	
BBD9-	00	BRK	
BBDA-	00	BRK	
BBDB-	00	BRK	
BBDC-	00	BRK	
BBDD-	00	BRK	
BBDE-	00	BRK	
BBDF-	50 28	BVC	##BC09
BBE1-	A0 D0	LDY	##D0
BBE3-	C0 A0	CPY	##A0
BBE5-	B8	CLV	
BBE6-	C0 C8	CPY	##C8
BBE8-	60	RTS	
BBE9-	28	PLP	
BBEA-	A8	TAY	
BBEB-	B0 00	BCS	##BBED
BBED-	60	RTS	
BBEE-	28	PLP	
BBEF-	C8	INY	
BBF0-	C0 00	CPY	##00
BBF2-	10 C0	BPL	##BBB4
BBF4-	D0 C0	BNE	##BBB6
BBF6-	28	PLP	
BBF7-	B0 00	BCS	##BBF9
BBF9-	A0 A0	LDY	##A0
BBFB-	10 B0	BPL	##BBAD
BBFD-	B8	CLV	
BBFE-	00	BRK	
BBFF-	00	BRK	

↑  
PRIMARY DATA  
BUFFER CONTINUED  
↓

BC00-	00	BRK	
BC01-	00	BRK	
BC02-	00	BRK	
BC03-	00	BRK	
BC04-	00	BRK	
BC05-	00	BRK	
BC06-	00	BRK	
BC07-	00	BRK	

CONTAINS MERGED  
LAST 3 BITS OF  
ALL BYTES

154 BYTE  
SECONDARY  
DATA BUFFER  
(5 BIT FORMAT)



BC08-	00		BRK	
BC09-	00		BRK	
BC0A-	00		BRK	
BC0B-	00		BRK	
BC0C-	00		BRK	
BC0D-	00		BRK	
BC0E-	00		BRK	
BC0F-	00		BRK	
BC10-	00		BRK	
BC11-	00		BRK	
BC12-	20	20 30	JSR	#3020
BC15-	A0	10	LDY	##10
BC17-	38		SEC	
BC18-	E0	00	CPX	##00
BC1A-	08		PHP	
BC1B-	08		PHP	
BC1C-	C0	E8	CPY	##E8
BC1E-	A8		TAY	
BC1F-	68		PLA	
BC20-	70	20	BVS	#BC42
BC22-	00		BRK	
BC23-	58		CLI	
BC24-	78		SEI	
BC25-	10	38	BPL	#BC5F
BC27-	78		SEI	
BC28-	40		RTI	
BC29-	B8		CLV	
BC2A-	60		RTS	
BC2B-	80		???	
BC2C-	30	80	BMI	#BBAE
BC2E-	00		BRK	
BC2F-	68		PLA	
BC30-	10	70	BPL	#BCA2
BC32-	E0	00	CPX	##00
BC34-	00		BRK	
BC35-	00		BRK	
BC36-	00		BRK	
BC37-	00		BRK	
BC38-	00		BRK	
BC39-	00		BRK	
BC3A-	00		BRK	
BC3B-	00		BRK	
BC3C-	00		BRK	
BC3D-	00		BRK	
BC3E-	00		BRK	
BC3F-	00		BRK	
BC40-	00		BRK	
BC41-	00		BRK	
BC42-	00		BRK	
BC43-	00		BRK	
BC44-	00		BRK	
BC45-	E0	A0	CPX	##A0
BC47-	D0	90	BNE	#BBD9
BC49-	60		RTS	
BC4A-	00		BRK	
BC4B-	20	80 28	JSR	#2880
BC4E-	18		CLC	
BC4F-	10	28	BPL	#BC79
BC51-	D0	28	BNE	#BC7B
BC53-	10	E8	BPL	#BC3D
BC55-	70	78	BVS	#BCCF
BC57-	30	00	BMI	#BC59
BC59-	F8		SED	

↑

SECONDARY DATA  
BUFFER CONTINUED

↓

BC5A-	A0 08	LDY	##08
BC5C-	20 00 C0	JSR	##C000
BC5F-	30 30	BMI	##BC91
BC61-	20 00 78	JSR	##7800
BC64-	70 10	BVS	##BC76
BC66-	00	BRK	
BC67-	00	BRK	
BC68-	00	BRK	
BC69-	00	BRK	
BC6A-	00	BRK	
BC6B-	00	BRK	
BC6C-	00	BRK	
BC6D-	00	BRK	
BC6E-	00	BRK	
BC6F-	00	BRK	
BC70-	00	BRK	
BC71-	00	BRK	
BC72-	00	BRK	
BC73-	00	BRK	
BC74-	00	BRK	
BC75-	00	BRK	
BC76-	00	BRK	
BC77-	00	BRK	
BC78-	00	BRK	
BC79-	08	PHP	
BC7A-	68	PLA	
BC7B-	A0 F0	LDY	##F0
BC7D-	08	PHP	
BC7E-	00	BRK	
BC7F-	18	CLC	
BC80-	58	CLI	
BC81-	18	CLC	
BC82-	78	SEI	
BC83-	38	SEC	
BC84-	A8	TAY	
BC85-	10 60	BPL	##BCE7
BC87-	08	PHP	
BC88-	C8	INY	
BC89-	30 08	BMI	##BC93
BC8B-	40	RTI	
BC8C-	38	SEC	
BC8D-	88	DEY	
BC8E-	98	TYA	
BC8F-	68	PLA	
BC90-	70 08	BVS	##BC9A
BC92-	38	SEC	
BC93-	70 00	BVS	##BC95
BC95-	30 78	BMI	##BD0F
BC97-	A8	TAY	
BC98-	00	BRK	
BC99-	00	BRK	
BC9A-	AB	???	
BC9B-	AD AE AF	LDA	##AFAE
BC9E-	B5 B6	LDA	##B6, X
BCA0-	B7	???	
BCA1-	BA	TSX	
BCA2-	BB	???	
BCA3-	BD BE BF	LDA	##BFBE, X
BCA6-	D6 D7	DEC	##D7, X
BCA8-	DA	???	
BCA9-	DB	???	
BCAA-	DD DE DF	CMP	##DFDE, X
BCAD-	EA	NOP	



SECONDARY DATA  
BUFFER CONTINUED



5 BIT TO DISK CODE (WRITE) TRANSLATE TABLE
---

BCAE-	EB		???	
BCAF-	ED	EE	EF	SBC \$EFEE
BCB2-	F5	F6		SBC \$F6, X
BCB4-	F7			???
BCB5-	FA			???
BCB6-	FB			???
BCB7-	FD	FE	FF	SBC \$FFFE, X

RWTS PATCH AREA
--------------------

BCBA-	1C			???	
BCBB-	1C			???	
BCBC-	1C			???	
BCBD-	00			BRK	
BCBE-	00			BRK	
BCBF-	00			BRK	
BCC0-	A4	2D		LDY	\$2D
BCC2-	B9	D0	3C	LDA	\$3C00, Y
BCC5-	A0	05		LDY	##05
BCC7-	4C	0A	3E	JMP	\$3E0A
BCCA-	00			BRK	
BCCB-	00			BRK	
BCCC-	00			BRK	
BCCD-	00			BRK	
BCCE-	00			BRK	
BCCF-	00			BRK	
BCD0-	00			BRK	
BCD1-	05	0A		ORA	\$0A
BCD3-	02			???	
BCD4-	07			???	
BCD5-	0C			???	
BCD6-	04			???	
BCD7-	09	01		ORA	##01
BCD9-	06	0B		ASL	\$0B
BCDB-	03			???	
BCDC-	08			PHP	
BCDD-	00			BRK	
BCDE-	00			BRK	
BCDF-	00			BRK	
BCE0-	00			BRK	
BCE1-	00			BRK	
BCE2-	00			BRK	
BCE3-	00			BRK	
BCE4-	00			BRK	
BCE5-	00			BRK	
BCE6-	00			BRK	
BCE7-	00			BRK	
BCE8-	00			BRK	
BCE9-	00			BRK	
BCEA-	00			BRK	
BCEB-	00			BRK	
BCEC-	00			BRK	
BCED-	00			BRK	
BCEE-	00			BRK	
BCEF-	00			BRK	
BCF0-	00			BRK	
BCF1-	00			BRK	
BCF2-	00			BRK	
BCF3-	00			BRK	
BCF4-	00			BRK	
BCF5-	00			BRK	
BCF6-	00			BRK	
BCF7-	00			BRK	
BCF8-	00			BRK	
BCF9-	00			BRK	
BCFA-	00			BRK	

OLD  
PATCH

OLD  
PATCH

DOS 3.2.1  
INSERTION

BCFB-	00	BRK
BCFC-	00	BRK
BCFD-	00	BRK
BCFE-	00	BRK
BCFF-	00	BRK

BD00-	84 48	STY	\$48	UPON ENTRY A:Y POINT AT THE
BD02-	85 49	STA	\$49	I/O CONTROL BLOCK (IOB)
BD04-	A0 01	LDY	##01	} GET SLOT #
BD06-	B1 48	LDA	(\$48),Y	
BD08-	AA	TAX		} ONE RECAL PER CALL
BD09-	8C F8 04	STY	\$04F8	
BD0C-	A0 0F	LDY	##0F	} CHANGE SLOTS?
BD0E-	D1 48	CMP	(\$48),Y	
BD10-	F0 1B	BEQ	##BD2D	} SAVE NEW SLOT #
BD12-	8A	TXA		
BD13-	48	PHA		} GET PREVIOUS SLOT #
BD14-	B1 48	LDA	(\$48),Y	
BD16-	AA	TAX		} GET NEW
BD17-	68	PLA		
BD18-	48	PHA		} IT WILL BE PREVIOUS
BD19-	91 48	STA	(\$48),Y	
BD1B-	BD 8E C0	LDA	\$C08E,X	} GET INTO READ MODE
BD1E-	A0 08	LDY	##08	
BD20-	BD 8C C0	LDA	\$C08C,X	} WAIT FOR DRIVE TO TURN OFF
BD23-	DD 8C C0	CMP	\$C08C,X	
BD26-	D0 F6	BNE	##BD1E	} DOES ANY DATA REMAIN
BD28-	88	DEY		
BD29-	D0 F8	BNE	##BD23	} STABLE FOR 96 MICROSECONDS?
BD2B-	68	PLA		
BD2C-	AA	TAX		} IF SO, DISK CAN'T BE SPINNING
BD2D-	BD 8E C0	LDA	\$C08E,X	
BD30-	BD 8C C0	LDA	\$C08C,X	} GET NEW SLOT #
BD33-	BD 8C C0	LDA	\$C08C,X	
BD36-	48	PHA		} GET NEW SLOT #
BD37-	68	PLA		
BD38-	8E F8 05	STX	\$05F8	} READ MODE ON NEW DRIVE
BD3B-	DD 8C C0	CMP	\$C08C,X	
BD3E-	08	PHP		} GET A DATA BYTE
BD3F-	BD 89 C0	LDA	\$C089,X	
BD42-	A0 06	LDY	##06	} DELAY
BD44-	B1 48	LDA	(\$48),Y	
BD46-	99 36 00	STA	\$0036,Y	} SAVE SLOT
BD49-	C8	INY		
BD4A-	C0 0A	CPY	##0A	} DATA CHANGED? (DISK SPINNING)
BD4C-	D0 F6	BNE	##BD44	
BD4E-	A0 02	LDY	##02	} MAYBE, REMEMBER
BD50-	B1 48	LDA	(\$48),Y	
BD52-	A0 10	LDY	##10	} TURN MOTOR ON IN CASE
BD54-	D1 48	CMP	(\$48),Y	
BD56-	F0 06	BEQ	##BD5E	} COPY DEVICE CHAR. TABLE PTR (DCT)
BD58-	91 48	STA	(\$48),Y	
BD5A-	28	PLP		} AND DATA BUFFER PTR TO ZPAGE
BD5B-	A0 00	LDY	##00	
BD5D-	08	PHP		} GET DRIVE #
BD5E-	6A	ROR		
BD5F-	BD 8A C0	LDA	\$C08A,X	} SAME AS BEFORE?
BD62-	B0 03	BCS	##BD67	
BD64-	BD 8B C0	LDA	\$C08B,X	} YES
BD67-	66 35	ROR	##35	
BD69-	A0 02	LDY	##02	} NO, IT WILL BE
BD6B-	B1 3C	LDA	(\$3C),Y	
BD6D-	85 46	STA	\$46	} FORCE WAIT FOR MOTOR TO COME ON
BD6F-	C8	INY		

RWTS  
MAIN  
ROUTINE

EXTERNAL ENT  
IS THROUGH \$B7E  
TO ALLOW DISAB  
FOR INTERRUPT

dos 3.2.1 (BD61)  
ELSE, SELECT 1 (2)  
SAVE CARRY IN ZPAGE AS DRIVE INDICATOR

GET MOTOR ON TIME FROM DCT

WHICH DRIVE?  
ASSUME DRIVE 0 (1)

dos 3.2.1 (BD61)  
ELSE, SELECT 1 (2)  
SAVE CARRY IN ZPAGE AS DRIVE INDICATOR

GET MOTOR ON TIME FROM DCT



BD70- B1 3C  
 BD72- 85 47  
 BD74- C8  
 BD75- B1 48  
 BD77- 20 3B BE  
 BD7A- 28  
 BD7B- D0 0D  
 BD7D- A0 12  
 BD7F- 88  
 BD80- D0 FD  
 BD82- E6 46  
 BD84- D0 F7  
 BD86- E6 47  
 BD88- D0 F3  
 BD8A- A0 0C  
 BD8C- B1 48  
 BD8E- F0 55  
 BD90- C9 04  
 BD92- F0 53  
 BD94- 6A  
 BD95- 08  
 BD96- B0 03  
 BD98- 20 00 B8  
 BD9B- A0 30  
 BD9D- 8C 78 05  
 BDA0- AE F8 05  
 BDA3- 20 65 B9  
 BDA6- 90 1F  
 BDA8- CE 78 05  
 BDAB- 10 F3  
 BDAD- AD 78 04  
 BDB0- 48  
 BDB1- A9 60  
 BDB3- 20 82 BE  
 BDB6- CE F8 04  
 BDB9- D0 23  
 BDBB- A9 00  
 BDBD- 20 3B BE  
 BDC0- 68  
 BDC1- 20 3B BE  
 BDC4- 4C 9B BD  
 BDC7- A4 2E  
 BDC9- CC 78 04  
 BDCC- F0 22  
 BDCE- AD 78 04  
 BDD1- 48  
 BDD2- 98  
 BDD3- 20 82 BE  
 BDD6- 68  
 BDD7- CE 78 05  
 BDDA- 10 E5  
 BDDC- 30 CA  
 BDDE- 68  
 BDDF- A9 40  
 BDE1- 28  
 BDE2- 4C 29 BE  
 BDE5- F0 40  
 BDE7- A0 03  
 BDE9- B1 48  
 BDEB- 85 2F  
 BDED- 4C 9C BE  
 BDF0- A0 03  
 BDF2- B1 48

LDA (\$3C),Y  
 STA \$47  
 INY  
 LDA (\$48),Y  
 JSR \$BE3B  
 PLP  
 BNE \$BD8A  
 LDY #\$12  
 DEY  
 BNE \$BD7F  
 INC \$46  
 BNE \$BD7D  
 INC \$47  
 BNE \$BD7D  
 LDY #\$0C  
 LDA (\$48),Y  
 BEQ \$BDE5  
 CMP #\$04  
 BEQ \$BDE7  
 ROR  
 PHP  
 BCS \$BD9B  
 JSR \$B800  
 LDY #\$30  
 STY \$0578  
 LDX \$05F8  
 JSR \$B965  
 BCC \$BDC7  
 DEC \$0578  
 BPL \$BDA0  
 LDA \$0478  
 PHA  
 LDA #\$60  
 JSR \$BE82  
 DEC \$04F8  
 BNE \$BDDE  
 LDA #\$00  
 JSR \$BE3B  
 PLA  
 JSR \$BE3B  
 JMP \$BD9B  
 LDY \$2E  
 CPY \$0478  
 BEQ \$BDF0  
 LDA \$0478  
 PHA  
 TYA  
 JSR \$BE82  
 PLA  
 DEC \$0578  
 BPL \$BDC1  
 BMI \$BDA8  
 PLA  
 LDA #\$40  
 PLP  
 JMP \$BE29  
 BEQ \$BE27  
 LDY #\$03  
 LDA (\$48),Y  
 STA \$2F  
 JMP \$BE9C  
 LDY #\$03  
 LDA (\$48),Y

SECOND HALF MOTOR ON TIME

GET DESTINATION TRACK

MAY AS WELL SEEK WHILE WAITING WAS MOTOR ALREADY ON?

100 USECS

WAIT FOR MOTOR TO COME UP TO SPEED

WHAT OPERATION?

NULL, DO NOTHING  
FORMAT (INIT) DISK?  
C=1 READ 0 WRITE

READ?  
CONVERT WRITE DATA TO 5 BIT CODES  
48 ERROR RETRYS

SLOT #  
SEARCH FOR SECTOR HDR  
GOT IT  
ELSE RETRY UP TO 48 TIMES

SAVE REAL TRACK

PRETEND WE ARE ON TRACK 80

ONE TRY ONLY AT THIS THEN DRIVE ERROR

RECAL THE ARM TO TRACK 0 TRACK I WANTED  
SEEK TO IT

TRY AGAIN FOR SECTOR HDR  
STILL ON TRACK WE WERE ON

YES  
NO, SAVE

COMPUTE CORRECTION NEEDED

COUNT RETRY AND SEEK TO IT  
THIS ISN'T WORKING, RECAL ARM

DRIVE ERROR 40

GO HANDLE ERROR  
GO TO EXIT

GET VALID FORMAT DISK

FORCE IT TO SECTOR HDRS  
GO FORMAT DISK

GET VALID EXPECTED

BDF4-	48		PHA		
BDF5-	A5	2F	LDA	\$2F	
BDF7-	A0	0E	LDY	##0E	
BDF9-	91	48	STA	(\$48),Y	
BDFB-	68		PLA		
BDFC-	F0	08	BEQ	\$BE06	
BDFE-	C5	2F	CMP	\$2F	
BE00-	F0	04	BEQ	\$BE06	
BE02-	A9	20	LDA	##20	
BE04-	D0	DB	BNE	\$BDE1	
BE06-	A0	05	LDY	##05	
BE08-	A5	2D	LDA	\$2D	
BE0A-	D1	48	CMP	(\$48),Y	
BE0C-	F0	09	BEQ	\$BE17	
BE0E-	CE	78	DEC	\$0578	
BE11-	10	8D	BPL	\$BDA0	
BE13-	A9	80	LDA	##80	
BE15-	D0	CA	BNE	\$BDE1	
BE17-	28		PLP		
BE18-	90	18	BCC	\$BE32	
BE1A-	20	FD	B8	JSR	\$B8FD
BE1D-	08		PHP		
BE1E-	B0	88	BCS	\$BDA8	
BE20-	28		PLP		
BE21-	20	C1	B9	JSR	\$B9C1
BE24-	AE	F8	05	LDX	\$05F8
BE27-	18		CLC		
BE28-	24	38	<del>BIT</del>	\$38	
BE2A-	A0	0D	LDY	##0D	
BE2C-	91	48	STA	(\$48),Y	
BE2E-	BD	88	C0	LDA	\$C088,X
BE31-	60		RTS		
BE32-	20	6A	B8	JSR	\$B86A
BE35-	90	F0	BCC	\$BE27	
BE37-	A9	10	LDA	##10	
BE39-	B0	EE	BCS	\$BE29	
BE3B-	48		PHA		
BE3C-	A0	01	LDY	##01	
BE3E-	B1	3C	LDA	(\$3C),Y	
BE40-	6A		ROR		
BE41-	68		PLA		
BE42-	90	08	BCC	\$BE4C	
BE44-	0A		ASL		
BE45-	20	4C	BE	JSR	\$BE4C
BE48-	4E	78	04	LSR	\$0478
BE4B-	60		RTS		
BE4C-	85	2E	STA	\$2E	
BE4E-	BD	80	C0	LDA	\$C080,X
BE51-	BD	82	C0	LDA	\$C082,X
BE54-	BD	84	C0	LDA	\$C084,X
BE57-	BD	86	C0	LDA	\$C086,X
BE5A-	20	7B	BE	JSR	\$BE7B
BE5D-	B9	78	04	LDA	\$0478,Y
BE60-	24	35	BIT	\$35	
BE62-	30	03	BMI	\$BE67	
BE64-	B9	F8	04	LDA	\$04F8,Y
BE67-	8D	78	04	STA	\$0478
BE6A-	A5	2E	LDA	\$2E	
BE6C-	24	35	BIT	\$35	
BE6E-	30	05	BMI	\$BE75	
BE70-	99	F8	04	STA	\$04F8,Y
BE73-	10	03	BPL	\$BE78	
BE75-	99	78	04	STA	\$0478,Y

GET VALID FOUND IN SECTOR HDR  
 RETURN IT TO IOB  
 VALID = 0 MATCHES ANY  
 OTHERWISE, DO THEY MATCH?  
 NO, VOL MISMATCH ERROR 20

IS THIS THE SECTOR WANTED?  
 NO, RETRY NEXT ONE  
 ELSE, READ ERROR 80

READ OR WRITE?  
 READ - GET 5 BIT DATA READ

BAD CHKSUM - RETRY  
 CONVERT READ DATA TO 8 BIT  
 GET SLOT #  
NORMAL RETURN EXIT

SEC ←  
~~BIT~~ \$38  
 STORE ERROR NUMBER IN IOB  
 TURN MOTOR OFF  
 AND LEAVE

WRITE 5 BIT CODES WRITE  
 WRITE PROTECT ERROR 10

SAVE TRACK  
 TWO PHASE DISC? SEEK SUBRTN  
 SEEK TO TRK  
 IN ACC.

NO  
 YES, DOUBLE TRACK #  
 SEEK  
 THEN HALVE RESULT

SAVE TRACK WANTED  
 } TURN OFF ALL STEP MOTOR PHASES

GET SLOT # IN Y  
 GET DRIVE 0'S CURR TRACK #  
 DRIVE 0?  
 YES  
 ELSE, GET DRIVE 1'S CURR TRACK #  
 SET CURRENT TRACK FOR ABS-SEEK  
 TRACK WANTED  
 WHICH DRIVE?  
 IF DRIVE 0, REMEMBER NEW ARM LOC.  
 ELSE DRIVE 1'S TABLE IS UPDATED

NOW GO TO ABSOLUTE SEEK

BE78-	4C 1E BA	JMP	#BA1E
BE7B-	8A	TXA	
BE7C-	4A	LSR	
BE7D-	4A	LSR	
BE7E-	4A	LSR	
BE7F-	4A	LSR	
BE80-	A8	TAY	
BE81-	60	RTS	
BE82-	48	PHA	
BE83-	A0 02	LDY	##02
BE85-	B1 48	LDA	(\$48),Y
BE87-	6A	ROR	
BE88-	66 35	ROR	\$35
BE8A-	20 7B BE	JSR	#BE7B
BE8D-	68	PLA	
BE8E-	0A	ASL	
BE8F-	24 35	BIT	\$35
BE91-	30 05	BMI	#BE98
BE93-	99 F8 04	STA	\$04F8,Y
BE96-	10 03	BPL	#BE9B
BE98-	99 78 04	STA	\$0478,Y
BE9B-	60	RTS	
BE9C-	A9 80	LDA	##80
BE9E-	8D 78 04	STA	\$0478
BEA1-	A9 00	LDA	##00
BEA3-	85 41	STA	\$41
BEA5-	20 1E BA	JSR	#BA1E
BEA8-	A9 AA	LDA	##AA
BEAA-	85 4A	STA	\$4A
BEAC-	A0 50	LDY	##50
BEAE-	84 47	STY	\$47
BEB0-	A9 27	LDA	##27
BEB2-	85 4B	STA	\$4B
BEB4-	BD 8D C0	LDA	\$C08D,X
BEB7-	BD 8E C0	LDA	\$C08E,X
BEBA-	A9 FF	LDA	##FF
BEBC-	9D 8F C0	STA	\$C08F,X
BEBF-	DD 8C C0	CMP	\$C08C,X
BEC2-	24 00	BIT	\$00
BEC4-	88	DEY	
BEC5-	F0 0F	BEQ	#BED6
BEC7-	48	PHA	
BEC8-	68	PLA	
BEC9-	EA	NOP	
BECA-	48	PHA	
BECB-	68	PLA	
BECC-	EA	NOP	
BECD-	EA	NOP	
BECE-	9D 8D C0	STA	\$C08D,X
BED1-	DD 8C C0	CMP	\$C08C,X
BED4-	B0 EE	BCS	#BEC4
BED6-	C6 4B	DEC	\$4B
BED8-	D0 F0	BNE	#BECA
BEDA-	A4 47	LDY	\$47
BEDC-	EA	NOP	
BEDD-	EA	NOP	
BEDE-	D0 06	BNE	#BEE6
BEE0-	48	PHA	
BEE1-	68	PLA	
BEE2-	48	PHA	
BEE3-	68	PLA	
BEE4-	C1 00	CMP	(\$00,X)
BEE6-	EA	NOP	

PUT SLOT # IN Y

SAVE TRACK SET TRACK #

GET DRIVE #

SAVE IT GET SLOT IN Y AND TRACK IN A ASSUME TWO PHASE TRACKS WHICH DRIVE?

STORE TRACK IN DRIVE 0 TABLE

OR IN DRIVE 1 TABLE

RECAL ARM TO TRACK 0

HANDY CONSTANT

80 SYNC'S BETWEEN SECTORS

FIRST TIME ON TRK WRITE 39\*256 SYNC

CLEAR WRITE MODE

READ MODE WHILE UPDATING LATCH

WRITE FF'S (AUTO SYNC CODES)

CLEAR LATCH (WRITE OFF) (TIMING)

FILL TRACK WITH FF'S

WRITE SYNC'S EVERY 27 USECS

31\*5 cycles

WRITE ON WRITE OFF

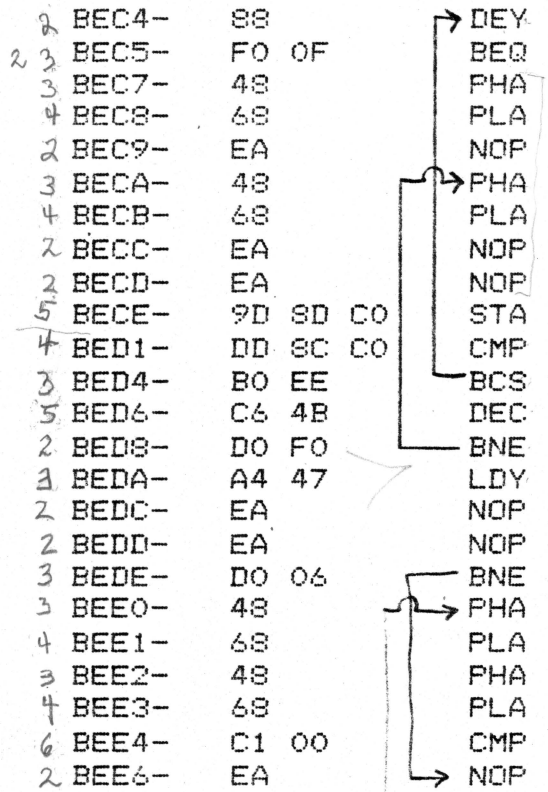
DONE 39 TIMES

80 BYTES BEFORE SECTOR HDR

JUMP INTO LOOP

(TIMING)

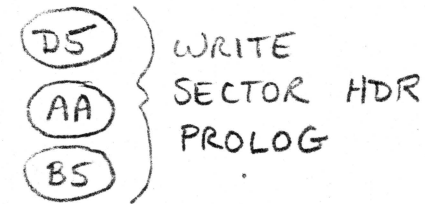
FORMAT DISK





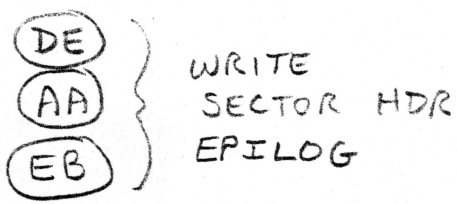
5	BEE7-	9D 8D CO	STA	\$C08D, X
4	BEEA-	DD 8C CO	CMP	\$C08C, X
2	BEED-	88	DEY	
2 3	BEEE-	D0 F0	BNE	\$BEE0
2	BEF0-	A9 D5	LDA	##D5
6	BEF2-	20 CA BF	JSR	\$BFCA <sup>11+5</sup>
2	BEF5-	A9 AA	LDA	##AA
6	BEF7-	20 CB BF	JSR	\$BFCE
	BEFA-	A9 B5	LDA	##B5
	BEFC-	20 CB BF	JSR	\$BFCE
3	BEFF-	A5 2F	LDA	\$2F
	BF01-	20 BB BF	JSR	\$BFBB
	BF04-	A5 41	LDA	\$41
	BF06-	20 BB BF	JSR	\$BFBB
	BF09-	A5 4B	LDA	\$4B
	BF0B-	20 BB BF	JSR	\$BFBB
	BF0E-	A5 2F	LDA	\$2F
	BF10-	45 41	EOR	\$41
	BF12-	45 4B	EOR	\$4B
	BF14-	48	PHA	
	BF15-	4A	LSR	
	BF16-	05 4A	ORA	\$4A
	BF18-	9D 8D CO	STA	\$C08D, X
	BF1B-	DD 8C CO	CMP	\$C08C, X
	BF1E-	68	PLA	
	BF1F-	09 AA	ORA	##AA
	BF21-	20 CA BF	JSR	\$BFCA
	BF24-	A9 DE	LDA	##DE
	BF26-	20 CB BF	JSR	\$BFCE
	BF29-	A9 AA	LDA	##AA
	BF2B-	20 CB BF	JSR	\$BFCE
	BF2E-	A9 EB	LDA	##EB
	BF30-	20 CB BF	JSR	\$BFCE
	BF33-	A9 FF	LDA	##FF
	BF35-	20 CB BF	JSR	\$BFCE
	BF38-	A0 02	LDY	##02
	BF3A-	84 46	STY	\$46
	BF3C-	A0 AD	LDY	##AD
	BF3E-	D0 06	BNE	\$BF46
	BF40-	88	DEY	
	BF41-	F0 0D	BEQ	\$BF50
	BF43-	48	PHA	
	BF44-	68	PLA	
	BF45-	EA	NOP	
	BF46-	48	PHA	
	BF47-	68	PLA	
	BF48-	9D 8D CO	STA	\$C08D, X
	BF4B-	DD 8C CO	CMP	\$C08C, X
	BF4E-	B0 F0	BCS	\$BF40
	BF50-	C6 46	DEC	\$46
	BF52-	D0 F2	BNE	\$BF46
	BF54-	A4 47	LDY	\$47
	BF56-	18	CLC	
	BF57-	24 00	BIT	\$00
	BF59-	9D 8D CO	STA	\$C08D, X
	BF5C-	BD 8C CO	LDA	\$C08C, X
	BF5F-	A5 4B	LDA	\$4B
	BF61-	69 0A	ADC	##0A
	BF63-	85 4B	STA	\$4B
	BF65-	E9 0C	SBC	##0C
	BF67-	F0 0A	BEQ	\$BF73
	BF69-	B0 01	BCS	\$BF6C
	BF6B-	2C 85 4B	BIT	\$4B85

WRITE 80 SYNC'S BETWEEN SECTOR



— VOL (TWO 5 BIT CODES)  
 — TRACK (TWO 5 BIT CODES)  
 — SECTOR (TWO 5 BIT CODES)  
 } COMPUTE CHECKSUM  
 } WRITE CHECKSUM (1ST HALF)

SECOND HALF OF CHECKSUM



BEGIN WRITING SYNC'S AGAIN  
 OVER DATA AREA  
 TWICE THRU NEXT SYNC LOOP  
 FIRST TIME 173 SYNC'S (NEXT 256)  
 JUMP INTO LOOP

DUMMY DATA PORTION

WRITE 429 SYNC'S

SYNC'S TO WRITE PRIOR TO NEXT SECT

SYNC  
 SECTOR #  
 INTERLEAVE  
 NEW SECTOR  
 DONE?

NO, SECTOR IS GOOD  
 STA \$4B IF MODULO IS NEEDED



BF6E- A9 FF LDA ##FF Go WRITE SYNC'S FOR THIS SECTOR  
BF70- 4C E7 BE JMP \$BEE7

BF73- 48 17 PHA 3 (TIMING) END OF TRACK  
BF74- 68 PLA 4 CHECK IT  
BF75- A4 47 LDY 3 \$47 SYNC'S BETWEEN SECTORS  
BF77- BD 8D CO LDA 4 \$C08D, X SENSE WRITE PROTECT  
BF7A- BD 8E CO LDA \$C08E, X READ MODE  
BF7D- 30 34 BMI \$BFB3 ITS WRITE PROTECTED  
BF7F- 88 DEY

BF80- 48 PHA  
BF81- 68 PLA  
BF82- EA NOP  
BF83- EA NOP  
BF84- 24 00 BIT \$00 DELAY PAST 80  
BF86- 48 PHA SYNC'S TO INSURE  
BF87- 68 PLA ENOUGH SPACE BETWEEN  
BF88- 88 DEY LAST SECTOR AND FIRST

BF89- D0 F5 BNE \$BF80  
BF8B- 20 65 B9 JSR \$B965 Go READ SECTOR HDR  
BF8E- B0 04 BCS \$BF94 oops, NO GOOD  
BF90- A5 2D LDA \$2D WHAT SECTOR WAS READ?  
BF92- F0 0A BEQ \$BF9E 0 - OK

BF94- A4 47 LDY \$47 OTHERWISE, TRY LESS SYNC'S  
BF96- 88 DEY FOR A SHORTER TRACK  
BF97- C0 10 CPY ##10 BUT NO LESS THAN 16  
BF99- 90 18 BCC \$BFB3 OR ITS A BAD DRIVE  
BF9B- 4C AE BE JMP \$BEAE TRY TO FORMAT IT AGAIN

BF9E- E6 41 INC \$41 NEXT TRACK  
BFA0- A5 41 LDA \$41  
BFA2- C9 23 CMP ##23 DONE ALL TRACKS?  
BFA4- B0 12 BCS \$BFB8 YES

BFA6- 0A ASL NO, DOUBLE FOR TWO PHASE SEEK  
BFA7- 20 1E BA JSR \$BA1E AND MOVE ARM  
BFAA- A4 47 LDY \$47  
BFAC- C8 INY TRY MORE SYNC'S BETWEEN SECTORS  
BFAD- C8 INY

BFAE- 84 47 STY \$47  
BFB0- 4C AE BE JMP \$BEAE Go FORMAT NEW TRACK  
BFB3- A9 40 LDA ##40 DRIVE ERROR  
BFB5- 4C 29 BE JMP \$BE29 EXIT

BFB8- 4C 27 BE JMP \$BE27 NO ERRORS  
BFBB- 48 PHA 3 SAVE BYTE  
BFBC- 4A LSR 2 USING ODD BITS FIRST  
BFBD- 05 4A ORA \$4A OR IN NON USEFUL BITS  
BFBF- 9D 8D CO STA \$C08D, X WRITE FIRST NIBBLE  
BFC2- DD 8C CO CMP \$C08C, X  
BFC5- 68 PLA 4 GET BYTE  
BFC6- C1 00 CMP (\$00, X) (TIMING)  
BFC8- 09 AA 2 ORA ##AA OR IN NON USEFUL BITS  
BFCA- EA 2 NOP (TIMING)

BFCB- 48 PHA 3  
BFCC- 68 PLA 4  
BFCD- EA 2 NOP  
BFCE- 9D 8D CO STA \$C08D, X WRITE EVEN NIBBLE  
BFD1- DD 8C CO CMP \$C08C, X  
BFD4- 60 RTS

BFD5- E8 INX  
BFD6- F0 01 BEQ \$BFD9 } NOT IN PATCH AREA  
BFD8- 60 RTS USE (PATCH TO PREVIOUS RELEASE)  
BFD9- 4C DD A5 JMP \$A5DD SEE CONTACT #3 PG 3

BFDC- 8D 63 AA STA \$AA63  
BFDf- 8D 70 AA STA \$AA70 } SET ADDITIONAL DEFAULTS  
BFE2- 8D 71 AA STA \$AA71 (B=0) SEE #APER

DOS 3.3 PATCH

WRITE ONE NIBBLE ENTRY

40

WRITE ONE BYTE TO DISK

(TIMING CRITICAL)

BFE5-	60			RTS		
BFE6-	20	5B	A7	JSR	#A75B	RESET STATE/WARMSTART
BFE9-	8C	B7	AA	STY	#AAB7	RUN NOT INTERRUPTED NOW
BFEC-	60			RTS		
BFED-	20	7E	AE	JSR	#AE7E	SAVE FLOWA
BFF0-	AE	9B	B3	LDX	#B39B	} RESTORE STACK
BFF3-	9A			TXS		
BFF4-	20	16	A3	JSR	#A316	CLOSE ALL FILES
BFF7-	BA			TSX		} SAVE STACK AGAIN
BFF8-	8E	9B	B3	STX	#B39B	
BFFB-	A9	09		LDA	##09	} EXIT FIO RC=9 "DISK FULL"
BFFD-	4C	85	B3	JMP	#B385	

97  
SEE  
#AGDS  
SEE  
#B377

NOTES

1. It is not clear who uses the value at #3F4 created by first entry processing.
2. DOS 3.2 differs from 3.1 as follows:
  - ECHO INPUT prompts under MON O not MON I
  - BASIC used to be passed "ⓈⓈ", now "Ⓢ" only for intercepted lines.
  - Some commands may not be entered in direct mode (READ, WRITE, POSITION, OPEN, APPEND)
  - DOS now automatically relocates an FP program from RAM FP BASIC to ROM FP BASIC and vice versa when it is loaded.
  - Lower case is now supported when disk files are read with INPUT statements.
  - Some commands are allowed to create a new file, others will not. Originally, any reference to a file that didn't exist created one which might have to be deleted (LOAD X created then deleted X)
  - Contents of the X register at entry to the File Manager (FIO) indicates whether a new file may be created.
    - X=Ⓢ create new file
    - X=Ⓢ̄ file must already exist
  - References in FIO to DOS close subroutine require that all of DOS must be preserved



when using FIO.

- After entering the monitor and then warmstarting DOS, a 48:00 is no longer necessary.
- All of DOS patch area is gone
- RWTS now disables from interrupts while in operation.
- B (byte offset) is now 0 by default.

3. a CTL-D output or any KSWL call force an exit from write mode.

4. FIO write range of bytes. Length must be one less than that desired.

5. DOS 3.2 documentation was incorrect in several areas:

Pg. 130 Deleted file track ptr is stored as the last byte in file name.

"END MARK" is really high byte of sector count.

Pg. 132 30-33 are:

- 30 track now being allocated
- 31 direction of track allocation (+1, toward track 35 -1, toward track 0)

- 32 } NOT USED
- 33 }



6. Hidden restrictions. DOS can not handle more than 16 sectors per track or more than 50 tracks per disk.

7. The DOS version number is always at \$B3BE (or as appropriate to machine size). It is 02 for DOS 3.2.

8. All file type bit patterns are now defined:

00	Text	08	Sequential?
01	Integer	10	Random?
02	Applesoft	20	Applesoft? LOADable
04	Binary	40	B?

FUNCTION	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	+10	+11
OPEN	01		OPTIONAL RECORD LENGTH		V	D	S	FILE TYPE	↑ FILE NAME									
CLOSE	02																	
READ	03	DOS CODE	RECORD No.	BYTE OFFSET	RANGE LENGTH	RANGE LENGTH	RANGE LENGTH	1 DATA BYTE	ADDRESS									
WRITE	04																	
DELETE	05				V	D	S		↑ FILE NAME									
CATALOG	06																	
LOCK	07				V	D	S											
UNLOCK	08				V	D	S		↑ FILE NAME									
RENAME	09		↑ NEW NAME		V	D	S											
POSITION	0A		RECORD No.	BYTE OFFSET														
INIT	0B	DOS PAGE NO.			V	D	S											
VERIFY	0C				V	D	S		↑ FILE NAME									

RETURN CODE (OUTPUT PARAM)

↑ FIOWA (45 BYTES)

↑ T/S LIST  
BUFFER

↑ T/S LIST  
BUFFER

↑ T/S  
LIST

↑ 256  
BYTE  
DATA  
SECTOR  
BUFFER

↑ DATA  
BUFF

FIO FILE MANAGER PARM LIST FORMAT  
REQUIRED INPUTS

DOS 3.2 ZERO PAGE USAGE

```

0000-
0008-
0010-
0018-
0020-      00      00 00
0028-  D0 07 D0 07 E7 0B 12 FE
0030-      AA      00 07 C4
0038-  1B FD      3C 00 FF 00
0040-  00 00 00 98 2D 98 EF D8
0048-  00 E7 00 08 00 86
0050-
0058-
0060-      FD
0068-  FC FD FD      D2
0070-  FD      ED EC      ED
0078-
0080-
0088-
0090-
0098-
00A0-
00A8-      FD
00B0-  FF
00B8-
00C0-
00C8-      61 95 00 08
00D0-      02
00D8-  FD 03
00E0-
00E8-
00F0-
00F8-
    
```

```

24  CURSOR HORIZONTAL (DOS)
26,27  SECTOR-READ BUFFER ↑ (ROM)
      ‡ SCRATCH SPACE (RWTS)
28,29  BASL/BASH (DOS)
2A  SEGMENT MERGE COUNTER (ROM, BOOT)
      SCRATCH SPACE (RWTS)
2B  BOOT SLOT*16 (ROM)
      SCRATCH SPACE (RWTS)
2C  CHECKSUM FROM SECTOR HDR (RWTS)
2D  SECTOR # FROM SECTOR HDR (RWTS)
2E  TRACK # FROM SECTOR HDR (RWTS)
2F  VOL # FROM SECTOR HDR (RWTS)
33  PROMPT CHARACTER (DOS)
35  DRIVE # IN HIGH BIT (RWTS)
36,37  CSWL, CSWH (DOS)
38,39  KSWL, KSWH (DOS)
3C  WORKBYTE (ROM)
      MERGE WORKBYTE (BOOT)
      DEVICE CHAR. TABLE ↑ (RWTS)
3D  SECTOR # (ROM)
      DEVICE CHAR. TABLE ↑ (RWTS)
3E,3F  ↑ ROM SECTOR-READ SUBRTN (BOOT)
      BUFFER ↑ (RWTS)
40,41  DOS IMAGE ↑ (BOOT)
      FILE BUFFER ↑ (DOS)
      (741) FORMAT TRACK COUNTER (RWTS)
42,43  BUFFER ↑ (DOS)
44,45  NUMERIC OPERAND (DOS)
46,47  SCRATCH SPACE (RWTS)
48,49  IOB ↑ (RWTS)
4A,4B  INT BASIC LOMEM ↑ (DOS)
      FORMAT DISKETTE WORKSPACE (RWTS)
4C,4D  INT BASIC HIMEM ↑ (DOS)
67,68  FP BASIC PGM START (DOS)
69,6A  FP BASIC VARS START (DOS)
6F,70  FP STRING START (DOS)
73,74  FP HIMEM ↑ (DOS)
76  FP LINE NO. HIGH (DOS)
AF, B0  FP PGM END (DOS)
CA, CB  INT BASIC PGM ↑ (DOS)
CC, CD  INT BASIC VARS ↑ (DOS)
D6  FP PGM TOO BIG (DOS)
D8, D9  INT BASIC LINE NO. (DOS)
      FP BASIC ONERR (DOS)
    
```

DOS 3.2 FILE BUFFERS (TYPICAL)  
(\$9600-\$9CF8, 48K system)



DOS 3.2 FILE BUFFERS

```

9600- 9F 00 10 0A 00 4B 03 4D
9608- 36 B9 A8 03 03 6F B3 03
9610- 00 01 11 14 00 61 28 A0
9618- A0 C4 C9 D3 CB A0 C9 C9
9620- 29 47 01 1C 16 00 50 B1
9628- 0B 00 03 61 28 CD C1 D3
9630- D4 C5 D2 A0 C4 C9 D3 CB
9638- C5 D4 D4 C5 29 47 01 17
9640- 17 00 50 B2 1C 00 03 61
9648- 28 D6 C5 D2 D3 C9 CF CE
9650- A0 B3 AE B2 29 01 17 18
9658- 00 63 03 50 B3 1E 00 03
9660- 61 28 B1 B6 AD C6 C5 C2
9668- AD B7 B9 29 01 2F 1E 00
9670- 63 03 61 28 A0 A0 C3 CF
9678- D0 D9 D2 C9 C7 C8 D4 A0
9680- B1 B9 B7 B9 A0 A0 A0 C1
9688- D0 D0 CC C5 A0 C3 CF CD
9690- D0 D5 D4 C5 D2 A0 C9 CE
9698- C3 AE 29 01 05 28 00 51
96A0- 01 9F 00 00 00 00 00 00
96A8- 00 00 00 00 00 00 00 00
96B0- 00 00 00 00 00 00 00 00
96B8- 00 00 00 00 00 00 00 00
96C0- 00 00 00 00 00 00 00 00
96C8- 00 00 00 00 00 00 00 00
96D0- 00 00 00 00 00 00 00 00
96D8- 00 00 00 00 00 00 00 00
96E0- 00 00 00 00 00 00 00 00
96E8- 00 00 00 00 00 00 00 00
96F0- 00 00 00 00 00 00 00 00
96F8- 00 00 00 00 00 00 00 00
-----
9700- 00 00 00 00 00 00 00 00
9708- 00 00 00 00 12 0B 00 00
9710- 00 00 00 00 00 00 00 00
9718- 00 00 00 00 00 00 00 00
9720- 00 00 00 00 00 00 00 00
9728- 00 00 00 00 00 00 00 00
9730- 00 00 00 00 00 00 00 00
9738- 00 00 00 00 00 00 00 00
9740- 00 00 00 00 00 00 00 00
9748- 00 00 00 00 00 00 00 00
9750- 00 00 00 00 00 00 00 00
9758- 00 00 00 00 00 00 00 00
9760- 00 00 00 00 00 00 00 00
9768- 00 00 00 00 00 00 00 00
9770- 00 00 00 00 00 00 00 00
9778- 00 00 00 00 00 00 00 00
9780- 00 00 00 00 00 00 00 00
9788- 00 00 00 00 00 00 00 00
9790- 00 00 00 00 00 00 00 00
9798- 00 00 00 00 00 00 00 00
97A0- 00 00 00 00 00 00 00 00
97A8- 00 00 00 00 00 00 00 00
97B0- 00 00 00 00 00 00 00 00
97B8- 00 00 00 00 00 00 00 00
97C0- 00 00 00 00 00 00 00 00
97C8- 00 00 00 00 00 00 00 00
97D0- 00 00 00 00 00 00 00 00
97D8- 00 00 00 00 00 00 00 00

```

3RD BUFFER

(SEE BUFFER #1 FOR DETAILS)

DATA SECTOR BUFFER

T/S LIST BUFFER

97E0- 00 00 00 00 00 00 00 00  
 97E8- 00 00 00 00 00 00 00 00  
 97F0- 00 00 00 00 00 00 00 00  
 97F8- 00 00 00 00 00 00 00 00

---

9800- 12 0C 12 0C 00 12 0B 01

9808- 00 7A 00 00 00 7A 00 00

9810- 00 00 01 01 00 00 00 01

9818- 00 00 00 00 00 02 00 00

9820- 00 00 00 00 00 81 70 01

---

9828- FF 11 00 00 00 00 C5 CC

9830- CC CF A0 A0 A0 A0 A0 A0

9838- A0 A0 A0 A0 A0 A0 A0 A0

9840- A0 A0 A0 A0 A0 A0 A0 A0

---

9848- A0 A0 A0 00 98 00 97 00

---

9850- 96 00 00 00 FF FF 00 00

9858- FF FF 00 00 FF FF 00 00

9860- FF FF 00 00 FF FF 00 00

9868- FF FF 00 00 FF FF 00 00

9870- FF FF 00 00 FF FF 00 00

9878- FF FF 00 00 FF FF 00 00

9880- FF FF 00 00 FF FF 00 00

9888- FF FF 00 00 FF FF 00 00

9890- FF FF 00 00 FF FF 00 00

9898- FF FF 00 00 FF FF 00 00

98A0- FF FF 00 00 FF FF 00 00

98A8- FF FF 00 00 FF FF 00 00

98B0- FF FF 00 00 FF FF 00 00

98B8- FF FF 00 00 FF FF 00 00

98C0- FF FF 00 00 FF FF 00 00

98C8- FF FF 00 00 FF FF 00 00

98D0- FF FF 00 00 FF FF 00 00

98D8- FF FF 00 00 FF FF 00 00

98E0- FF FF 00 00 FF FF 00 00

98E8- FF FF 00 00 FF FF 00 00

98F0- FF FF 00 00 FF FF 00 00

98F8- FF FF 00 00 FF FF 00 00

9900- FF FF 00 00 FF FF 00 00

9908- FF FF 00 00 FF FF 00 00

9910- FF FF 00 00 FF FF 00 00

9918- FF FF 00 00 FF FF 00 00

9920- FF FF 00 00 FF FF 00 00

9928- FF FF 00 00 FF FF 00 00

9930- FF FF 00 00 FF FF 00 00

9938- FF FF 00 00 FF FF 00 00

9940- FF FF 00 00 FF FF 00 00

9948- FF FF 00 00 FF FF 00 00

---

9950- FF FF 00 00 FF FF 00 00

9958- FF FF 00 00 FF FF 00 00

9960- FF FF 00 00 FF FF 00 00

9968- FF FF 00 00 FF FF 00 00

9970- FF FF 00 00 FF FF 00 00

9978- FF FF 00 00 FF FF 00 00

9980- FF FF 00 00 FF FF 00 00

9988- FF FF 00 00 FF FF 00 00

9990- FF FF 00 00 FF FF 00 00

9998- FF FF 00 00 FF FF 00 00

99A0- FF FF 00 00 FF FF 00 00

99A8- FF FF 00 00 FF FF 00 00

99B0- FF FF 00 00 FF FF 00 00

99B8- FF FF 00 00 FF FF 00 00

99C0- FF FF 00 00 FF FF 00 00

99C8- FF FF 00 00 FF FF 00 00

99D0- FF FF 00 00 FF FF 00 00

FIO WORKAREA IMAGE

FILENAME

LINK PTRS

2ND BUFFER

DATA SECTOR  
 BUFFER

T/S LIST  
 BUFFER

```

99D8- FF FF 00 00 FF FF 00 00
99E0- FF FF 00 00 FF FF 00 00
99E8- FF FF 00 00 FF FF 00 00
99F0- FF FF 00 00 FF FF 00 00
99F8- FF FF 00 00 FF FF 00 00
9A00- FF FF 00 00 FF FF 00 00
9A08- FF FF 00 00 FF FF 00 00
9A10- FF FF 00 00 FF FF 00 00
9A18- FF FF 00 00 FF FF 00 00
9A20- FF FF 00 00 FF FF 00 00
9A28- FF FF 00 00 FF FF 00 00
9A30- FF FF 00 00 FF FF 00 00
9A38- FF FF 00 00 FF FF 00 00
9A40- FF FF 00 00 FF FF 00 00
9A48- FF FF 00 00 FF FF 00 00
9A50- FF FF 00 00 FF FF 00 00

```

```

9A58- FF FF 00 00 FF FF 00 00
9A60- FF FF 00 00 FF FF 00 00
9A68- FF FF 00 00 FF FF 00 00
9A70- FF FF 00 00 FF FF 00 00
9A78- FF FF 00 00 FF FF 00 00

```

FIO WORKAREA IMAGE

```

9A80- 00 FF 00 00 FF FF 00 00
9A88- FF FF 00 00 FF FF 00 00
9A90- FF FF 00 00 FF FF 00 00
9A98- FF FF 00 00 FF FF 53 9A

```

FILENAME

LINK POINTERS

```

9AA0- 53 99 53 98 2D 98 00 00
9AA8- FF FF 00 00 FF FF 00 00
9AB0- FF FF 00 00 FF FF 00 00
9AB8- FF FF 00 00 FF FF 00 00
9AC0- FF FF 00 00 FF FF 00 00
9AC8- FF FF 00 00 FF FF 00 00
9AD0- FF FF 00 00 FF FF 00 00
9AD8- FF FF 00 00 FF FF 00 00
9AE0- FF FF 00 00 FF FF 00 00
9AE8- FF FF 00 00 FF FF 00 00
9AF0- FF FF 00 00 FF FF 00 00
9AF8- FF FF 00 00 FF FF 00 00
9B00- FF FF 00 00 FF FF 00 00
9B08- FF FF 00 00 FF FF 00 00
9B10- FF FF 00 00 FF FF 00 00
9B18- FF FF 00 00 FF FF 00 00
9B20- FF FF 00 00 FF FF 00 00
9B28- FF FF 00 00 FF FF 00 00
9B30- FF FF 00 00 FF FF 00 00
9B38- FF FF 00 00 FF FF 00 00
9B40- FF FF 00 00 FF FF 00 00
9B48- FF FF 00 00 FF FF 00 00
9B50- FF FF 00 00 FF FF 00 00
9B58- FF FF 00 00 FF FF 00 00
9B60- FF FF 00 00 FF FF 00 00
9B68- FF FF 00 00 FF FF 00 00
9B70- FF FF 00 00 FF FF 00 00
9B78- FF FF 00 00 FF FF 00 00
9B80- FF FF 00 00 FF FF 00 00
9B88- FF FF 00 00 FF FF 00 00
9B90- FF FF 00 00 FF FF 00 00
9B98- FF FF 00 00 FF FF 00 00
9BA0- FF FF 00 00 FF FF 00 00

```

1ST BUFFER

CURRENT  
DATA  
SECTOR  
BUFFER

```

9BA8- FF FF 00 00 FF FF 00 00
9BB0- FF FF 00 00 FF FF 00 00
9BB8- FF FF 00 00 FF FF 00 00
9BC0- FF FF 00 00 FF FF 00 00
9BC8- FF FF 00 00 FF FF 00 00

```

9BD0- FF FF 00 00 FF FF 00 00  
 9BD8- FF FF 00 00 FF FF 00 00  
 9BE0- FF FF 00 00 FF FF 00 00  
 9BE8- FF FF 00 00 FF FF 00 00  
 9BF0- FF FF 00 00 FF FF 00 00  
 9BF8- FF FF 00 00 FF FF 00 00  
 9C00- FF FF 00 00 FF FF 00 00  
 9C08- FF FF 00 00 FF FF 00 00  
 9C10- FF FF 00 00 FF FF 00 00  
 9C18- FF FF 00 00 FF FF 00 00  
 9C20- FF FF 00 00 FF FF 00 00  
 9C28- FF FF 00 00 FF FF 00 00  
 9C30- FF FF 00 00 FF FF 00 00  
 9C38- FF FF 00 00 FF FF 00 00  
 9C40- FF FF 00 00 FF FF 00 00  
 9C48- FF FF 00 00 FF FF 00 00  
 9C50- FF FF 00 00 FF FF 00 00  
 9C58- FF FF 00 00 FF FF 00 00  
 9C60- FF FF 00 00 FF FF 00 00  
 9C68- FF FF 00 00 FF FF 00 00  
 9C70- FF FF 00 00 FF FF 00 00  
 9C78- FF FF 00 00 FF FF 00 00  
 9C80- FF FF 00 00 FF FF 00 00  
 9C88- FF FF 00 00 FF FF 00 00  
 9C90- FF FF 00 00 FF FF 00 00  
 9C98- FF FF 00 00 FF FF 00 00

CURRENT  
 TRACK/SECTOR  
 LIST  
 BUFFER

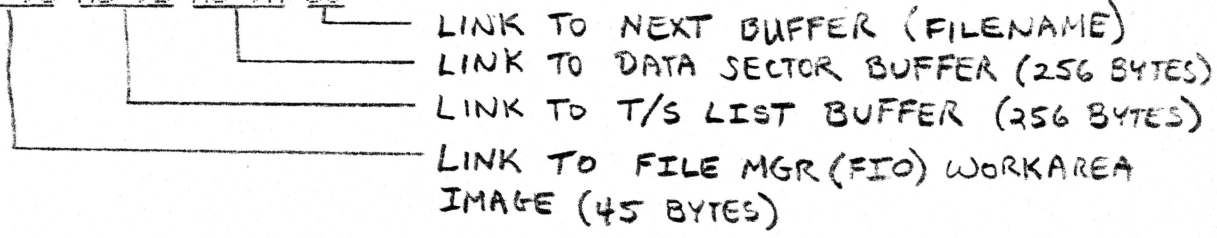
9CA0- FF FF 00 00 FF FF 00 00  
 9CA8- FF FF 00 00 FF FF 00 00  
 9CB0- FF FF 00 00 FF FF 00 00  
 9CB8- FF FF 00 00 FF FF 00 00  
 9CC0- FF FF 00 00 FF FF 00 00  
 9CC8- FF FF 00 00 FF FF 00 00

FIO WORKAREA IMAGE

9CD0- FF FF 00 00 FF FF 00 00  
 9CD8- FF FF 00 00 FF FF 00 00  
 9CE0- FF FF 00 00 FF FF 00 00  
 9CE8- FF FF 00 00 FF FF 00 00

FILENAME  
 [1ST BYTE = #00 MEANS CLOSED]

9CF0- FF A6 9C A6 9B A6 9A 80  
 9CF8- 9A





DOS 3.2 UPDATE 3.2 PROGRAM

(A\$800,L\$700)

```

0800- A9 0C LDA #00
0802- 20 05 09 JSR $0905
0805- 20 23 09 JSR $0923
0808- A9 00 LDA #00
080A- 20 E3 08 JSR $08E3
080D- 20 89 09 JSR $0989
0810- 90 1B BCC $082D
0812- A9 08 LDA #08
0814- 20 05 09 JSR $0905
0817- AD 65 0E LDA $0E65
081A- C9 02 CMP #02
081C- D0 07 BNE $0825
081E- AD 66 0E LDA $0E66
0821- C9 0B CMP #0B
0823- F0 21 BEQ $0846
0825- A9 07 LDA #07
0827- 20 E3 08 JSR $08E3
082A- 4C B5 08 JMP $08B5
082D- AD 06 1E LDA $1E06
0830- 4D 0D 1E EOR $1E0D
0833- 49 6E EOR #6E
0835- D0 0F BNE $0846
0837- AD BE 34 LDA $34BE
083A- 49 02 EOR #02
083C- D0 08 BNE $0846
083E- 8D 66 2B STA $2B66
0841- 8D EB 13 STA $13EB
0844- F0 08 BEQ $084E
0846- A9 08 LDA #08
0848- 20 E3 08 JSR $08E3
084B- 4C B5 08 JMP $08B5
084E- A9 0A LDA #0A
0850- 20 05 09 JSR $0905
0853- A9 01 LDA #01
0855- 20 E3 08 JSR $08E3
0858- 20 6A FD JSR $FD6A
085B- 20 48 09 JSR $0948
085E- B0 EE BCS $084E
0860- A0 1D LDY #1D
0862- B9 2E 0E LDA $0E2E,Y
0865- 99 75 2B STA $2B75,Y
0868- 88 DEY
0869- 10 F7 BPL $0862
086B- A9 02 LDA #02
086D- 20 05 09 JSR $0905
0870- A9 02 LDA #02
0872- 20 E3 08 JSR $08E3
0875- 20 0D 09 JSR $090D
0878- A9 03 LDA #03
087A- 20 E3 08 JSR $08E3
087D- 20 39 09 JSR $0939
0880- B0 CC BCS $084E
0882- 20 85 09 JSR $0985
0885- 90 4A BCC $08D1
0887- A9 0A LDA #0A
0889- 20 05 09 JSR $0905
088C- AD 6E 0E LDA $0E6E
088F- 0A ASL
0890- B0 03 BCS $0895
0892- 0A ASL
0893- 90 04 BCC $0899
    
```

HOME, VTAB 12  
 INIT COUT/CSWL  
 DISPLAY MSG #0 ("LOADING...")  
 READ DOS IMAGE T/S 04-2/A TO \$1200-\$3600  
 IT WORKED  
 ERROR  
 HOME, VTAB 8  
 GET TRK  
 2?  
 NO, 0, 1  
 YES, GET SEC  
 SEC 8?  
 YES  
 NO, MSG #7 ("UNABLE TO READ...")  
 SAY SO, ERROR BEFORE END

TRACK	SEC	DOSADDR	BUFADR
0	0	300	1200
0	1	3000	1300
0	9	3F00	1800
0	A	1B00	1C00
2	D	3500	3600

EXIT  
 OK READ, GET TRACK 0, SECTOR C, OFFSET  
 DOS ADDR 1D06 XOR 1D0D (2A75 XOR 1B00)  
 =6E  
 NO, NOT RIGHT DOS  
 YES, GET TRACK 2, SECTOR B, OFFS#BE  
 DOS LOC \$33BE=2 (VTOC DOS RELEASE #)  
 NO, WRONG RELEASE DOS  
 YES, CLEAR T1, SC, OFF 66 (2A66, VOLUME)  
 TO, S1, OFF EB (36EB + 0)  
 UNCOND BRANCH  
 MSG #8 ("IMAGE NOT AVAIL...")  
 SAY IT  
 QUIT  
 EXIT  
 HOME, VTAB 10, ALL IS WELL, GET NAME  
 MSG #1 ("INPUT NAME...")  
 GET NAME  
 PROCESS IT  
 ERROR, REASK  
 COPY 30 BYTES  
 TO TRACK 1 SECTOR C OFFSET 75  
 (DOS ADDR 2A75 FIRST FILE)  
 HOME VTAB 2  
 MSG #2 ("REMEMBER...")  
 ECHO NAME  
 MSG #3 ("INSERT... [RETURN]... [ESC]...")  
 WAIT  
 ESC, GET NAME  
 WRITE DOS IMAGE  
 OK, TRY ANOTHER  
 VTAB 10  
 GET STATUS  
 READ ERROR \$80  
 VOLUME MIS/PROTECTED \$20/\$10

ADZU

```

0895- A9 09 LDA #09
0897- D0 02 BNE $089B
0899- A9 0A LDA #0A
089B- 20 E3 08 JSR $08E3
089E- A9 05 LDA #05
08A0- 20 E3 08 JSR $08E3
08A3- 20 39 09 JSR $0939
08A6- B0 08 BCS $08B0
08A8- A9 0A LDA #0A
08AA- 20 05 09 JSR $0905
08AD- 4C 6B 08 JMP $086B
08B0- A9 0A LDA #0A
08B2- 20 05 09 JSR $0905
08B5- A9 06 LDA #06
08B7- 20 E3 08 JSR $08E3
08BA- 20 39 09 JSR $0939
08BD- B0 FB BCS $08BA
08BF- AD 62 0E LDA $0E62
08C2- 4A LSR
08C3- 4A LSR
08C4- 4A LSR
08C5- 4A LSR
08C6- 09 C0 ORA #C0
08C8- 85 F1 STA $F1
08CA- A9 00 LDA #00
08CC- 85 F0 STA $F0
08CE- 6C F0 00 JMP ($00F0)
08D1- A9 06 LDA #06
08D3- 20 05 09 JSR $0905
08D6- A9 04 LDA #04
08D8- 20 E3 08 JSR $08E3
08DB- 20 39 09 JSR $0939
08DE- B0 D5 BCS $08B5
08E0- 4C 4E 08 JMP $084E

```

```

MSG #9 ("UNABLE TO WRITE...")
MSG #4 ("PROTECTED...")
MSG #5 ("RETRY...")
WAIT
ESC, QUIT
RET, RETRY, VTAB 10
RETRY
VTAB 10 RETRY
MSG #6 ("... REBOOT") ← EXIT
WAIT FOR <CR> (NOT <ESC>)
LAST SLOT
} SHIFT S0 TO 05
} F0, F1 → CTRL ROM
BOOT CS00
VTAB 6
"DID IT" MSG #4
WAIT
ESC, REBOOT
RET, GET NAME

```

112

```

08E3- 0A ASL *Z INDEX
08E4- AA TAX
08E5- BD E2 09 LDA $09E2,X
08E8- 85 F0 STA $F0
08EA- BD E3 09 LDA $09E3,X
08ED- 85 F1 STA $F1
08EF- A0 00 LDY #00
08F1- B1 F0 LDA ($F0),Y
08F3- 48 PHA
08F4- 09 80 ORA #80
08F6- 20 ED FD JSR $FDED
08F9- 68 PLA
08FA- 30 08 BMI $0904
08FC- E6 F0 INC $F0
08FE- D0 EF BNE $08EF
0900- E6 F1 INC $F1
0902- D0 EB BNE $08EF
0904- 60 RTS

```

```

MSG DISP
A = #msg
GET ADDR L
ADDR H
F0, F1 = MSGADDR
0 INDEX
GET BYTE
SAVE FOR END TEST
MAKE NORMAL VIDED
COUT
GET
DONE
NEXT BYTE, MORE TO DO TILL HI BITON

```

```

0905- 48 PHA
0906- 20 58 FC JSR $FC58
0909- 68 PLA
090A- 85 25 STA $25
090C- 60 RTS

```

```

HOME
A: set to CV

```

```

090D- A9 A0 LDA #A0
090F- A2 00 LDX #00
0911- 20 ED FD JSR $FDED
0914- E0 1E CPX #1E
0916- F0 06 BEQ $091E
0918- BD 2E 0E LDA $0E2E,X

```

```

ECHO FILE NAME
OUTPUT 5
30 LEN
YES, CR
NO, NEXT NAME CTRZ

```

```

091B- E8      INX
091C- D0 F3   BNE  $0911
091E- A9 8D   LDA  ##8D
0920- 4C ED FD JMP  $FDED
0923- A9 F0   LDA  ##F0
0925- 85 36   STA  #36
0927- A9 1B   LDA  ##1B
0929- 85 38   STA  #38
092B- A9 FD   LDA  ##FD
092D- 85 37   STA  #37
092F- 85 39   STA  #39
0931- 8D F4 03 STA  $03F4
0934- A9 87   LDA  ##87
0936- 85 33   STA  #33
0938- 60      RTS

```

OUTPUT  
QUIT WITH (CR)

INIT COUT/CSWL

COUT (36,37) ← FDF0 ("COUT1")  
CSWL (38,39) ← FD1B ("KEYIN")

PUT #FD AT 3F4  
PROMPT= BELL (14)

```

0939- 20 0C FD JSR  $FD0C
093C- C9 8D   CMP  ##8D
093E- F0 06   BEQ  $0946
0940- C9 9B   CMP  ##9B
0942- F0 03   BEQ  $0947
0944- D0 F3   BNE  $0939
0946- 18      CLC
0947- 60      RTS

```

RET/ESC

(CR) YES, OK  
(ESC) YES  
NO, WAIT

CR: CARRY CLEAR  
ESC: CARRY SET

```

0948- 8A      TXA
0949- F0 2C   BEQ  $0977
094B- 86 F0   STX  $F0
094D- A0 00   LDY  ##00
094F- B9 00 02 LDA  $0200, Y
0952- C8      INY
0953- C9 A0   CMP  ##A0
0955- F0 F8   BEQ  $094F
0957- C9 C1   CMP  ##C1
0959- 90 1C   BCC  $0977
095B- A2 00   LDX  ##00
095D- 9D 2E 0E STA  $0E2E, X
0960- E8      INX
0961- E0 1E   CPX  ##1E
0963- F0 1E   BEQ  $0983
0965- B9 00 02 LDA  $0200, Y
0968- C9 8D   CMP  ##8D
096A- F0 0D   BEQ  $0979
096C- C9 AC   CMP  ##AC
096E- F0 09   BEQ  $0979
0970- C4 F0   CPY  $F0
0972- F0 05   BEQ  $0979
0974- C8      INY
0975- D0 E6   BNE  $095D
0977- 38      SEC
0978- 60      RTS
0979- A9 A0   LDA  ##A0
097B- 9D 2E 0E STA  $0E2E, X
097E- E8      INX
097F- E0 1E   CPX  ##1E
0981- D0 F8   BNE  $097B
0983- 18      CLC
0984- 60      RTS

```

PROCESS FILE NAME

NULL  
YES, ERROR  
SAVE LEN

GET BYTE  
SKIP  
'5'  
YES, SKIP TO 12<sup>th</sup> NON-BLANK  
alpha? 'A'  
<A, error

TO COPY BUF .EZE (FILENAME)  
NEXT  
30? MAX?  
YES, OK, QUIT  
NO, NEXT  
CR THERE?  
YES, BLANK REST BUF, QUIT OK  
NO, ',' COMMA?  
YES, TREAT AS CR  
NO, END OF INPUT BUF (@ LEN)  
YES, TREAT AT CR  
NO, NEXT

ERROR  
QUIT

BLANK TRAIL OF BUF ON CR

TO 30

OK  
QUIT

```

0985- A9 02   LDA  #$02
0987- D0 02   BNE  $098B

```

WRITE

READ/WRITE DOS IMAGE

```

0989- A9 01   LDA  #$01
098B- 20 B3 09 JSR  $09B3
098E- A9 0E   LDA  #$0E
0990- A0 61   LDY  #$61
0992- 20 D9 03 JSR  $03D9

```

READ

SET UP IOB (2)

POINT TO IOB E61

CALL RWTS

TRACK 0: 1200-1ED0  
TRACK 1: 1F00-2BD0  
TRACK 2: 2C00-3600 (TO SEC A)



0995-	B0 1B	BCS	\$09B2	ERROR, QUIT CARRY
0997-	AC 66 0E	LDY	\$0E66	OK, GET SEC
099A-	CS	INY		NEXT
099B-	C0 0D	CPY	##0D	13?
099D-	D0 05	BNE	\$09A4	NO, OK
099F-	A0 00	LDY	##00	YES, S=0
09A1-	EE 65 0E	INC	\$0E65	NEXT TRK
09A4-	8C 66 0E	STY	\$0E66	NEXT SECTOR
09A7-	EE 6A 0E	INC	\$0E6A	NEXT BUF PAGE, 1200-36FF
09AA-	AD 6A 0E	LDA	\$0E6A	
09AD-	C9 37	CMP	##37	DONE?
09AF-	D0 DD	BNE	\$098E	NO
09B1-	18	CLC		YES, NO ERROR
09B2-	60	RTS		

09B3-	48	PHA		SAVE CMD TYPE
09B4-	C9 01	CMP	##01	READ?
09B6-	D0 1A	BNE	\$09D2	NO, WRITE
09B8-	20 E3 03	JSR	\$03E3	YES, GET RWTS TAB ADDR (JOB)
09BB-	85 F1	STA	#F1	
09BD-	84 F0	STY	#F0	F0, F1 → RWTS JOB
09BF-	A0 01	LDY	##01	OFFSET SLOT
09C1-	B1 F0	LDA	(#F0), Y	GET SLOT * 16
09C3-	8D 4D 0E	STA	\$0E4D	SAVE SLOT * 16 IN JOB 1
09C6-	8D 5B 0E	STA	\$0E5B	OFFSET DRIVE
09C9-	C8	INY		DRIVE
09CA-	B1 F0	LDA	(#F0), Y	SAVE DRIVE IN JOB 1
09CC-	8D 4E 0E	STA	\$0E4E	
09CF-	8D 5C 0E	STA	\$0E5C	
09D2-	A0 11	LDY	##11	LENGTH
09D4-	B9 4C 0E	LDA	\$0E4C, Y	} COPY JOB 1 → JOB 2
09D7-	99 61 0E	STA	\$0E61, Y	
09DA-	88	DEY		
09DB-	10 F7	BPL	\$09D4	
09DD-	68	PLA		GET CMD
09DE-	8D 6D 0E	STA	\$0E6D	SET TO JOB 2
09E1-	60	RTS		

INIT JOB (2)

MSG TABLE

09E2-	F8	SED		
09E3-	09 87	ORA	##87	
09E5-	0A	ASL		A:
09E6-	B9 0A 60	LDA	\$600A, Y	0: 9F8
09E9-	0B	???		1: A87
09EA-	EC 0B 96	CPX	\$960B	2: AB9
09ED-	0C	???		3: B60
09EE-	E8	INX		4: BEC
09EF-	0C	???		5: C96
09F0-	25 0D	AND	\$0D	6: CEB
09F2-	42	???		7: D25
09F3-	0D 8B 0D	ORA	\$0D8B	
09F6-	F0 0D	BEQ	\$0A05	

MSG#0

MSG5

09F8-	0D 20 20	ORA	\$2020	
09FB-	20 20 44	JSR	\$4420	
09FE-	4F	???		
09FF-	53	???		
0A00-	20 33 2E	JSR	\$2E33	<CR> 6666 DO S 63.26 MASTER 6 - 6 UPDATE 6
0A03-	32	???		UTILITY <CR> <CR>
0A04-	20 4D 41	JSR	\$414D	66 COPYRIGHT 6 1979 6 BY 6
0A07-	53	???		APPLE 6 COMPUTER 6 INC. <CR>
0A08-	54	???		(10) 6 ALL 6 RIGHTS 6 RESERVED.
0A09-	45 52	EOR	\$52	(4) <CR> (8) 6 (NOW 6
0A0B-	20 2D 20	JSR	\$202D	LOADING 6 DOS 6 IMAGE) <CR>
0A0E-	55 50	EOR	\$50, X	
0A10-	44	???		
0A11-	41 54	EOR	(\$54, X)	

0A13-	45	20	EOR	\$20
0A15-	55	54	EOR	\$54, X
0A17-	49	4C	EOR	##4C
0A19-	49	54	EOR	##54
0A1B-	59	0D 0D	EOR	\$0D0D, Y
0A1E-	20	20 43	JSR	\$4320
0A21-	4F		???	
0A22-	50	59	BVC	\$0A7D
0A24-	52		???	
0A25-	49	47	EOR	##47
0A27-	48		PHA	
0A28-	54		???	
0A29-	20	31 39	JSR	\$3931
0A2C-	37		???	
0A2D-	39	20 42	AND	\$4220, Y
0A30-	59	20 41	EOR	\$4120, Y
0A33-	50	50	BVC	\$0A85
0A35-	4C	45 20	JMP	\$2045
0A38-	43		???	
0A39-	4F		???	
0A3A-	4D	50 55	EOR	\$5550
0A3D-	54		???	
0A3E-	45	52	EOR	\$52
0A40-	20	49 4E	JSR	\$4E49
0A43-	43		???	
0A44-	0D	20 20	ORA	\$2020
0A47-	20	20 20	JSR	\$2020
0A4A-	20	20 20	JSR	\$2020
0A4D-	20	20 41	JSR	\$4120
0A50-	4C	4C 20	JMP	\$204C
0A53-	52		???	
0A54-	49	47	EOR	##47
0A56-	48		PHA	
0A57-	54		???	
0A58-	53		???	
0A59-	20	52 45	JSR	\$4552
0A5C-	53		???	
0A5D-	45	52	EOR	\$52
0A5F-	56	45	LSR	\$45, X
0A61-	44		???	
0A62-	2E	0D 0D	ROL	\$0D0D
0A65-	0D	0D 20	ORA	\$200D
0A68-	20	20 20	JSR	\$2020
0A6B-	20	20 20	JSR	\$2020
0A6E-	20	28 4E	JSR	\$4E28
0A71-	4F		???	
0A72-	57		???	
0A73-	20	4C 4F	JSR	\$4F4C
0A76-	41	44	EOR	(\$44, X)
0A78-	49	4E	EOR	##4E
0A7A-	47		???	
0A7B-	20	44 4F	JSR	\$4F44
0A7E-	53		???	
0A7F-	20	49 4D	JSR	\$4D49
0A82-	41	47	EOR	(\$47, X)
0A84-	45	29	EOR	\$29
0A86-	8D	0D 50	STA	\$500D
0A89-	4C	45 41	JMP	\$4145
0A8C-	53		???	
0A8D-	45	20	EOR	\$20
0A8F-	49	4E	EOR	##4E
0A91-	50	55	BVC	\$0A85
0A93-	54		???	

MSG #1

<CR> PLEASE INPUT TO THE  
 "GREETING" & PROGRAM'S <CR>  
 FILE & NAME: &

0A94-	20	54	48	JSR	\$4854
0A97-	45	20		EOR	\$20
0A99-	22			???	
0A9A-	47			???	
0A9B-	52			???	
0A9C-	45	45		EOR	\$45
0A9E-	54			???	
0A9F-	49	4E		EOR	##4E
0AA1-	47			???	
0AA2-	22			???	
0AA3-	20	50	52	JSR	\$5250
0AA6-	4F			???	
0AA7-	47			???	
0AA8-	52			???	
0AA9-	41	4D		EOR	(\$4D, X)
0AAB-	27			???	
0AAC-	53			???	
0AAD-	0D	46	49	ORA	\$4946
0AB0-	4C	45	20	JMP	\$2045
0AB3-	4E	41	4D	LSR	\$4D41
0AB6-	45	3A		EOR	\$3A
0AB8-	A0	0D		LDY	##0D
0ABA-	0D	52	45	ORA	\$4552
0ABD-	4D	45	4D	EOR	\$4D45
0AC0-	42			???	
0AC1-	45	52		EOR	\$52
0AC3-	20	54	48	JSR	\$4854
0AC6-	41	54		EOR	(\$54, X)
0AC8-	20	22	55	JSR	\$5522
0ACB-	50	44		BVC	\$0B11
0ACD-	41	54		EOR	(\$54, X)
0ACF-	45	22		EOR	\$22
0AD1-	20	44	4F	JSR	\$4F44
0AD4-	45	53		EOR	\$53
0AD6-	20	4E	4F	JSR	\$4F4E
0AD9-	54			???	
0ADA-	20	43	52	JSR	\$5243
0ADD-	45	41		EOR	\$41
0ADF-	54			???	
0AE0-	45	0D		EOR	\$0D
0AE2-	54			???	
0AE3-	48			PHA	
0AE4-	45	20		EOR	\$20
0AE6-	22			???	
0AE7-	47			???	
0AE8-	52			???	
0AE9-	45	45		EOR	\$45
0AEB-	54			???	
0AEC-	49	4E		EOR	##4E
0AEE-	47			???	
0AEF-	22			???	
0AF0-	20	50	52	JSR	\$5250
0AF3-	4F			???	
0AF4-	47			???	
0AF5-	52			???	
0AF6-	41	4D		EOR	(\$4D, X)
0AF8-	2C	20	4F	BIT	\$4F20
0AFB-	52			???	
0AFC-	20	50	4C	JSR	\$4C50
0AFF-	41	43		EOR	(\$43, X)
0B01-	45	20		EOR	\$20
0B03-	49	54		EOR	##54
0B05-	20	49	4E	JSR	\$4E49

MSG #2

<CR><CR> REMEMBER THAT  
 IS "UPDATE" DOES NOT IS  
 CREATE <CR> THE IS "GREETING" IS  
 PROGRAM, IS OR IS PLACE IS IN <CR>  
 THE IS DISK IS DIRECTORY <CR><CR>  
 IS THIS IS IS THE IS FILE IS  
 NAME IS THAT IS WILL IS BE <CR>  
 PLACE IS WITH IN IS THE IS  
 IMAGE; <CR><CR> bbbb

OB08-	0D 54 48	ORA	\$4854
OB0B-	45 20	EOR	\$20
OB0D-	44	???	
OB0E-	49 53	EOR	##53
OB10-	4B	???	
OB11-	20 44 49	JSR	\$4944
OB14-	52	???	
OB15-	45 43	EOR	\$43
OB17-	54	???	
OB18-	4F	???	
OB19-	52	???	
OB1A-	59 0D 0D	EOR	\$0D0D, Y
OB1D-	20 20 54	JSR	\$5420
OB20-	48	PHA	
OB21-	49 53	EOR	##53
OB23-	20 49 53	JSR	\$5349
OB26-	20 54 48	JSR	\$4854
OB29-	45 20	EOR	\$20
OB2B-	46 49	LSR	\$49
OB2D-	4C 45 20	JMP	\$2045
OB30-	4E 41 4D	LSR	\$4D41
OB33-	45 20	EOR	\$20
OB35-	54	???	
OB36-	48	PHA	
OB37-	41 54	EOR	(\$54, X)
OB39-	20 57 49	JSR	\$4957
OB3C-	4C 4C 20	JMP	\$204C
OB3F-	42	???	
OB40-	45 0D	EOR	\$0D
OB42-	50 4C	BVC	\$0B90
OB44-	41 43	EOR	(\$43, X)
OB46-	45 44	EOR	\$44
OB48-	20 57 49	JSR	\$4957
OB4B-	54	???	
OB4C-	48	PHA	
OB4D-	49 4E	EOR	##4E
OB4F-	20 54 48	JSR	\$4854
OB52-	45 20	EOR	\$20
OB54-	49 4D	EOR	##4D
OB56-	41 47	EOR	(\$47, X)
OB58-	45 3A	EOR	\$3A
OB5A-	0D 0D 20	ORA	\$200D
OB5D-	20 20 A0	JSR	\$A020
<hr/>			
OB60-	0D 20 20	ORA	\$2020
OB63-	50 4C	BVC	\$0BB1
OB65-	41 43	EOR	(\$43, X)
OB67-	45 20	EOR	\$20
OB69-	54	???	
OB6A-	48	PHA	
OB6B-	45 20	EOR	\$20
OB6D-	44	???	
OB6E-	49 53	EOR	##53
OB70-	4B	???	
OB71-	45 54	EOR	\$54
OB73-	54	???	
OB74-	45 20	EOR	\$20
OB76-	54	???	
OB77-	4F	???	
OB78-	20 42 45	JSR	\$4542
OB7B-	20 22 55	JSR	\$5522
OB7E-	50 44	BVC	\$0BC4
OB80-	41 54	EOR	(\$54, X)
OB82-	45 44	EOR	\$44

MSG#3

<CR> to PLACE the diskette  
to to be "UPDATED" in <CR>  
THE disk drive, <CR><CR>  
to PRESS [RETURN] when  
READY <CR><CR> NOTE: IF YOU  
WANT a DIFFERENT FILE  
NAME, PRESS [ESC]; <CR>



OB84-	22		???	
OB85-	20	49	4E	JSR \$4E49
OB88-	0D	54	48	ORA \$4854
OB8B-	45	20		EOR \$20
OB8D-	44			???
OB8E-	49	53		EOR ##53
OB90-	4B			???
OB91-	20	44	52	JSR \$5244
OB94-	49	56		EOR ##56
OB96-	45	2E		EOR \$2E
OB98-	0D	0D	20	ORA \$200D
OB9B-	20	50	52	JSR \$5250
OB9E-	45	53		EOR \$53
OBA0-	53			???
OBA1-	20	5B	52	JSR \$525B
OBA4-	45	54		EOR \$54
OBA6-	55	52		EOR \$52, X
OBA8-	4E	5D	20	LSR \$205D
OBAB-	57			???
OBAC-	48			PHA
OBAD-	45	4E		EOR \$4E
OBAF-	20	52	45	JSR \$4552
OB2-	41	44		EOR (\$44, X)
OB4-	59	0D	0D	EOR \$0D0D, Y
OB7-	4E	4F	54	LSR \$544F
OB8A-	45	3A		EOR \$3A
OB8C-	20	49	46	JSR \$4649
OB8F-	20	59	4F	JSR \$4F59
OB2-	55	20		EOR \$20, X
OB4-	57			???
OB5-	41	4E		EOR (\$4E, X)
OB7-	54			???
OB8-	20	41	20	JSR \$2041
OB8B-	44			???
OB8C-	49	46		EOR ##46
OB8E-	46	45		LSR \$45
OB80-	52			???
OB81-	45	4E		EOR \$4E
OB83-	54			???
OB84-	20	46	49	JSR \$4946
OB87-	4C	45	20	JMP \$2045
OB8A-	4E	41	4D	LSR \$4D41
OB8D-	45	2C		EOR \$2C
OB8F-	50	52		BVC \$0C33
OB81-	45	53		EOR \$53
OB83-	53			???
OB84-	20	5B	45	JSR \$455B
OB87-	53			???
OB88-	43			???
OB89-	5D	2E	8D	EOR \$8D2E, X
OB8C-	0D	20	20	ORA \$2020
OB8F-	54			???
OB80-	48			PHA
OB81-	45	20		EOR \$20
OB83-	44			???
OB84-	49	53		EOR ##53
OB86-	4B			???
OB87-	45	54		EOR \$54
OB89-	54			???
OB8A-	45	20		EOR \$20
OB8C-	48			PHA
OB8D-	41	53		EOR (\$53, X)
OB8F-	20	42	45	JSR \$4542

MSG #4

(CR) TO THE DISKETTE HAS BEEN  
 UPDATED, & YOU MAY REMOVE IT  
 AT THIS TIME (CR) (CR) IF  
 YOU WISH TO "UPDATE"  
 ANOTHER DISK - (CR)  
 ETTE, PRESS [RETURN]. (CR) (CR)  
 OTHERWISE PRESS [ESC]  
 TO EXIT "UPDATE" (CR)

0C02-	45 4E	EOR	\$4E
0C04-	20 55 50	JSR	\$5055
0C07-	44	???	
0C08-	41 54	EOR	(\$54, X)
0C0A-	45 44	EOR	\$44
0C0C-	2C 20 59	BIT	\$5920
0C0F-	4F	???	
0C10-	55 20	EOR	\$20, X
0C12-	4D 41 59	EOR	\$5941
0C15-	52	???	
0C16-	45 4D	EOR	\$4D
0C18-	4F	???	
0C19-	56 45	LSR	\$45, X
0C1B-	20 49 54	JSR	\$5449
0C1E-	20 41 54	JSR	\$5441
0C21-	20 54 48	JSR	\$4854
0C24-	49 53	EOR	##53
0C26-	20 54 49	JSR	\$4954
0C29-	4D 45 2E	EOR	\$2E45
0C2C-	0D 0D 20	ORA	\$200D
0C2F-	20 49 46	JSR	\$4649
0C32-	20 59 4F	JSR	\$4F59
0C35-	55 20	EOR	\$20, X
0C37-	57	???	
0C38-	49 53	EOR	##53
0C3A-	48	PHA	
0C3B-	20 54 4F	JSR	\$4F54
0C3E-	20 22 55	JSR	\$5522
0C41-	50 44	BVC	\$0C87
0C43-	41 54	EOR	(\$54, X)
0C45-	45 22	EOR	\$22
0C47-	20 41 4E	JSR	\$4E41
0C4A-	4F	???	
0C4B-	54	???	
0C4C-	48	PHA	
0C4D-	45 52	EOR	\$52
0C4F-	20 44 49	JSR	\$4944
0C52-	53	???	
0C53-	4B	???	
0C54-	2D 0D 45	AND	\$450D
0C57-	54	???	
0C58-	54	???	
0C59-	45 2C	EOR	\$2C
0C5B-	20 50 52	JSR	\$5250
0C5E-	45 53	EOR	\$53
0C60-	53	???	
0C61-	20 5B 52	JSR	\$525B
0C64-	45 54	EOR	\$54
0C66-	55 52	EOR	\$52, X
0C68-	4E 5D 2E	LSR	\$2E5D
0C6B-	0D 0D 20	ORA	\$200D
0C6E-	20 4F 54	JSR	\$544F
0C71-	48	PHA	
0C72-	45 52	EOR	\$52
0C74-	57	???	
0C75-	49 53	EOR	##53
0C77-	45 20	EOR	\$20
0C79-	50 52	BVC	\$0CCD
0C7B-	45 53	EOR	\$53
0C7D-	53	???	
0C7E-	20 5B 45	JSR	\$455B
0C81-	53	???	
0C82-	43	???	

0C83-	5D 20 54	EOR	\$5420, X
0C86-	4F	???	
0C87-	20 45 58	JSR	\$5845
0C8A-	49 54	EOR	##54
0C8C-	20 22 55	JSR	\$5522
0C8F-	50 44	BVC	\$0CD5
0C91-	41 54	EOR	(\$54, X)
0C93-	45 22	EOR	\$22
0C95-	8D 0D 0D	STA	\$0D0D
0C98-	20 20 49	JSR	\$4920
0C9B-	46 20	LSR	\$20
0C9D-	59 4F 55	EOR	\$554F, Y
0CA0-	20 57 49	JSR	\$4957
0CA3-	53	???	
0CA4-	48	PHA	
0CA5-	20 54 4F	JSR	\$4F54
0CA8-	20 52 45	JSR	\$4552
0CAB-	54	???	
0CAC-	52	???	
0CAD-	59 20 50	EOR	\$5020, Y
0CB0-	52	???	
0CB1-	45 53	EOR	\$53
0CB3-	53	???	
0CB4-	20 5B 52	JSR	\$525B
0CB7-	45 54	EOR	\$54
0CB9-	55 52	EOR	\$52, X
0CBB-	4E 5D 0D	LSR	\$0D5D
0CBE-	0D 20 20	ORA	\$2020
0CC1-	4F	???	
0CC2-	54	???	
0CC3-	48	PHA	
0CC4-	45 52	EOR	\$52
0CC6-	57	???	
0CC7-	49 53	EOR	##53
0CC9-	45 20	EOR	\$20
0CCB-	50 52	BVC	\$0D1F
0CCD-	45 53	EOR	\$53
0CCF-	53	???	
0CD0-	20 5B 45	JSR	\$455B
0CD3-	53	???	
0CD4-	43	???	
0CD5-	5D 20 54	EOR	\$5420, X
0CD8-	4F	???	
0CD9-	20 45 58	JSR	\$5845
0CDC-	49 54	EOR	##54
0CDE-	20 22 55	JSR	\$5522
0CE1-	50 44	BVC	\$0D27
0CE3-	41 54	EOR	(\$54, X)
0CE5-	45 22	EOR	\$22
0CE7-	8D 0D 20	STA	\$200D
0CEA-	20 49 4E	JSR	\$4E49
0CED-	53	???	
0CEE-	45 52	EOR	\$52
0CF0-	54	???	
0CF1-	20 41 20	JSR	\$2041
0CF4-	53	???	
0CF5-	59 53 54	EOR	\$5453, Y
0CF8-	45 4D	EOR	\$4D
0CFA-	20 44 49	JSR	\$4944
0CFD-	53	???	
0CFE-	4B	???	
0CFF-	45 54	EOR	\$54
0D01-	54	???	

MSG #5

<CR><CR> TO IF TO YOU TO WISH TO  
TO RETRY TO PRESS TO [RETURN]  
<CR><CR> TO OTHERWISE TO  
PRESS TO [ESC] TO TO EXIT  
TO "UPDATE" <CR>

MSG #6

<CR> TO INSERT TO A TO  
SYSTEM TO DISKETTE TO AND TO  
PRESS <CR> [RETURN] TO TO  
REBOOT TO DOS TO

OD02-	45 20	EOR	\$20
OD04-	41 4E	EOR	(\$4E, X)
OD06-	44	???	
OD07-	20 50 52	JSR	\$5250
OD0A-	45 53	EOR	\$53
OD0C-	53	???	
OD0D-	0D 5B 52	ORA	\$525B
OD10-	45 54	EOR	\$54
OD12-	55 52	EOR	\$52, X
OD14-	4E 5D 20	LSR	\$205D
OD17-	54	???	
OD18-	4F	???	
OD19-	20 52 45	JSR	\$4552
OD1C-	42	???	
OD1D-	4F	???	
OD1E-	4F	???	
OD1F-	54	???	
OD20-	20 44 4F	JSR	\$4F44
OD23-	53	???	

---

OD24-	A0 0D	LDY	#\$0D	MSG #7
-------	-------	-----	-------	--------

OD26-	07	???	
OD27-	07	???	
OD28-	07	???	
OD29-	20 20 55	JSR	\$5520
OD2C-	4E 41 42	LSR	\$4241
OD2F-	4C 45 20	JMP	\$2045
OD32-	54	???	
OD33-	4F	???	
OD34-	20 52 45	JSR	\$4552
OD37-	41 44	EOR	(\$44, X)
OD39-	20 49 4D	JSR	\$4D49
OD3C-	41 47	EOR	(\$47, X)
OD3E-	45 2E	EOR	\$2E

<CR> (3) <BEL> to UNABLE to  
TO to READ to IMAGE. <CR><CR>

---

OD40-	0D 8D 0D	ORA	\$0D8D	MSG #8
-------	----------	-----	--------	--------

OD43-	07	???	
OD44-	07	???	
OD45-	07	???	
OD46-	20 20 49	JSR	\$4920
OD49-	4D 41 47	EOR	\$4741
OD4C-	45 20	EOR	\$20
OD4E-	4F	???	
OD4F-	46 20	LSR	\$20
OD51-	44	???	
OD52-	4F	???	
OD53-	53	???	
OD54-	20 33 2E	JSR	\$2E33
OD57-	32	???	
OD58-	20 28 4D	JSR	\$4D28
OD5B-	41 53	EOR	(\$53, X)
OD5D-	54	???	
OD5E-	45 52	EOR	\$52
OD60-	29 20	AND	#\$20
OD62-	49 53	EOR	#\$53
OD64-	20 4E 4F	JSR	\$4F4E
OD67-	54	???	
OD68-	0D 0D 41	ORA	\$410D
OD6B-	56 41	LSR	\$41, X
OD6D-	49 4C	EOR	##4C
OD6F-	41 42	EOR	(\$42, X)
OD71-	4C 45 2E	JMP	\$2E45
OD74-	20 20 43	JSR	\$4320
OD77-	48	PHA	
OD78-	45 43	EOR	\$43

<CR> (3) <BEL> to IMAGE to  
OF to DOS to 3.2 to (MASTER) to  
IS to NOT <CR> <CR> AVAILABLE.  
to CHECK to INSTRUCTIONS. <CR><CR>



OD7A-	4B		???	
OD7B-	20	49	4E	JSR \$4E49
OD7E-	53		???	
OD7F-	54		???	
OD80-	52		???	
OD81-	55	43		EOR \$43,X
OD83-	54		???	
OD84-	49	4F		EOR #\$4F
OD86-	4E	53	2E	LSR \$2E53
OD89-	0D	8D	0D	ORA \$0D8D
OD8C-	07		???	
OD8D-	07		???	
OD8E-	07		???	
OD8F-	20	20	55	JSR \$5520
OD92-	4E	41	42	LSR \$4241
OD95-	4C	45	20	JMP \$2045
OD98-	54		???	
OD99-	4F		???	
OD9A-	20	57	52	JSR \$5257
OD9D-	49	54		EOR #\$54
OD9F-	45	2E		EOR \$2E
ODA1-	20	20	44	JSR \$4420
ODA4-	49	53		EOR #\$53
ODA6-	4B		???	
ODA7-	45	54		EOR \$54
ODA9-	54		???	
ODAA-	45	20		EOR \$20
ODAC-	4D	55	53	EOR \$5355
ODAF-	54		???	
ODB0-	20	42	45	JSR \$4542
ODB3-	0D	49	4E	ORA \$4E49
ODB6-	49	54		EOR #\$54
ODB8-	49	41		EOR #\$41
ODBA-	4C	49	5A	JMP \$5A49
DBD-	45	44		EOR \$44
ODBF-	20	50	52	JSR \$5250
ODC2-	4F		???	
ODC3-	50	45		BVC \$0E0A
ODC5-	52		???	
ODC6-	4C	59	2E	JMP \$2E59
ODC9-	20	20	43	JSR \$4320
ODCC-	48			FHA
ODCD-	45	43		EOR \$43
ODCF-	4B		???	
ODD0-	20	44	49	JSR \$4944
ODD3-	53		???	
ODD4-	4B		???	
ODD5-	45	54		EOR \$54
ODD7-	54		???	
ODD8-	45	0D		EOR \$0D
ODDA-	46	4F		LSR \$4F
ODDC-	52		???	
ODDD-	20	50	52	JSR \$5250
ODE0-	4F		???	
ODE1-	50	45		BVC \$0E28
ODE3-	52		???	
ODE4-	20	49	4E	JSR \$4E49
ODE7-	53		???	
ODE8-	45	52		EOR \$52
ODEA-	54		???	
ODEB-	49	4F		EOR #\$4F
ODED-	4E	2E	8D	LSR \$8D2E
ODF0-	0D	07	07	ORA \$0707

MSG #9

<CR> (3) <BEL> TO BE UNABLE TO BE  
 WRITE, TO DISKETTE TO MUST BE BECOME  
 INITIALIZED TO PROPERLY, TO  
 CHECK TO DISKETTE TO FOR TO  
 PROPER TO INSERTION, <CR>

MSG #9A

ODF3-	07			???	
ODF4-	20	20	44	JSR	#4420
ODF7-	49	53		EOR	#53
ODF9-	4B			???	
ODFA-	45	54		EOR	#54
ODFC-	54			???	
ODFD-	45	20		EOR	#20
ODFF-	49	53		EOR	#53
OE01-	20	57	52	JSR	#5257
OE04-	49	54		EOR	#54
OE06-	45	20		EOR	#20
OE08-	50	52		BVC	#0E5C
OE0A-	4F			???	
OE0B-	54			???	
OE0C-	45	43		EOR	#43
OE0E-	54			???	
OE0F-	45	44		EOR	#44
OE11-	2E	20	20	ROL	#2020
OE14-	52			???	
OE15-	45	4D		EOR	#4D
OE17-	4F			???	
OE18-	56	45		LSR	#45, X
OE1A-	0D	57	52	ORA	#5257
OE1D-	49	54		EOR	#54
OE1F-	45	20		EOR	#20
OE21-	50	52		BVC	#0E75
OE23-	4F			???	
OE24-	54			???	
OE25-	45	43		EOR	#43
OE27-	54			???	
OE28-	20	54	41	JSR	#4154
OE2B-	42			???	
OE2C-	2E	8D	00	ROL	#008D
OE2F-	00			BRK	
OE30-	00			BRK	
OE31-	00			BRK	
OE32-	00			BRK	
OE33-	00			BRK	
OE34-	00			BRK	
OE35-	00			BRK	
OE36-	00			BRK	
OE37-	00			BRK	
OE38-	00			BRK	
OE39-	00			BRK	
OE3A-	00			BRK	
OE3B-	00			BRK	
OE3C-	00			BRK	
OE3D-	00			BRK	
OE3E-	00			BRK	
OE3F-	00			BRK	
OE40-	00			BRK	
OE41-	00			BRK	
OE42-	00			BRK	
OE43-	00			BRK	
OE44-	00			BRK	
OE45-	00			BRK	
OE46-	00			BRK	
OE47-	00			BRK	
OE48-	00			BRK	
OE49-	00			BRK	
OE4A-	00			BRK	
OE4B-	00			BRK	
OE4C-	01	60		ORA	(#60, X)

<CR> (3) <BEL> 66 DISKETTE 6 IS  
 6 WRIT 6 PROTECTED, 66 REMOVE  
 WRIT 6 PROTECT 6 TAB. <CR>

END MSGS

GREETING FILE NAME

SLOT #16

TYPE  
 JOB

RWTS JOB 1 "MASK"

Address	Drive	Operation	Value
0E4E-	01 00	ORA	(#00,X)
0E50-	00	BRK	
0E51-	00	BRK	
0E52-	5D 0E 00	EOR	\$000E,X
0E55-	12	???	
0E56-	00	BRK	
0E57-	00	BRK	
0E58-	00	BRK	
0E59-	00	BRK	
0E5A-	00	BRK	
0E5B-	60	RTS	
0E5C-	01 00	ORA	(#00,X)
0E5E-	01 EF	ORA	(#EF,X)
0E60-	D8	CLD	
0E61-	01 60	ORA	(#60,X)
0E63-	01 00	ORA	(#00,X)
0E65-	00	BRK	
0E66-	00	BRK	
0E67-	5D 0E 00	EOR	\$000E,X
0E6A-	12	???	
0E6B-	00	BRK	
0E6C-	00	BRK	
0E6D-	00	BRK	
0E6E-	00	BRK	
0E6F-	00	BRK	
0E70-	60	RTS	
0E71-	01 57	ORA	(#57,X)
0E73-	52	???	
0E74-	49 54	EOR	##54
0E76-	54	???	
0E77-	45 4E	EOR	\$4E
0E79-	20 42 59	JSR	\$5942
0E7C-	20 4A 41	JSR	\$414A
0E7F-	4D 45 53	EOR	\$5345
0E82-	20 52 2E	JSR	\$2E52
0E85-	20 48 55	JSR	\$5548
0E88-	53	???	
0E89-	54	???	
0E8A-	4F	???	
0E8B-	4E 20 44	LSR	\$4420
0E8E-	45 43	EOR	\$43
0E90-	45 4D	EOR	\$4D
0E92-	42	???	
0E93-	45 52	EOR	\$52
0E95-	20 31 35	JSR	\$3531
0E98-	2C 20 31	BIT	\$3120
0E9B-	39 37 38	AND	\$3837,Y
0E9E-	20 28 54	JSR	\$5428
0EA1-	48	PHA	
0EA2-	49 53	EOR	##53
0EA4-	20 4D 45	JSR	\$454D
0EA7-	53	???	
0EA8-	53	???	
0EA9-	41 47	EOR	(#47,X)
0EAB-	45 20	EOR	\$20
0EAD-	49 53	EOR	##53
0EAF-	20 46 49	JSR	\$4946
0EB2-	4C 4C 45	JMP	\$454C
0EB5-	52	???	
0EB6-	2C 20 57	BIT	\$5720
0EB9-	48	PHA	
0EBA-	59 20 52	EOR	\$5220,Y
0EBD-	45 41	EOR	\$41

DRIVE

DEV  
CHAR  
TAB

RWTS JOB 2

SIGNATURE

NOT USED

WRITTEN BY  
 JAMES R.  
 HUSTON  
 15, 6 1978  
 (THIS MESSAGE IS  
 FILLER, WHY  
 READ IT?)

0EBF-	44		???	
0EC0-	20	49 54	JSR	\$5449
0EC3-	3F		???	
0EC4-	29	00	AND	##00
0EC6-	00		BRK	
0EC7-	00		BRK	
0EC8-	00		BRK	
0EC9-	00		BRK	
0ECA-	00		BRK	
0ECB-	00		BRK	
0ECC-	00		BRK	
0ECD-	00		BRK	
0ECE-	00		BRK	
0ECF-	00		BRK	
0ED0-	00		BRK	
0ED1-	00		BRK	
0ED2-	00		BRK	
0ED3-	00		BRK	
0ED4-	00		BRK	
0ED5-	00		BRK	
0ED6-	00		BRK	
0ED7-	00		BRK	
0ED8-	00		BRK	
0ED9-	00		BRK	
0EDA-	00		BRK	
0EDB-	00		BRK	
0EDC-	00		BRK	
0EDD-	00		BRK	
0EDE-	00		BRK	
0EDF-	00		BRK	
0EE0-	00		BRK	
0EE1-	00		BRK	
0EE2-	00		BRK	
0EE3-	00		BRK	
0EE4-	00		BRK	
0EE5-	00		BRK	
0EE6-	00		BRK	
0EE7-	00		BRK	
0EE8-	00		BRK	
0EE9-	00		BRK	
0EEA-	00		BRK	
0EEB-	00		BRK	
0EEC-	00		BRK	
0EED-	00		BRK	
0EEE-	00		BRK	
0EEF-	00		BRK	
0EF0-	00		BRK	
0EF1-	00		BRK	
0EF2-	00		BRK	
0EF3-	00		BRK	
0EF4-	00		BRK	
0EF5-	00		BRK	
0EF6-	00		BRK	
0EF7-	00		BRK	
0EF8-	00		BRK	
0EF9-	00		BRK	
0EFA-	00		BRK	
0EFB-	00		BRK	
0EFC-	00		BRK	
0EFD-	00		BRK	
0EFE-	00		BRK	
0EFF-	00		BRK	
0F00-	FF		???	