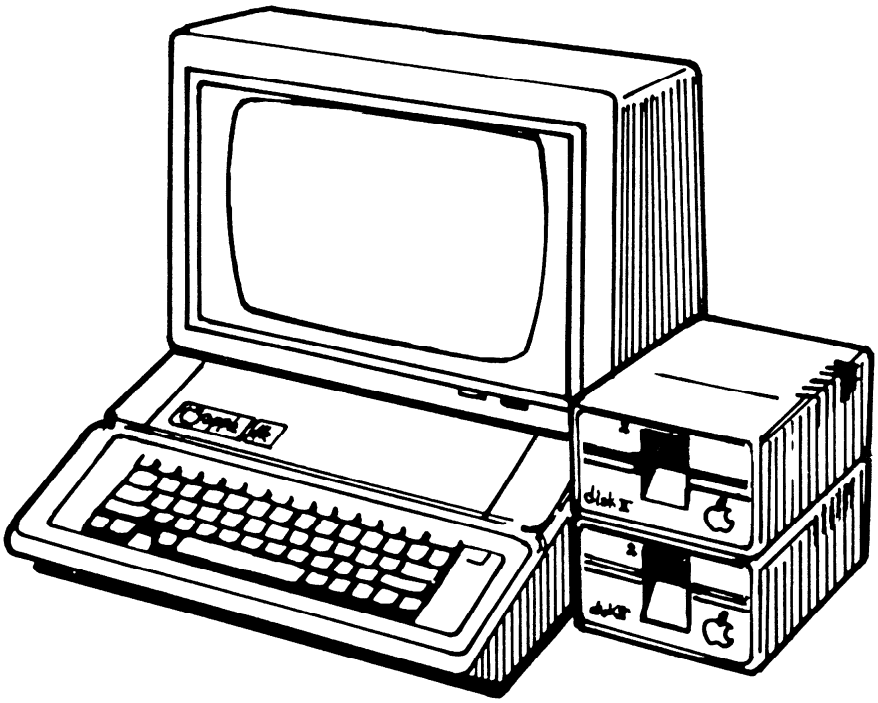
Apple][Computer Information

Apple][ProDOS 1.7 Operating System Source Listing



Apple Computer

July 1988

Source File Catalog

Name	Type	Crtr	Size	Flags	Last-Mod-Date	Creation-Date
BigTextDocument	TEXT	DAVE	884K	lvbspoimad	5/16/06 5:06 PM	5/16/06 5:06 PM
BLD.ONLY.PRODOS.hex	TEXT	MPS	8K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM
BUILDP8.PRODOS.hex	TEXT	MPS	8K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM
BUS.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
CCLOCK.pretty	TEXT	MPS	12K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.CCLOCK.pretty	TEXT	MPS	12K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.MLI.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.RAM.pretty	TEXT	MPS	28K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.SEL.pretty	TEXT	MPS	24K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.TCLOCK.pretty	TEXT	MPS	8K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
L.XRW.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
MAS.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
MASTER.BUILD.hex	TEXT	MPS	4K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM
MASTER.INIT.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
MLI.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
MLI.SRC.ALLOC.pretty	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.ATPREPLY.pre…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.ATPREQUEST.p…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.ATREPLY.pret…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.BFMGR.ONE.pr…	TEXT	MPS	24K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.BFMGR.pretty	TEXT	MPS	28K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.BOOT.SRC.pre…	TEXT	MPS	40K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.CLOSE.EOF.pr…	TEXT	MPS	24K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.COMMANDS.pre…	TEXT	MPS	12K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.CREATE.prett…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DATATBLS.pre…	TEXT	MPS	8K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DDPDRIVER.pr…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DESTROY.pret…	TEXT	MPS	24K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DETREE.prett…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DEVSRCH.pret…	TEXT	MPS	36K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.DOATALK.pret…	TEXT	MPS	4K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.EQUATES.pret…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.FNDFIL.prett…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.GLOBALS.pret…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.INTERRUPT.pr…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.MEMMGR.prett…	TEXT	MPS	28K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.NBPDRIVER.pr…	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.NEWFNDVOL.pr…	TEXT	MPS	24K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.OPENDRIVER.p…	TEXT	MPS	16K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.PLDR.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.POSN.OPEN.pr…	TEXT	MPS	36K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.PRODIR.prett…	TEXT	MPS	4K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.PROLDR.prett…	TEXT	MPS	60K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.READ.WRITE.p…	TEXT	MPS	36K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.RELOC.pretty	TEXT	MPS	20K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.REVISIONS.pr…	TEXT	MPS	32K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.ROM.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.WRKSPACE.pre…	TEXT	MPS	8K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.XDOSMLI.pret…	TEXT	MPS	24K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.XRW1.pretty	TEXT	MPS	28K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
MLI.SRC.XRW2.pretty	TEXT	MPS	32K	lvbspoimad	5/16/06 4:58 PM	5/16/06 4:58 PM
PRODOS.EXEC.LST.pret…	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
PRODOS.EXEC.pretty	TEXT	MPS	4K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
RAM.pretty	TEXT	MPS	28K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
RPM.0.hex	TEXT	MPS	8K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM
SEL.0.hex	TEXT	MPS	4K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM
SEL.pretty	TEXT	MPS	24K	lvbspoimad	5/16/06 4:57 PM	5/16/06 4:57 PM
TCLOCK.0.hex	TEXT	MPS	4K	lvbspoimad	5/16/06 4:56 PM	5/16/06 4:56 PM

```
TCLOCK.pretty          TEXT MPS      8K lvbspoimad    5/16/06  4:57 PM    5/16/06  4:57 PM
XRW.pretty             TEXT MPS      4K lvbspoimad    5/16/06  4:57 PM    5/16/06  4:57 PM
```

```
================================================================================
DOCUMENT BLD.ONLY.PRODOS.hex
================================================================================

File ............. "BLD.ONLY.PRODOS"
Fork ............. DATA
Size (bytes) ..... 940 (0KB) / $000003AC
Created .......... Wednesday, April 12, 2006 -- 9:20:30 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:30 AM

D/000000: 1C080A00 4424D0E7 2834293A 4224D044 [....D$..(4):B$.D]
D/000010: 24C82242 4C4F4144 20220026 08140097 [$."BLOAD.".&....]
D/000020: 3AA23130 0044081E 00BA222D 3E425549 [:.10.D...."->BUI]
D/000030: 4C44494E 47202F50 524F444F 53204D4C [LDING./PRODOS.ML]
D/000040: 4922004E 082800A5 AB343130 00670832 [I".N.(...410.g.2]
D/000050: 00BA4424 3B225645 52494659 202F5052 [..D$;"VERIFY./PR]
D/000060: 4F444F53 22007108 3C00A5AB 31303000 [ODOS".q.<...100.]
D/000070: 91084600 BA44243B 22564552 49465920 [..F..D$;"VERIFY.]
D/000080: 2F50524F 444F532F 50524F44 4F532200 [/PRODOS/PRODOS".]
D/000090: B1085000 BA44243B 22554E4C 4F434B20 [..P..D$;"UNLOCK.]
D/0000A0: 2F50524F 444F532F 50524F44 4F532200 [/PRODOS/PRODOS".]
D/0000B0: D1085A00 BA44243B 2244454C 45544520 [..Z..D$;"DELETE.]
D/0000C0: 2F50524F 444F532F 50524F44 4F532200 [/PRODOS/PRODOS".]
D/0000D0: F6086400 B9323136 2C303AB2 20545552 [..d..216,0:..TUR]
D/0000E0: 4E204F46 46204552 524F5220 54524150 [N.OFF.ERROR.TRAP]
D/0000F0: 50494E47 0008096E 00903A91 3A893AA2 [PING...n..:.:.:.]
D/000100: 31303A96 32340012 097800BA 222E223B [10:.24...x.."."; ]
D/000110: 00310982 00BA4224 3B224D4C 492E302C [.1....B$;"MLI.0,]
D/000120: 41243230 3030223A B24C4F41 44455200 [A$2000":.LOADER.]
D/000130: 3B098C00 BA222E22 3B005209 9600BA42 [;...."."; .R....B]
D/000140: 243B2252 414D2E31 2C412432 42303022 [$;"RAM.1,A$2B00"]
D/000150: 005C09A0 00BA222E 223B0073 09AA00BA [.\....".";.s....]
D/000160: 42243B22 52414D2E 302C4124 32433030 [B$;"RAM.0,A$2C00]
D/000170: 22007D09 B400BA22 2E223B00 9409BE00 [".}...."."; .....]
D/000180: BA42243B 2252414D 2E322C41 24324530 [.B$;"RAM.2,A$2E0]
D/000190: 3022009E 09C800BA 222E223B 00BC09D2 [0"......"."; ....]
D/0001A0: 00BA4224 3B224D4C 492E322C 41243246 [..B$;"MLI.2,A$2F]
D/0001B0: 3030223A B220204D 4C4900C6 09DC00BA [00":...MLI......]
D/0001C0: 222E223B 00EB09E6 00BA4224 3B224D4C ["."; ......B$;"ML]
D/0001D0: 492E312C 41243530 3030223A B220474C [I.1,A$5000":..GL]
D/0001E0: 4F42414C 20504147 4500F509 F000BA22 [OBAL.PAGE......"]
D/0001F0: 2E223B00 1F0A0E01 BA42243B 2254434C [."; ......B$;"TCL]
D/000200: 4F434B2E 302C4124 35313030 223AB220 [OCK.0,A$5100":..]
D/000210: 54434C4F 434B2044 52495645 5200290A [TCLOCK.DRIVER.).]
D/000220: 1801BA22 2E223B00 4D0A2201 BA42243B [..."."; .M."..B$;]
D/000230: 224D4C49 2E332C41 24353139 42223AB2 ["MLI.3,A$519B":.]
D/000240: 20494E54 45525255 50545300 570A2C01 [.INTERRUPTS.W.,.]
D/000250: BA222E22 3B007E0A 3601BA42 243B2258 [."."; .~.6..B$;"X]
D/000260: 52572E30 2C412435 32303022 3AB22043 [RW.0,A$5200":..C]
D/000270: 4F524520 524F5554 494E4553 00880A40 [ORE.ROUTINES...@]
D/000280: 01BA222E 223B00AC 0A4A01BA 42243B22 [.."."; ...J..B$;"]
D/000290: 53454C2E 302C4124 35393030 223AB220 [SEL.0,A$5900":..]
D/0002A0: 44495350 41544348 455200B6 0A5401BA [DISPATCHER...T..]
D/0002B0: 222E223B 00EA0A58 01BA4224 3B224343 ["."; ...X..B$;"CC]
D/0002C0: 4C4F434B 2E302C41 24354330 30223AB2 [LOCK.0,A$5C00":.]
D/0002D0: 20202043 4F52544C 414E4420 434C4F43 [...CORTLAND.CLOC]
D/0002E0: 4B204452 49564552 00F40A59 01BA222E [K.DRIVER...Y.."."]
D/0002F0: 223B0019 0B5E01BA 44243B22 43524541 ["; ...^..D$;"CREA]
D/000300: 5445202F 50524F44 4F532F50 524F444F [TE./PRODOS/PRODO]
D/000310: 532C5424 46462200 230B6801 BA222E22 [S,T$FF".#.h.."."]
D/000320: 3B00550B 7201BA44 243B2242 53415645 [;.U.r..D$;"BSAVE]
D/000330: 202F5052 4F444F53 2F50524F 444F532C [./PRODOS/PRODOS,]
```

```
D/000340: 54244646 2C412432 3030302C 4C243343 [T$FF,A$2000,L$3C]
D/000350: 37442200 5F0B7C01 BA222122 3B00720B [7D"._.|.."!";.r.]
D/000360: 8601BA3A BA22414C 4C20444F 4E452122 [...:."ALL.DONE!"]
D/000370: 00780B90 018000AB 0B9A01B9 3231362C [.x.........216,]
D/000380: 303ABA3A BAE72837 293B2256 4F4C554D [0:.:..(7);"VOLUM]
D/000390: 45202F50 524F444F 53204E4F 5420464F [E./PRODOS.NOT.FO]
D/0003A0: 554E442E 223ABA3A 80000000          [UND.":.:....    ]

File ............. "BLD.ONLY.PRODOS"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

FINIS
```

```
================================================================================
DOCUMENT BUILDP8.PRODOS.hex
================================================================================

File ............. "BUILDP8.PRODOS"
Fork ............. DATA
Size (bytes) ..... 1,209 (1KB) / $000004B9
Created .......... Wednesday, April 12, 2006 -- 9:20:28 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:28 AM

D/000000: 11080500 BAE72834 29225052 23332200 [......(4)"PR#3".]
D/000010: 17080700 BA003208 0A004424 D0E72834 [......2...D$..(4]
D/000020: 293A4224 D04424C8 22424C4F 41442022 [):B$.D$."BLOAD."]
D/000030: 003C0814 00973AA2 3130006C 081E00BA [.<....:.10.l....]
D/000040: 222D3E42 75696C64 696E6720 50726F44 ["->Building.ProD]
D/000050: 4F53204D 4C492066 6F722050 3820616E [OS.MLI.for.P8.an]
D/000060: 64205072 6F444F53 38220076 082800A5 [d.ProDOS8".v.(..]
D/000070: AB343130 00950832 00BA4424 3B225645 [.410...2..D$;"VE]
D/000080: 52494659 202F5052 4F444F53 31362F4D [RIFY./PRODOS16/M]
D/000090: 4C492200 9F083C00 A5AB3130 3000C508 [LI"...<...100...]
D/0000A0: 4600BA44 243B2256 45524946 59202F50 [F..D$;"VERIFY./P]
D/0000B0: 524F444F 5331362F 4D4C492F 50524F44 [RODOS16/MLI/PROD]
D/0000C0: 4F532200 E7084B00 BA44243B 22564552 [OS"...K..D$;"VER]
D/0000D0: 49465920 2F50524F 444F5331 362F4D4C [IFY./PRODOS16/ML]
D/0000E0: 492F5038 22000D09 5000BA44 243B2255 [I/P8"...P..D$;"U]
D/0000F0: 4E4C4F43 4B202F50 524F444F 5331362F [NLOCK./PRODOS16/]
D/000100: 4D4C492F 50524F44 4F532200 2F095500 [MLI/PRODOS"./.U.]
D/000110: BA44243B 22554E4C 4F434B20 2F50524F [.D$;"UNLOCK./PRO]
D/000120: 444F5331 362F4D4C 492F5038 22005509 [DOS16/MLI/P8".U.]
D/000130: 5A00BA44 243B2244 454C4554 45202F50 [Z..D$;"DELETE./P]
D/000140: 524F444F 5331362F 4D4C492F 50524F44 [RODOS16/MLI/PROD]
D/000150: 4F532200 77095F00 BA44243B 2244454C [OS".w._..D$;"DEL]
D/000160: 45544520 2F50524F 444F5331 362F4D4C [ETE./PRODOS16/ML]
D/000170: 492F5038 22009C09 6400B932 31362C30 [I/P8"...d..216,0]
D/000180: 3AB22054 55524E20 4F464620 4552524F [:..TURN.OFF.ERRO]
D/000190: 52205452 41505049 4E4700AE 096E0090 [R.TRAPPING...n..]
D/0001A0: 3A913A89 3AA23130 3A963430 00B80978 [:.:.:.10:.40...x]
D/0001B0: 00BA222E 223B00D7 098200BA 42243B22 [.."."....B$;"]
D/0001C0: 4D4C492E 302C4124 32303030 223AB24C [MLI.0,A$2000":.L]
D/0001D0: 4F414445 5200E109 8C00BA22 2E223B00 [OADER......".";.]
D/0001E0: F8099600 BA42243B 2252414D 2E312C41 [.....B$;"RAM.1,A]
D/0001F0: 24324230 30220002 0AA000BA 222E223B [$2B00"......".";]
D/000200: 00190AAA 00BA4224 3B225241 4D2E302C [......B$;"RAM.0,]
D/000210: 41243243 30302200 230AB400 BA222E22 [A$2C00".#....".".]
D/000220: 3B003A0A BE00BA42 243B2252 414D2E32 [;.:....B$;"RAM.2]
D/000230: 2C412432 45303022 00440AC8 00BA222E [,A$2E00".D....".]
D/000240: 223B0062 0AD200BA 42243B22 4D4C492E [";.b....B$;"MLI.]
D/000250: 322C4124 32463030 223AB220 204D4C49 [2,A$2F00":...MLI]
D/000260: 006C0ADC 00BA222E 223B0091 0AE600BA [.l....".";......]
D/000270: 42243B22 4D4C492E 312C4124 35303030 [B$;"MLI.1,A$5000]
D/000280: 223AB220 474C4F42 414C2050 41474500 [":..GLOBAL.PAGE.]
D/000290: 9B0AF000 BA222E22 3B00C50A 0E01BA42 [.....".";......B]
D/0002A0: 243B2254 434C4F43 4B2E302C 41243531 [$;"TCLOCK.0,A$51]
D/0002B0: 3030223A B2205443 4C4F434B 20445249 [00":..TCLOCK.DRI]
D/0002C0: 56455200 CF0A1801 BA222E22 3B00F30A [VER......".";...]
D/0002D0: 2201BA42 243B224D 4C492E33 2C412435 ["..B$;"MLI.3,A$5]
D/0002E0: 31394222 3AB22049 4E544552 52555054 [19B":..INTERRUPT]
D/0002F0: 5300FD0A 2C01BA22 2E223B00 240B3601 [S...,.."."";.$.6.]
D/000300: BA42243B 22585257 2E302C41 24353230 [.B$;"XRW.0,A$520]
D/000310: 30223AB2 20434F52 4520524F 5554494E [0":..CORE.ROUTIN]
D/000320: 4553002E 0B4001BA 222E223B 00520B4A [ES...@.."."";.R.J]
D/000330: 01BA4224 3B225345 4C2E302C 41243539 [..B$;"SEL.0,A$59]
```

```
D/000340: 3030223A B2204449 53504154 43484552 [00":..DISPATCHER]
D/000350: 005C0B54 01BA222E 223B0090 0B5801BA [.\.T.."."";...X..]
D/000360: 42243B22 43434C4F 434B2E30 2C412435 [B$;"CCLOCK.0,A$5]
D/000370: 43303022 3AB22020 20434F52 544C414E [C00":....CORTLAN]
D/000380: 4420434C 4F434B20 44524956 4552009A [D.CLOCK.DRIVER..]
D/000390: 0B5901BA 222E223B 00C50B5E 01BA4424 [.Y.."."";...^..D$]
D/0003A0: 3B224352 45415445 202F5052 4F444F53 [;"CREATE./PRODOS]
D/0003B0: 31362F4D 4C492F50 524F444F 532C5424 [16/MLI/PRODOS,T$]
D/0003C0: 46462200 EC0B6301 BA44243B 22435245 [FF"...c..D$;"CRE]
D/0003D0: 41544520 2F50524F 444F5331 362F4D4C [ATE./PRODOS16/ML]
D/0003E0: 492F5038 2C542446 462200F6 0B6801BA [I/P8,T$FF"...h..]
D/0003F0: 222E223B 002E0C72 01BA4424 3B224253 ["."";...r..D$;"BS]
D/000400: 41564520 2F50524F 444F5331 362F4D4C [AVE./PRODOS16/ML]
D/000410: 492F5052 4F444F53 2C542446 462C4124 [I/PRODOS,T$FF,A$]
D/000420: 32303030 2C4C2433 43374422 00620C77 [2000,L$3C7D".b.w]
D/000430: 01BA4424 3B224253 41564520 2F50524F [..D$;"BSAVE./PRO]
D/000440: 444F5331 362F4D4C 492F5038 2C542446 [DOS16/MLI/P8,T$F]
D/000450: 462C4124 32303030 2C4C2433 43374422 [F,A$2000,L$3C7D"]
D/000460: 006C0C7C 01BA2221 223B007F 0C8601BA [.l.|.."!";......]
D/000470: 3ABA2241 4C4C2044 4F4E4521 2200850C [:."ALL.DONE!"...]
D/000480: 90018000 B80C9A01 B9323136 2C303ABA [.........216,0:.]
D/000490: 3ABAE728 37293B22 564F4C55 4D45202F [:..(7);"VOLUME./]
D/0004A0: 50524F44 4F53204E 4F542046 4F554E44 [PRODOS.NOT.FOUND]
D/0004B0: 2E223ABA 3A800000 00                [.":.:....       ]

File ............. "BUILDP8.PRODOS"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

FINIS
```

```
================================================================================
DOCUMENT BUS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  BUS
; ###############################################################################################

                LST          ON,A,V
*
                REP          60
*                                                              *
*                 PRODOS APPLEBUS DRIVER ROUTINE               *
*                                                              *
*              COPYRIGHT APPLE COMPUTER, INC., 1985            *
*                                                              *
*                     ALL RIGHTS RESERVED                      *
*                                                              *
                REP          60
*
*
*  AppleBus driver code resides in the Main Language Card,
*    Bank 2, starting at $D400.
*
                MSB          ON
*
                sbtl         "AppleTalk Drivers"
DEBUG           equ          0
SAFECLOSE       EQU          0
TWOC            EQU          1
SAVESPACE       EQU          1
NOLAP           EQU          1
PRODOS          EQU          0
KERNEL          EQU          1
PASCAL          EQU          2
OPERS           equ          ProDOS
*
*** The following identifier is used to determine which ProDOS system
*** you wish to build.  Right now there are 2 choices: ProDOS1.2,
*** or EdNet.  To build one or the other, you must change the EDNET
*** identifier to a 1 for EdNet, and to a 0 for ProDOS 1.2.
*
* Also note that the ATALK.DRIVERS file used in EdNet ProDOS version
* must be of file type "SYS".
*
* WARNING: Because of EdAsm's lack of condition assembly directives,
*          EDNET must be set to 0 before assembling the relocatable
*          driver version!
*
EDNET           equ          0
*
                IFEQ         OPERS-PRODOS+EDNET
                ORG          $D400
                FIN
*
                IFNE         EDNET
                SYS
                ORG          $D000
                FIN
*
                IFNE         OPERS-PRODOS
```

```
                rel
                org             $100
                FIN
*
                include         mli.src/commands
                page
                include         mli.src/opendriver
                page
                include         mli.src/interrupt
                page
                include         mli.src/ddpdriver
                page
                include         mli.src/nbpdriver
                page
                include         mli.src/atprequest
                page
                include         mli.src/atpreply
zzzz            equ             *

; ###############################################################################################
; #   END OF FILE:  BUS
; #   LINES      :  71
; #   CHARACTERS :  2311
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT CCLOCK.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  CCLOCK
; ###############################################################################################

               LST          OFF,NOASYM,NOVSYM,NOGEN
               X65816                              ; Must go here for 24 bit expressions.
               XREFSLOT     6
               SEG          $00
*********************************************************
*
*       ProDOS 8 CORTLAND CLOCK DRIVER
*
*         COPYRIGHT APPLE COMPUTER, INC., 1986
*
*         ALL RIGHTS RESERVED
*
*       Written by Kerry Laidlaw, 2/12/86
*       Modified by Mike Askins, 9/6/86
*       Modified by Fern Bachman, 9/7/86
*
*********************************************************
*
* This is the ProDOS8 Cortland built-in clock driver.
* Its sole function in life is to fetch the time from the Cortland
* clock via the Read Hex Time misc. tool call, and transfer this
* time into the ProDOS global page time format.
*
* This routine will IGNORE any errors passed back to it from the
* Read Hex Time call.  This was done since existing ProDOS8 programs
* cannot deal with some new time error code.
* Thus the only way that a user can tell if his Cortland clock is
* broken, is by noticing that the date and time fields are zeroed.
*
* Note: There are some interesting facts to know regarding the
* slot clock driver for ProDOS8 and the built-in
* Cortland clock.  The year value returned from the Cortland clock
* is an offset from the year 1900.  Thus Cortland is capable of
* reporting the year correctly until 1900+255=2155.  Only 7 bits
* are used for the year in the ProDOS8 global page, so theoretically
* 1900+127=2027 is the last year that ProDOS could represent on a
* Cortland.  But this is only if the ProDOS8 year value is interpreted
* as being an offset from 1900.
*
* Historically, the year value has been interpreted as the binary
* representation of the last two digits of the year 19xx.
* So this means that programs that display the year as a concatination
* of 19 and the ascii equivalent of the year value will work until 1999.
* And programs that just display the last two digits of the year will
* still work correctly until (20)27 if they convert the year value
* correctly, but ignore any hundredths place digit.
*
* Apple //e's that use slot clocks that utilize the slot clock
* driver have further restrictions of the year value.  The slot
* clock driver calculates the year given the position of the day
* of the week in the month.  This algorithm then uses a year look
* up table that has seven possible values.  Leap years are repeated
* in the table.  Since 1988 is a leap year, then the updated slot
```

```
* clock driver (file TCLOCK) will yield the six year offset values
* rather then seven.
* So before 1992, if ProDOS8 still exists, the slot clock driver
* routine must be updated again!
*
* So, we now have the following definition:
*    The value placed in the year field is defined as the
*    number of years past the year 1900.
*    Numerically speaking: Current Year = 1900 + year value.
*
*
statereg        EQU          $C068                    ;Cortland memory state register
clock.begin     equ          $d742                    ; Entry address of Cortland clock
driver.
MISC.TSN        equ          3                        ; Misc. tool set number.
Read.Time.Hex   equ          $0d                      ; Read Time Hex function number.
dispatch        equ          $e10000                  ; Tool Dispatcher entry address.
minutes         equ          $bf92                    ; Minutes in ProDOS global page.
hours           equ          $bf93                    ;
year            equ          $bf91                    ;
day             equ          $bf90                    ;
*
                org          clock.begin
*
************************* See Note #66,67 *************************
*
* This mod will force read/write main memory for the tool
* call by resetting the read/write auxillary memory bits
* in the state register (statereg).
*
                memory8
                sep          #$30                     ;Make sure we're in 8 bit mode
                lda          statereg                 ;Get the state reg
                sta          savestate                ;Keep for restore after tool call
                and          #$CF                     ;Clear the Read/Write aux memory bits
                sta          statereg                 ;Make it real
*
************************************************************
*
                INDEX16
                MEMORY16
*
* First off, lets get into native mode with 16 bit m & x.
*
                clc                                   ; Set e = 0, to set native mode.
                xce                                   ;
                rep          #$30                     ; Zero m & x for 16 bit mode.
                lda          #0                       ; Zero out result space.
                pha                                   ; Push 4 words for hex time result...
                pha                                   ;
                pha                                   ;
                pha                                   ;
                ldx          #Read.Time.Hex*$100+MISC.TSN ; Read Time Hex.
                jsl          =dispatch                ; Make the ReadTimeHex call...
*
* Note that no error condition is checked for, so the date will
* be zeroed by default if an error indeed happened.
*
*  Back to 8 bit m to access results on stack...
*
                MEMORY8
                sep          #$20                     ; 8 bit m
*
```

```
************************ See Note #66 ************************
*
                lda           savestate                 ;Restore the state register
                sta           statereg
*
************************************************************
*
*
* Now let's pull the time off the stack and stick it in the global page.
*
                pla                                     ; Pull off Seconds, and ignore.
                pla                                     ; Pull off Minutes.
                sta           minutes                   ; Store in global page.
                pla                                     ; Pull off Hours.
                sta           Hours                     ; Store in global page.
                pla                                     ; Pull off Year value.
                sta           year                      ; Store in global page.
                pla                                     ; Pull off Day.
                inc           a                         ; Increment day value for ProDOS8
format.
                sta           day                       ; Store in global page.
                pla                                     ; Pull off Month.
                inc           a                         ; Inc month value for ProDOS8 format.
                asl           a                         ; Shift month as it sits in between
                asl           a                         ; the year and day values.
                asl           a                         ;
                asl           a                         ;
                asl           a                         ;
                ora           day                       ; Put all but the top bit of month
value
                sta           day                       ; in the day byte.
                rol           year                      ; Put hi bit of mo. in lo bit of yr
byte.
                pla                                     ; Pull off unused byte.
                pla                                     ; Pull off Day of Week. Stack now
clean.
*
                sec                                     ; Now go back to emulation mode
                xce                                     ; to continue with ProDOS8.
                rts                                     ; That's all.
*
savestate       dfb           0                         ;Keep the state of state register
                asc           'JIMJAYKERRY&MIKE' ;   '
clock.end       equ           *
                ds            125-clock.end+clock.begin,0 ; Zero rest of 125 bytes.
size            equ           *-clock.begin             ; MUST be $7D (125) bytes in length!

; ###########################################################################################
; #   END OF FILE:  CCLOCK
; #   LINES      :  157
; #   CHARACTERS :  8022
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT L.CCLOCK.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.CCLOCK
; ###############################################################################################

               LST          ON,ASYM,VSYM,GEN
               X65816                                  ; Must go here for 24 bit expressions.
               XREFSLOT     6
               SEG          $00
**********************************************************
*
*       ProDOS 8 CORTLAND CLOCK DRIVER
*
*         COPYRIGHT APPLE COMPUTER, INC., 1986
*
*         ALL RIGHTS RESERVED
*
*       Written by Kerry Laidlaw, 2/12/86
*       Modified by Mike Askins, 9/6/86
*       Modified by Fern Bachman, 9/7/86
*
**********************************************************
*
* This is the ProDOS8 Cortland built-in clock driver.
* Its sole function in life is to fetch the time from the Cortland
* clock via the Read Hex Time misc. tool call, and transfer this
* time into the ProDOS global page time format.
*
* This routine will IGNORE any errors passed back to it from the
* Read Hex Time call.  This was done since existing ProDOS8 programs
* cannot deal with some new time error code.
* Thus the only way that a user can tell if his Cortland clock is
* broken, is by noticing that the date and time fields are zeroed.
*
* Note: There are some interesting facts to know regarding the
* slot clock driver for ProDOS8 and the built-in
* Cortland clock.  The year value returned from the Cortland clock
* is an offset from the year 1900.  Thus Cortland is capable of
* reporting the year correctly until 1900+255=2155.  Only 7 bits
* are used for the year in the ProDOS8 global page, so theoretically
* 1900+127=2027 is the last year that ProDOS could represent on a
* Cortland.  But this is only if the ProDOS8 year value is interpreted
* as being an offset from 1900.
*
* Historically, the year value has been interpreted as the binary
* representation of the last two digits of the year 19xx.
* So this means that programs that display the year as a concatination
* of 19 and the ascii equivalent of the year value will work until 1999.
* And programs that just display the last two digits of the year will
* still work correctly until (20)27 if they convert the year value
* correctly, but ignore any hundredths place digit.
*
* Apple //e's that use slot clocks that utilize the slot clock
* driver have further restrictions of the year value.  The slot
* clock driver calculates the year given the position of the day
* of the week in the month.  This algorithm then uses a year look
* up table that has seven possible values.  Leap years are repeated
* in the table.  Since 1988 is a leap year, then the updated slot
```

```
* clock driver (file TCLOCK) will yield the six year offset values
* rather then seven.
* So before 1992, if ProDOS8 still exists, the slot clock driver
* routine must be updated again!
*
* So, we now have the following definition:
*    The value placed in the year field is defined as the
*    number of years past the year 1900.
*    Numerically speaking: Current Year = 1900 + year value.
*
*
statereg        EQU          $C068                    ;Cortland memory state register
clock.begin     equ          $d742                    ; Entry address of Cortland clock
driver.
MISC.TSN        equ          3                        ; Misc. tool set number.
Read.Time.Hex   equ          $0d                      ; Read Time Hex function number.
dispatch        equ          $e10000                  ; Tool Dispatcher entry address.
minutes         equ          $bf92                    ; Minutes in ProDOS global page.
hours           equ          $bf93                    ;
year            equ          $bf91                    ;
day             equ          $bf90                    ;
*
                org          clock.begin
*
************************* See Note #66,67 *************************
*
* This mod will force read/write main memory for the tool
* call by resetting the read/write auxillary memory bits
* in the state register (statereg).
*
                memory8
                sep          #$30                     ;Make sure we're in 8 bit mode
                lda          statereg                 ;Get the state reg
                sta          savestate                ;Keep for restore after tool call
                and          #$CF                     ;Clear the Read/Write aux memory bits
                sta          statereg                 ;Make it real
*
************************************************************
*
                INDEX16
                MEMORY16
*
* First off, lets get into native mode with 16 bit m & x.
*
                clc                                   ; Set e = 0, to set native mode.
                xce                                   ;
                rep          #$30                     ; Zero m & x for 16 bit mode.
                lda          #0                       ; Zero out result space.
                pha                                   ; Push 4 words for hex time result...
                pha                                   ;
                pha                                   ;
                pha                                   ;
                ldx          #Read.Time.Hex*$100+MISC.TSN ; Read Time Hex.
                jsl          =dispatch                ; Make the ReadTimeHex call...
*
* Note that no error condition is checked for, so the date will
* be zeroed by default if an error indeed happened.
*
*  Back to 8 bit m to access results on stack...
*
                MEMORY8
                sep          #$20                     ; 8 bit m
*
```

```
************************ See Note #66 ************************
*
                lda          savestate                 ;Restore the state register
                sta          statereg
*
************************************************************
*
*
* Now let's pull the time off the stack and stick it in the global page.
*
                pla                                    ; Pull off Seconds, and ignore.
                pla                                    ; Pull off Minutes.
                sta          minutes                   ; Store in global page.
                pla                                    ; Pull off Hours.
                sta          Hours                     ; Store in global page.
                pla                                    ; Pull off Year value.
                sta          year                      ; Store in global page.
                pla                                    ; Pull off Day.
                inc          a                         ; Increment day value for ProDOS8
format.
                sta          day                       ; Store in global page.
                pla                                    ; Pull off Month.
                inc          a                         ; Inc month value for ProDOS8 format.
                asl          a                         ; Shift month as it sits in between
                asl          a                         ; the year and day values.
                asl          a                         ;
                asl          a                         ;
                asl          a                         ;
                ora          day                       ; Put all but the top bit of month
value
                sta          day                       ; in the day byte.
                rol          year                      ; Put hi bit of mo. in lo bit of yr
byte.
                pla                                    ; Pull off unused byte.
                pla                                    ; Pull off Day of Week. Stack now
clean.
*
                sec                                    ; Now go back to emulation mode
                xce                                    ; to continue with ProDOS8.
                rts                                    ; That's all.
*
savestate       dfb          0                         ;Keep the state of state register
                asc          'JIMJAYKERRY&MIKE' ;   '
clock.end       equ          *
                ds           125-clock.end+clock.begin,0 ; Zero rest of 125 bytes.
size            equ          *-clock.begin             ; MUST be $7D (125) bytes in length!

; ###########################################################################################
; #   END OF FILE:  L.CCLOCK
; #   LINES      :  157
; #   CHARACTERS :  8015
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT L.MLI.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.MLI
; ###############################################################################################

                LST          ON,GEN,ASYM,VSYM
                X65816
                XREFSLOT     1
                SEG          $00
************************************************************
*                                                          *
*        PRODOS 8 - PROFESSIONAL DISK OPERATING SYSTEM     *
*                                                          *
*            COPYRIGHT APPLE COMPUTER, INC., 1983-87       *
*                                                          *
*                     ALL RIGHTS RESERVED                  *
*                                                          *
*                      February 3, 1987                    *
*                                                          *
************************************************************
*
* Note that the "os" variable must be set to the system you
* wish to build.
*
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
                INCLUDE      MLI.SRC/REVISIONS
                ORG          $2000
                INCLUDE      MLI.SRC/EQUATES
                INCLUDE      MLI.SRC/PROLDR
                page
                INCLUDE      MLI.SRC/DEVSRCH
                INCLUDE      MLI.SRC/RELOC
                MSB          OFF
                INCLUDE      MLI.SRC/GLOBALS
                INCLUDE      MLI.SRC/XDOSMLI
                INCLUDE      MLI.SRC/BFMGR
                INCLUDE      MLI.SRC/CREATE
                INCLUDE      MLI.SRC/FNDFIL
                INCLUDE      MLI.SRC/NEWFNDVOL
                INCLUDE      MLI.SRC/ALLOC
                INCLUDE      MLI.SRC/POSN.OPEN
                INCLUDE      MLI.SRC/READ.WRITE
                INCLUDE      MLI.SRC/CLOSE.EOF
                INCLUDE      MLI.SRC/DESTROY
                INCLUDE      MLI.SRC/DETREE
                INCLUDE      MLI.SRC/MEMMGR
                INCLUDE      MLI.SRC/DATATBLS
                INCLUDE      MLI.SRC/WRKSPACE
                ifeq         os-EdNet
                include      mli.src/doatalk
                fin
                INCLUDE      MLI.SRC/ROM

; ###############################################################################################
; #   END OF FILE:  L.MLI
```

```
; #   LINES      :  50
; #   CHARACTERS :  2070
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT L.RAM.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.RAM
; ###############################################################################################

               LST          ON,ASYM,VSYM,GEN
               X65816
               XREFSLOT     6
               SEG          $00
*
************************************************************
*                                                          *
*            PRODOS 8 KERNEL 59.5K RAMDISK (REV-E)         *
*                                                          *
*            COPYRIGHT APPLE COMPUTER, INC., 1983-86       *
*                                                          *
*                     ALL RIGHTS RESERVED                  *
*                                                          *
************************************************************
*
**** Note that the "os" variable must be updated depending on
**** the system you are building.
*
ProDOS         equ          0
EdNet          equ          1
os             equ          ProDOS
*
* Creates 3 object files
               SBTL         'EXTENDED 80 COL RAMDISK'
               DSECT
               ORG          $42
CMD            DS           1,0                     ;command from ProDOS
UNIT           DS           1,0                     ;unitnum from ProDOS
BUFPTR         DS           2,0                     ;User Buffer pointer
BLOCK          DS           2,0                     ;block requested
               DEND
*
XDIOERR        EQU          $27
XDWPERR        EQU          $2B                     ;write protect error
A1L            EQU          $3C
A1H            EQU          $3D                     ;SOURCE OF TRANSFER
A2L            EQU          $3E
A2H            EQU          $3F                     ;END OF SOURCE
A3L            EQU          $40
A3H            EQU          $41
A4L            EQU          $42
A4H            EQU          $43                     ;DESTINATION OF TRANSFER
*
MOVE           EQU          $C311                   ;monitor move data routine
XFER           EQU          $C314                   ;monitor XFER control
*
EnblRAM2       EQU          $C083                   ;enable 2nd bank of LC
EnblRAM1       EQU          $C08B                   ;enable 1st bank of LC
*
Store80Off     EQU          $C000
Store80On      EQU          $C001
Store80Sw      EQU          $C018
WRMAINRAM      EQU          $C004                   ;Write data to main ram
```

```
WRCARDRAM       EQU         $C005                   ;Write data to card ram
ALTZP           EQU         $C009                   ;enable alternate ZP,STK,LC
REGZP           EQU         $C008                   ;enable regular ZP,STK,LC
*
ABUF1           EQU         $C00                    ;temporary buffer
VBLOCK1         EQU         $E00                    ;where the Vdir goes
*
EnterCard       EQU         $200                    ;card entry point
PassIt          EQU         $3ED                    ;XFER param
*
                PAGE
                ORG         EnterCard
*
* After the main routine has determined that the command
* is ok, and the block to be read/written is within
* range, it tranfers control to this routine.  This routine
* remaps the block requested as follows:
*     Request blocks 0,1 :invalid
*                    2 :returns VDIR (card block 3)
*                    3 :returns bitmap (synthesized)
*                    4 :returns card block 0
*                 $5-$5F :returns card blocks $5-$5F
*                $60-$67 :returns blocks $68-$7F in bank 1 of
*                           card's language card
*                $68-$7F :returns blocks $68-$7F in bank 2
*                           of card's language card
*
DoCmd           LDA         Store80Sw               ;Read 80STORE
                PHA                                 ;save for later
                STA         Store80Off              ;turn off 80STORE
*
                LDX         #4                      ;move the params for our use
DoCmd1          LDA         CMD,X                   ;CMD,UNIT,BUFPTR,&BLOCK(lo)
                STA         TCMD,X                  ;->TCMD,TUNIT,R2L,R2,R1
                DEX
                BPL         DoCMD1
                AND         FormatFlag              ;Format the volume first time
                BNE         DoCommand               ;thru, or when requested
*
DoFormat        LDX         BLOCK                   ;save R1 during format
                LDA         #<VBLOCK1               ;block to be cleared
                JSR         CLRBUF1                 ;ClrBuf clears all buffers
                LDY         #$3                     ;Format volume in 2 chunks
DF2             LDA         Vdir,Y
                STA         VBLOCK1+4,Y
                DEY
                BPL         DF2
*
                lda         #$FE                    ;Set last block as unusable to protect
vectors
                sta         bitmap+$F
                TYA                                 ;Set bitmap bits to $FF
                LDY         #$E                     ;15 bytes to set
DF2.1           STA         BITMAP,Y
                DEY
                BNE         DF2.1
                STY         BITMAP                  ;first byte=0
*
                LDY         #$7                     ;do other chunk
DF3             LDA         Access,Y
                STA         VBLOCK1+34,Y
                DEY
                BPL         DF3
```

```
               LDA          FormatFlag               ;if 0, set to FF
               BNE          DFX                      ;else exitcard
               STY          FormatFlag               ;Y=FF, won't format next time
               STX          R1                       ;restore R1
*
* Now use the requested block number to determine
* which routine performs the transfer
*
DoCommand      ASL          R1                       ;block requested->page requested
               LDA          R1                       ;get page requested
               CMP          #$BF                     ;in Language card?
               BCS          TLC1                     ;yes, do it
CB1            CMP          #6                       ;bit map?
               BNE          CB2
               JMP          TBMAP                    ;yes, transfer bitmap
CB2            JMP          TREG                     ;else normal transfer
*
* When a block between $60 and $7F is requested, it must
* be spirited into/from the language card area of the
* 64K card.  This requires a two-stage move: into the temp
* buffer and then to its real destination.
*
TLC1           TAX                                   ;save R1 for later
               JSR          SETPTR                   ;get direction
               PHP                                   ;save direction
               BCS          LCWrt                    ;it is a write
LCRd           TXA                                   ;get R1 back
               CMP          #$CF                     ;which bank is it in
               BCS          TLC2                     ;in main bank
*
               ORA          #$10                     ;in secondary bank
               BNE          TLC3                     ;branch always
*
TLC2           STA          EnblRAM2                 ;turn on main $D000
               STA          EnblRAM2
*
TLC3           STA          R1                       ;restore R1
               LDA          R2                       ;save R2 for later
               PHA
               LDX          R2L
               STA          ALTZP                    ;now switch to other ZP
               LDA          #<Abuf1                  ;set R2 to Abuf
               STA          R2
               LDA          #>Abuf1
               STA          R2L
               JSR          SetPtr                   ;set pointers
               TAY                                   ;A=0 from SETPTR
TLC4           LDA          (A1L),Y                  ;move A1,A2 to A4,A3
               STA          (A4L),Y
               LDA          (A2L),Y
               STA          (A3L),Y
               DEY
               BNE          TLC4
*
               STA          REGZP                    ;restore normal ZP
               STX          R2L
               PLA                                   ;restore R2
               STA          R2
               PLP                                   ;get direction
DFX            BCS          XLCWrt                   ;write, done with move
*
               STA          EnblRAM1                 ;now switch MLI part of LC in
               STA          EnblRAM1
```

```
                JSR         BLOCKDO0                ;read, transfer Abuf to main
XLCWrt          JMP         ExitCard
*
LCWrt           JSR         BLOCKDO0                ;transfer main to Abuf
                JMP         LCRd                    ;transfer Abuf to Lang card
*
* BLOCKDO transfers a block between main memory and the
*  64K card.  R1 contains the page address of the block
*  in the card; R2 contains the page address of the block
*  in main memory.  The address in main memory is always
*  in the language card, so the language card is always
*  switched in.  If CMD is 2, a write is done (R2->R1);
*  if CMD is 1, a read is done (R1->R2).
*
BLOCKDO0        LDA         #<ABUF1                 ;set up R1 = Abuf
BLOCKDO1        STA         R1
BLOCKDO         JSR         SETPTR                  ;set pointers
                BCS         BlockWrite              ;it's a write
                STA         WrMainRAM               ;transfer buffer directly to main ram
                TAY                                 ;0 left from SETPTR
BD1             LDA         (A1L),Y                 ;transfer A1,A2 to A4,A3
                STA         (A4L),Y
                LDA         (A2L),Y
                STA         (A3L),Y
                DEY
                BNE         BD1
                STA         WrCardRAM               ;back the way it was
DoneWrt         RTS                                 ;MainWrt returns here
*
BlockWrite      LDA         #>MainWrt               ;pointers set up,
                STA         PassIt                  ;pass control to main ram
                LDA         #<MainWrt
                JMP         Ex1                     ;set PassIt+1 and transfer
*
* SETPTR is used by other routines to set up pointers
* and to detect read or write.
*
SETPTR          LDA         TCMD                    ;the rest depends on read
                LSR         A                       ;or write.  Which is it?
                BCS         CMDWRT                  ;it's write
*
CMDRD           LDA         R2                      ;get dest page
                STA         A4H                     ;1st dest page (MOVE)
                STA         A3H                     ;2nd dest page
                LDA         R2L                     ;low byte dest page
                STA         A4L                     ;1st dest page low
                STA         A3L                     ;2nd dest page low
                LDA         R1                      ;get source page
                STA         A1H                     ;1st source page
                STA         A2H                     ;2nd source page
                LDA         #0                      ;source page aligned
                STA         A1L                     ;1st source page
                STA         A2L                     ;2nd source page
                BEQ         CMDBOTH                 ;update second pages
*
CMDWRT          LDA         R2                      ;get source page
                STA         A1H                     ;1st source page
                STA         A2H                     ;2nd source page
                LDA         R2L                     ;get source page low
                STA         A1L                     ;1st source page low
                STA         A2L                     ;2nd source page low
                LDA         R1                      ;get dest page
                STA         A4H                     ;1st dest page
```

```
                STA         A3H                     ;2nd dest page
                LDA         #0                      ;dest page aligned
                STA         A4L                     ;1st dest page
                STA         A3L                     ;2nd dest page
*
CMDBOTH         INC         A2H                     ;update 2nd source page
                INC         A3H                     ;update 2nd dest page
                RTS
*
* TZIP is called if Blocks 0,1,4,5 are requested.
* On write it does nothing, on read, it returns 0's
*
TZIP            JSR         CLRBUF0                 ;fill ABUF with 0's
                JSR         BLOCKDO                 ;transfer them 0's
ZIPOUT          JMP         ExitCard                ;and return
*
* CLRBUF fills the buffer indicated by R1 to 0's
* Should only be called on a read or format.
*
CLRBUF0         LDA         #<ABUF1                 ;ABUF is temp buffer
CLRBUF1         STA         R1                      ;assign to BLOCK
CLRBUF2         JSR         SETPTR                  ;set pointers
                TAY                                 ;A set to 0 by setptr
CLRBUF3         STA         (A1L),Y
                STA         (A2L),Y
                DEY
                BNE         CLRBUF3
                RTS
*
* TREG maps the requested block into the aux card
* so that 8K data files will be contiguous (the index
* blocks will not be placed within data).
*
TREG            CMP         #4                      ;page 4 = vdir
                BNE         T1                      ;not vdir, continue
                LDA         #$7                     ;else xfer block 7
                BNE         GoTimes2
*
***************** See Rev Note #43 ********************
*
T1              CMP         #$f                     ;if any page<f (<block 8) requested
*****************************************************
                BCC         TZIP                    ;it is invalid
*
TREG1           LDX         #0                      ;X contains number of iterations
                LDA         BLOCK                   ;use true block number
                CMP         #$5D                    ;beyond 8K blocks?
                BCC         TR1                     ;no, do normal
                SBC         #$50                    ;else subtract offset
GoTimes2        JMP         Times2                  ;and multiply by 2
*
* Determine which 8K chunk it is in, place in X;
* block offset into chunk goes into Y.
*
TR1             SEC
                SBC         #$8                     ;block = block -6
TR2             CMP         #$11                    ;if <=17, done
                BCC         TR3                     ;yup, got iteration
                SBC         #$11                    ;else block =block -17
                INX                                 ;count iteration
                BPL         TR2                     ;branch always
                DFB         $00                     ;just in case
TR3             TAY                                 ;remainder in Y
```

```
*
* If remainder is 1 it's an index block: start index
* blocks at $1000,$2000..$19FF)
*    If remainder is 0, it is first data block in 8K
* chunk.  Page is 32 + (16 * X).
*    Otherwise, it is some other data block.
* Page is 32 + (16 * X) + (2 * Y)
*
                CPY         #1                      ;is it index block?
                BNE         TR4                     ;no
                TXA                                 ;index = 2*(8+X)
                CLC
                ADC         #8
                BNE         Times2                  ;multiply by 2
TR4             INX                                 ;need iteration+1
                TXA                                 ;page = 2 * (16 + 8X)
                ASL         A
                ASL         A
                ASL         A
                ASL         A
                STA         R1
                TYA                                 ;get offset into 8K chunk
                BEQ         TR5                     ;if 0, no offset
                DEY                                 ;else offset = 2 * Y
                TYA
TR5             CLC
                ADC         R1
Times2          ASL         A                       ;acc=2*acc
                JSR         BLOCKDO1                ;store in R1 and xfer
                JMP         ExitCard                ;and return
*
* When Block 3 is requested, the bitmap is returned.  The
* Real bitmap is only 16 bytes long (BITMAP); the rest of
* the block is synthesized.  The temporary buffer at $800
* is used to build/read a full size bitmap block.
*
TBMAP           LDA         #<ABUF1                 ;use temporary buffer as BLOCK
                STA         R1
                JSR         SETPTR                  ;Set pointers/test read-write
                BCS         BITWRT                  ;its a write!
*
BITRD           JSR         CLRBUF2
*
                LDY         #$F                     ;now put real bitmap there
BITRD2          LDA         BITMAP,Y
                STA         (A1L),Y
                DEY
                BPL         BITRD2
                JSR         BLOCKDO                 ;move temp buf to user buf
                JMP         ExitCard
*
BITWRT          JSR         BLOCKDO                 ;move user buf to temp buf
                JSR         SETPTR                  ;Set pointers
                LDY         #$F                     ;move temp buf to bitmap
BITWRT1         LDA         (A4L),Y                 ;(pointer set by SETPTR)
                STA         BITMAP,Y
                DEY
                BPL         BITWRT1
                JMP         ExitCard
*
FormatFlag      DFB         0                       ;not formatted yet
*
TCMD            DS          1,0                     ;command byte
```

```
TUNIT           DS          1,0                     ;unit byte (not used)
R2L             DS          1,0                     ;low byte of user buffer
R2              DS          1,0                     ;hi byte of user buffer
R1              DS          1,0                     ;page requested
*
BITMAP          DFB         $00,$FF,$FF,$FF         ;blocks 0-7 used
                DFB         $FF,$FF,$FF,$FF
                DFB         $FF,$FF,$FF,$FF
                DFB         $FF,$FF,$FF,$FE
*
Vdir            EQU         *                       ;start of vdir
Type_NameL      DFB         $F3                     ;storage type F, namelength 3
                MSB         OFF
VName           ASC         "RAM"
*
Access          DFB         $C3                     ;Destroy, Rename, Read enabled
Entry_Len       DFB         $27                     ;entry length
Entries_Per_Block DFB         $0D
File_Count      DFB         0,0
Map_Pointer     DFB         3,0                     ;Block 3
Total_Blocks    DFB         $7f                     ;128 blocks
*
ExitCard        LDA         EnblRAM1                ;restore lang card
                LDA         EnblRAM1
                PLA                                 ;get 80STORE
                BPL         Ex0                     ;80STORE wasn't on
                STA         Store80On
Ex0             JMP         $3EF                    ;Jump around PassIt (3ED,3EE)
                DS          $3EF-*,0                ;pad thru $3EE
                LDA         #>NoErr                 ;set up return to NoErr
                STA         PassIt
                LDA         #<NoErr
Ex1             STA         PassIt+1                ;also used by BlockWrite
                CLC                                 ;transfer card to main
                CLV                                 ;use standard zp/stk
                JMP         XFER                    ;there's no place like home...
*  NOTE: The previous section of code MUST NOT use $3FE & $3FF
*        since the Interrupt Vector must go there if AUX interrupts
*        are to be used.
                PAGE
*
* Transfer card part of the driver to the card
* Transfer front part of the driver to lang.card
*
RAMDest         EQU         $0200
RAMDest1        EQU         RAMDest+$100
LCDest          EQU         $FF00
*
                ifeq        os-ProDOS
*
**************** See Rev Note #60 *****************
*
Srce            equ         $2b00                   ; Note: THIS MUST CHANGE WHEN THINGS
GROW!
************************************************
                fin
                ifeq        os-EdNet
Srce            equ         $2b00                   ; The loader is bigger in EdNet's
Version.
                fin
RAMSrc          EQU         Srce+$100
RAMSrc1         EQU         Srce+$200
LCSrc           EQU         Srce+$300
```

```
*
DEVNUM          EQU          $B0                        ;Slot 3, Drive 2 ($30 + hi bit set)
DEVADR          EQU          $BF10                      ;Base of Device addresses
DEVCNT          EQU          $BF31
DEVLST          EQU          $BF32
*
                ORG          Srce
*
* Move LCSrc to LCDest
*
                LDY          #$99                       ;move $9A bytes
MvLC            LDA          LCSrc,Y                    ;get a byte of source
                STA          LCDest,Y
                DEY
                CPY          #$FF
                BNE          MVLC
*
* Move RAMSrc to RAMDest
*
                LDX          #>RAMSrc
                STX          A1L                        ;source low
                DEX                                     ;source end low
                STX          A2L
                LDX          #<RAMSrc                   ;source high
                STX          A1H
                INX
                STX          A2H                        ;end high
                LDA          #>RAMDest
                STA          A4L
                LDA          #<RAMDest
                STA          A4H
                SEC                                     ;RAM to Card
                JSR          MOVE
*
* Now install it into the system
*
                LDA          #>LCDest                   ;put LC address into
                STA          DEVNUM/8+DEVADR            ;slot 3, drive 2
                LDA          #<LCDEST
                STA          DEVNUM/8+DEVADR+1
                INC          DEVCNT
                LDX          DEVCNT
                LDA          #DEVNUM+$F                 ;unit num of /RAM
                STA          DEVLST,X
                RTS
                PAGE
*
                ORG          LCDest
EnterRAM        EQU          *                          ;/RAM entry point
                CLD
*
                LDX          #$B                        ;save 13 bytes of params
SAVPARMS        LDA          A1L,X
                STA          A1L1,X
                DEX
                BPL          SAVPARMS
*
                LDX          #1                         ;save XFER Vectors too
SVP1            LDA          PassIt,X
                STA          SP1,X
                DEX
                BPL          SVP1
*
```

```
                LDA         CMD                     ;get command
                BEQ         STAT                    ;0=STATUS
                CMP         #4                      ;check for command too high
                BCS         IOERR                   ;if it is, IO ERR
                EOR         #$3                     ;0=FORMAT,2=READ,1=WRITE
                STA         CMD                     ;CMD=>0=Format,2=Read,1=Write
                BEQ         FORMAT                  ;format the volume
                LDY         BLOCK+1                 ;check for enormous blocknum
                BNE         IOERR                   ;io error if too big
                LDA         BLOCK                   ;get the block number
                BMI         IOERR                   ;largest block is $7F
*
* At this point, control is passed to the code in the
* alternate 64K.  It is used for read, write, and
* format.  After the request is completed, control
* is always passed back to NoErr.
*
Format          EQU         *
*
                LDA         #>EnterCard             ;card entry point
                STA         PassIt                  ;figure it out on card
                LDA         #<EnterCard
GoCard          STA         PassIt+1                ;also used by MainWrt
                SEC                                 ;RAM->Card
                CLV                                 ;start with original z.p.
                JMP         XFER                    ;transfer control
*
*
IOERR           LDA         #XDIOERR                ;get err num
                BNE         ERROUT                  ;and return
WPERR           LDA         #XDWPERR                ;write protect err
ERROUT          SEC                                 ;flag error
                BCS         Restore                 ;restore cmd and unitnum
*
STAT            EQU         *
NoErr           LDA         #0                      ;no error
                CLC                                 ;flag no error
Restore         PHP                                 ;save status
                PHA                                 ;save error code
*
                LDX         #$B                     ;restore 13 bytes of params
RSTPRMS         LDA         A1L1,X
                STA         A1L,X
                DEX
                BPL         RSTPRMS
*
                LDA         SP1                     ;restore XFER params
                BIT         $6060                   ;This instruction is to put an RTS at
$FF58 as in ROM
                STA         PassIt
                LDA         SP1+1
                STA         Passit+1
*
; -------------------- See rev note 21 -----------------------
                PLA                                 ;get error
                PLP                                 ;get status
; ------------------------------------------------------------
                RTS                                 ;and return
*
MainWrt         STA         WrCardRAM               ;xfer data to card
                LDY         #0
MW1             LDA         (A1L),Y                 ;pointers set in card by SETPTR
                STA         (A4L),Y
```

```
                LDA           (A2L),Y
                STA           (A3L),Y
                DEY
                BNE           MW1
                STA           WrMainRAM                 ;done writing Card
*
                LDA           #>DoneWrt
                STA           PassIt
                LDA           #<DoneWrt
                JMP           GoCard
*
SP1             DS            2,0
A1L1            DS            $C,0                      ;13 bytes of storage
FrontLen        EQU           *-EnterRAM
*
***************** See Rev Note #EN1 *****************
*
                ifeq          os-EdNet
                lda           $c083                     ; Switch in bank 2.
                jsr           $d400                     ; Go do RWTS.
                sta           $c08b                     ; Switch bank 1 back in preserving
status!
                rts                                     ; Back to caller.
wherearewe      equ           *
                fin
*************************************************

; ###############################################################################################
; #   END OF FILE:  L.RAM
; #   LINES      :  578
; #   CHARACTERS :  27268
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT L.SEL.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.SEL
; ###############################################################################################

                LST          ON,ASYM,VSYM,GEN
                X65816
                XREFSLOT     6
                SEG          $00
*
************************************************************
*                                                          *
*          PRODOS 8 LOBOTOMIZED DISPATCHER ROUTINE         *
*                                                          *
*           COPYRIGHT APPLE COMPUTER, INC., 1983-86        *
*                                                          *
*                     ALL RIGHTS RESERVED                  *
*                                                          *
************************************************************
*
                SBTL         "DISPATCHER I" "
************************************************************
*
*  DISPATCHER 1 - This code ORGs and operates at $1000 but
*    is resident in memory at $D100 in the Alt 4K bank of the
*    Language Card.  The QUIT call vectors to a routine high
*    in the MLI that moves DISPATCHER 1 down and jumps to it.
*    The move routine MUST remain somewhere between $E000-$F7FF.
*
*  NOTE: This entire routine MUST remain no larger than 3 pages.
*
************************************************************
                ORG          $1000
                MSB          ON
PROMPT          EQU          $FF
CV              EQU          $25
CH              EQU          $24
INFOCALL        EQU          $C4
SETPFXCALL      EQU          $C6
GETPFXCALL      EQU          $C7
OPENCALL        EQU          $C8
READCALL        EQU          $CA
CLOSECALL       EQU          $CC
EOFCALL         EQU          $D1
ERRNUM          EQU          $DE                     ;Applesoft ERRNUM but who cares?
PATH            EQU          $280                    ;Second half of input buffer
*
line3           equ          $600                    ; 3rd line of screen (starting from 0).
*
ENTRY           EQU          $2000
MLI             EQU          $BF00
BITMAP          EQU          $BF58
PFIXPTR         EQU          $BF9A
KBD             EQU          $C000
STB             EQU          $C010
RAMIN           EQU          $C08B
ROMIN           EQU          $C082
RAMIN2          EQU          $C083
```

```
CLEOL           EQU          $FC9C                    ;Clear to End of Line
settxt          equ          $fb39                    ; Monitor routine to set 40 col window.
HOME            EQU          $FC58
RDKEY           EQU          $FD0C
COUT            EQU          $FDED
CROUT           EQU          $FD8E
BELL            EQU          $FF3A
*
* Monitor equates, soft switch equates for fix #55....
*
setvid          equ          $fe93                    ; Puts COUT1 in CSW.
setnorm         equ          $fe84                    ; Normal white text on black backround.
init            equ          $fb2f                    ; Text pg1;text mode;sets 40/80 col
window.
setkbd          equ          $fe89                    ; Does an IN#1.
clr80vid        equ          $c00c                    ; Disable 80 column hardware.
clraltchar      equ          $c00e                    ; Switch in primary character set.
clr80col        equ          $c000                    ; Disable 80 column store.
*
*  Reset the Rev e soft switches
*
*********** See rev note #55 ***************
HEREIN          LDA          ROMIN                    ;Switch in Bank 1 RAM, then ROM
* JSR $FE93 ;SETVID
* JSR $FE89 ;SETNORM
* STA $C00C ;80 col off
* STA $C00F ;Alt chrset on
* STA $C000 ;Main ram
                sta          clr80vid                 ; Disable 80 column hardware.
                sta          clraltchar               ; Switch in primary char set.
                sta          clr80col                 ; Disable 80 column store.
                jsr          setnorm                  ; Normal white chars on black
backround.
                jsr          init                     ; Text pg1; text mode; set 40 col
window.
                jsr          setvid                   ; Does a PR#0 (puts COUT1 in CSW).
                jsr          setkbd                   ; Does an IN#0 to set Basic input to
kbd.
*************************************************
*
*  Clear the memory Bit Map
*
CLRMAP          LDX          #$17                     ;Do all the bytes
                LDA          #1
                STA          BITMAP,X                 ;Protect page $BF00
                DEX
                LDA          #$0                      ;Clear the rest
CLRLOOP         STA          BITMAP,X
                DEX
                BPL          CLRLOOP
                LDA          #$CF
                STA          BITMAP                   ;Protect pages 0,1 & $400-$7FF (Screen)
START           EQU          *
*
***************** See Rev Note #39 *********************
*
* jsr settxt ; No longer necessary with fix #55...
*************************************************
                JSR          HOME                     ; Clear the screen
                JSR          CROUT
*
******************** Rev Note #55  ********************
*
```

```
                ldx         #>msg0-msgstart          ; Load offset to message into x...
*************************************************
*
                JSR         PRNTLOOP
                LDA         #3                       ;Set CV to 3rd line
                STA         CV
                JSR         CROUT                    ;  and col 1
                JSR         MLI                      ;Call the MLI (Remember, this code
executes at $1000)
                DFB         GETPFXCALL
                DW          PREFIX
                LDX         PATH                     ;Get PREFIX length
                LDA         #0                       ;Put a 0 at end of Prefix
                STA         PATH+1,X
*
****************** See Rev Note #69 ******************
*
                ldx         path                     ; Get length byte back.
                beq         nilpfx                   ; Branch if no prefix to display!!
*******************************************************
*
********************* Rev Note #55  ******************
*
*LDA #0 ;Put a 0 at end of Prefix
*STA PATH+1,X
*LDA #>PATH+1 ;Put the prefix address in print loop
*STA LOOP+1
*LDA #<PATH
*STA LOOP+2
*JSR PRNTLOOP
m1              lda         path,x                   ; Display prefix directly
                ora         #$80                     ; Set hi bit for NORMAL text.
                sta         line3-1,x                ; to the screen.
                dex                                  ; Next....
                bne         m1                       ; Branch until prefix displayed.
*************************************************
*
****************** See Rev Note #69 ******************
*
nilpfx          equ         *
*************************************************
                LDX         #0
                DEC         CV                       ;
                JSR         CROUT                    ;Put the cursor on the first char
GETKEY          JSR         RDKEY                    ;Wait for keyboard input
                CMP         #$8D                     ;Is it CR?
                BEQ         GOTPFX                   ;Yes, and we accept what was entered
                PHA                                  ;No, save the char
                JSR         CLEOL                    ;Clear rest of line
                PLA                                  ;Get char back
                CMP         #$9B                     ;Is it ESC?
                BEQ         START                    ;Yes, start over again
                CMP         #$98                     ;If it is CTRL-X, start over.
RESTRT          BEQ         START                    ;(Used as an extended BEQ from PRMPT)
                CMP         #$89                     ;Is it TAB?
                BEQ         BADKEY                   ;No good if it is!
                CMP         #$FF                     ;Delete?
                BEQ         X2                       ;Branch if it is
                CMP         #$88                     ;Back Space?
                BNE         NOTBS                    ;Branch if not
X2              CPX         #$0                      ;If it is, are we at col 0?
                BEQ         *+5                      ;If col 0, do nothing
                DEC         CH                       ; else move left 1 char
```

```
                DEX                                     ; decrement char count,
                JSR         CLEOL                       ; clear rest of line
                JMP         GETKEY                      ;Go get another char
NOTBS           BCS         MAYBE
BADKEY          JSR         BELL                        ;Ring the speaker (bell) if it isn't.
                JMP         GETKEY
MAYBE           CMP         #$DB                        ;Ok, is it below 'Z'?
                BCC         *+4                         ;Branch if yes
                AND         #$DF                        ;If not, shift it up upper case
                CMP         #$AE                        ;Is it below '.'?
                BCC         BADKEY                      ;If yes, it ain't good!
                CMP         #$DB                        ;Is it above 'Z'?
                BCS         BADKEY                      ;If so, it also ain't good
                CMP         #$BA                        ;Is it below ':'? ('.' - '9' range)
                BCC         GOODKEY                     ;Yes, it's good!
                CMP         #$C1                        ;If not, is it at or above 'A'? ('A' -
'Z')
                BCC         BADKEY                      ;No, reject it
GOODKEY         INX                                     ;It's OK.  Hallelulah!
                CPX         #39                         ;Were there more than 39 chars?
                BCS         RESTRT                      ;Yes, too many!  Go restart.
                STA         PATH,X                      ;No, save the lucky char
                JSR         COUT                        ;Print it
                JMP         GETKEY                      ;  and go get another.
GOTPFX          EQU         *
                CPX         #$0                         ;OK, is our Prefix length (chars
entered)=0?
                BEQ         PRMPT                       ;If yes, don't bother re-setting it
                STX         PATH                        ;Set prefix length
                JSR         MLI                         ;Call the MLI
                DFB         SETPFXCALL
                DW          PREFIX
                BCC         PRMPT                       ;If ok, go get Filename
                JSR         BELL                        ;If not, ring Bell
                LDA         #0                          ;  and try again
BADPFX          BEQ         RESTRT                      ;Z flag must be set for extended Branch
PRMPT           JSR         HOME                        ;Clear the screen for application name
PRMPT1          JSR         CROUT                       ;Output a CR
*
********************* Rev Note #55  *******************
*
                ldx         #>msg-msgstart              ; Load offset to message into x...
*****************************************************
*
                JSR         PRNTLOOP
retryrich       LDA         #3                          ;Set CV to 3rd line
                STA         CV
                JSR         CROUT                       ;  and col 1
                LDX         #0
*
********************* Rev Note #69  *******************
*
*LOOP1 LDA #PROMPT ;Our cursor char
* JSR COUT ;Print it
* DEC CH ;  and point to it so a typed
* LDA KBD ;  character can replace it.
*BPL *-3 ;(We'll wait here till a keypress.)
* STA STB ;Hit the strobe
loop1           equ         *
                jsr         rdkey
                CMP         #$9B                        ;ESC
                BNE         NOTESC
                LDA         CH
```

```
                BNE         PRMPT
                BEQ         BADPFX                  ;If ESC in col 0 go get PREFIX again
NOTESC          CMP         #$98                    ;CTRL-X
EXTNDBR         BEQ         PRMPT                   ;(Used as a branch extender)
                CMP         #$89                    ;TAB
                BEQ         NOTGUD
                CMP         #$FF                    ;Delete?
                BEQ         X3
                CMP         #$88                    ;BACK SPACE
                BNE         X1
X3              JMP         EATEM                   ;Eat the previous character.
X1              BCS         GETIN1                  ;> $88 and the char may be acceptable
NOTGUD          JSR         BELL                    ;Ring the bell (speaker)
                JMP         LOOP1
GETIN1          CMP         #$8D                    ;Is it a CR?
                BEQ         DONE
                CMP         #$DB                    ;> than Z
                BCC         *+4                     ;No.
                AND         #$DF                    ;Make sure its Upper case
                CMP         #$AE                    ;Is it "."?
                BCC         NOTGUD                  ;Branch if less
                CMP         #$DB                    ;Must be less than "[".
                BCS         NOTGUD
                CMP         #$BA                    ;OK if less than or equal to "9"
                BCC         ITSGUD
                CMP         #$C1                    ;Else must be > than "A"
                BCC         NOTGUD
ITSGUD          EQU         *
                PHA
                JSR         CLEOL
                PLA
                JSR         COUT                    ;No, print it
                INX
                CPX         #39
                BCS         EXTNDBR
                STA         PATH,X
                JMP         LOOP1                   ;Go get the next one
DONE            EQU         *
                LDA         #'                      '
                JSR         COUT                    ;After the CR, blank out the cursor.
                STX         PATH                    ;Put the length in front of the name.
*
*  At this point the specified Pathname is in PATH ($280)
*    and we can do a GET_FILE_INFO on it.
*
                JSR         MLI
                DFB         INFOCALL
                DW          INFO
                BCC         INFOOK
                JMP         ERROR
INFOOK          LDA         TYPE
                CMP         #$FF                    ;Is it a type SYS file?
                BEQ         DOIT
                LDA         #1                      ;Not SYS File
                JMP         ERROR
DOIT            EQU         *                       ;It's a type SYS all right!
                LDA         #0
                STA         CLSNUM
                JSR         MLI
                DFB         CLOSECALL               ;CLOSE all open files first
                DW          CLS
                BCC         CHKACS
                JMP         ERROR
```

```
*
*  Now check for the proper access
*
CHKACS          LDA         ACESS                   ;Get the allowed access
                AND         #1                      ;Is READ disabled?
                BNE         ACSOK                   ;No. Access ok.
                LDA         #$27                    ;I/O error
                JMP         ERROR                   ;Never returns!
ACSOK           EQU         *
                JSR         MLI
                DFB         OPENCALL
                DW          OPN                     ;OPEN it.
                BCC         *+5
                JMP         ERROR
                LDA         REFNUM
                STA         REEDNUM                 ;Spread REFNUM around
                STA         EOFNUM
*
*  Ok it's OPEN, let's get the EOF
*
                JSR         MLI
                DFB         EOFCALL
                DW          EOF
                BCS         ERROR
                LDA         EOFB+2                  ;3rd of 3 bytes
                BEQ         EOFOK
                LDA         #$27                    ;I/O ERROR even though the file is
                BNE         ERROR                   ;  simply too large
EOFOK           LDA         EOFB                    ;Move EOF to Read # bytes
                STA         RCOUNT
                LDA         EOFB+1
                STA         RCOUNT+1
                JSR         MLI
                DFB         READCALL                ;Do the READ
                DW          REED
                PHP                                 ;Push the processor status
                JSR         MLI
                DFB         CLOSECALL               ;Close it
                DW          CLS
                BCC         *+6
                PLP                                 ;Get status back (it is irrevalent now)
                BNE         ERROR                   ;(if CLOSE generated an error)
                PLP                                 ;We're here if CLOSE was OK
                BCS         *-4                     ;JMP ERROR
                JMP         ENTRY
*
EATEM           EQU         *
                LDA         CH                      ;Is the cursor in col 0?
                BEQ         EATEMBAK                ;Yes, ignore it.
                DEX
                LDA         #'                      '
                JSR         COUT                    ;Blank out the cursor
                DEC         CH                      ;Point to last character
                DEC         CH                      ;  entered...
                JSR         COUT                    ;  and blank it too.
                DEC         CH                      ;Point to that location
EATEMBAK        JMP         LOOP1                   ;Go back & get the next char
*
****************** See Rev Note #55  *****************
*
*PRNTLOOP LDX #0 ;Index into message
*LOOP LDA *,X ;CAUTION:  SELF-MODIFYING CODE
* BEQ LOOPRTN ;  ADDR FILLED W/ LOC OF MESSAGE.
```

```
* ORA #$80
* JSR COUT ;Print message on screen
* INX
* BNE LOOP ;Loop til done (rotisserie mode)
*LOOPRTN RTS
*
prntloop        equ          *
                lda          msgstart,x                ; Display string; offset is in X.
                beq          done1                     ; Branch if done.
                jsr          cout                      ; Output character...
                inx                                    ; Next..
                bne          prntloop                  ; Branch always.
done1           rts                                    ; Done.
************************************************
ERROR           EQU          *
                STA          ERRNUM
                LDA          #$0C                      ;Put error message on line 13
                STA          CV
                JSR          CROUT
                LDA          ERRNUM
                CMP          #1
                BNE          NEXTERR
*
*************** See Rev Note #55  **************
*
* LDA #>ERR1
* STA LOOP+1
* LDA #<ERR1
* STA LOOP+2
*
                ldx          #>err1-msgstart           ; Load x with offset to message.
*********************************************
                BNE          DOERROR
NEXTERR         CMP          #$40
                BEQ          ERROR3
                CMP          #$44
                BEQ          ERROR3
                CMP          #$45
                BEQ          ERROR3
                CMP          #$46
                BEQ          ERROR3
*
*************** See Rev Note #55  **************
*
* LDA #>ERR2
* STA LOOP+1
* LDA #<ERR2
* STA LOOP+2
                ldx          #>err2-msgstart           ; Load x with offset to message.
*********************************************
                BNE          DOERROR
*
*************** See Rev Note #55  **************
*
*ERROR3 LDA #>ERR3
* STA LOOP+1
* LDA #<ERR3
* STA LOOP+2
error3          equ          *
                ldx          #>err3-msgstart           ; Load x with offset to message.
*********************************************
DOERROR         JSR          PRNTLOOP
*
```

```
******************** Rev Note #69  ******************
*
* LDA #0 ; Printloop will leave A=0...See Rev Note #39
*STA CV
                JMP          retryrich
*
************************************************************
msgstart        equ          *
                MSB          ON
MSG0            ASC          'ENTER PREFIX (PRESS "RETURN" TO ACCEPT)'
                DFB          0
MSG             ASC          "ENTER PATHNAME OF NEXT APPLICATION" "
                DFB          0
ERR1            DFB          $87                        ;BELL
                ASC          'NOT A TYPE "SYS" FILE'
                DFB          0
ERR2            DFB          $87
                ASC          'I/O ERROR            ' '
                DFB          0
ERR3            DFB          $87
                ASC          'FILE/PATH NOT FOUND  ' '
                DFB          0
*
************************************************************
*
INFO            DFB          $A                         ;10 PARAMETERS ON GFI
                DW           PATH                       ;Pathname buffer pointer
ACESS           DFB          0                          ;ACCESS
TYPE            DFB          0                          ;File Type
                DS           $0D,0                      ;All the rest are unimportant
*
OPN             DFB          3                          ;3 parameters on an OPEN
                DW           PATH
                DW           $1800                      ;FCB Buffer
REFNUM          DFB          0
*
CLS             DFB          1
CLSNUM          DFB          0                          ;REFERENCE #
*
REED            DFB          4                          ;4 Parameters for a READ
REEDNUM         DFB          0
                DW           ENTRY                      ;SYS files always load at $2000
RCOUNT          DW           0
                DW           0
*
EOF             DFB          2
EOFNUM          DFB          0
EOFB            DS           3,0                        ;Three byte EOF
*
PREFIX          DFB          1
PBUF            DW           PATH
*
ZZSIZ           EQU          *-HEREIN
ZZFRE           EQU          $2FF-ZZSIZ
*

; ###############################################################################################
; #   END OF FILE:  L.SEL
; #   LINES      :  479
; #   CHARACTERS :  22188
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT L.TCLOCK.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.TCLOCK
; ###############################################################################################

                LST             ON,ASYM,VSYM,GEN
                X65816
                XREFSLOT        6
                SEG             $00
*
*************************************************************
*                                                           *
*          PRODOS 8 CLOCK DRIVER INTERFACE ROUTINE          *
*                                                           *
*           COPYRIGHT APPLE COMPUTER, INC., 1983-86         *
*                                                           *
*                     ALL RIGHTS RESERVED                   *
*                                                           *
*************************************************************
*
ProDOS          equ             0
EdNet           equ             1
os              equ             ProDOS
*
                ifeq            os-ProDOS
                org             $d742
                fin
                ifeq            os-EdNet
                org             $f842
                fin
                SBTL            "ProDOS Clock Interface"
****************************************
*    CLOCK - PRODOS INTERFACE ROUTINE
*          BY JAMES R. HUSTON
*     WARNING: HARD CODED FOR SLOT 1
****************************************
                SKP             1
TENS            EQU             $3A                     ;NO CONFLICT SINCE MONTH IS LAST
PROCESSED.
MONTH           EQU             $3A
WKDAY           EQU             $3B
DAY             EQU             $3C
HOUR            EQU             $3D
MINUTE          EQU             $3E
                SKP             1
WTTCP           EQU             $C10B
RDTCP           EQU             $C108                   ;CLOCK READ ENTRY POINTS.
CLKMODE         EQU             $538                    ;(+$CN=$5F8+N)
                SKP             1
DATE            EQU             $BF90                   ;PRODOS DATE
TIME            EQU             $BF92                   ;PRODOS TIME.
                SKP             1
INBUF           EQU             $200                    ;INPUT BUFFER.
                SKP             1
READCLK         LDX             CLKSLT                  ;PRESERVE CURRENT MODE FOR THUNDERCLOCK
                LDA             CLKMODE,X
                PHA
                LDA             #$A3                    ;SEND NUMERIC MODE BYTE TO THUNDERCLOCK
```

```
                JSR         WTTCP
CLKSLT          EQU         *+2
                JSR         RDTCP                   ;READ MONTH, DAY OF WEEK, DAY OF MONTH,
AND TIME
                CLC                                 ; INTO INPUT BUFFER.
                LDX         #4                      ;INDEX FOR 5 VALUES
                LDY         #$C                     ;READ MINUTES FIRST, MONTH LAST.
CONVRT          LDA         INBUF,Y                 ;CONVERT VALUES TO BINARY.
                AND         #$7                     ;NO VALUE > 5 DECIMAL.
                STA         TENS                    ;MULTIPLY 'TENS' PLACE VALUE.
                ASL         A
                ASL         A
                ADC         TENS                    ;NOW IT'S TIMES 5.
                ASL         A                       ;NOW IT IS TIMES 10!
                ADC         INBUF+1,Y               ;ADD TO ASCII 'ONES' PLACE
                SEC                                 ;AND SUBTRACT OUT THE ASCII...
                SBC         #$B0
                SKP         1
CNVRT2          STA         MONTH,X                 ;SAVE CONVERTED VALUE.
                DEY                                 ;INDEX TO NEXT LOWER VALUE.
                DEY
                DEY
                DEX                                 ;ARE THERE MORE VALUES?
                BPL         CONVRT                  ;BRANCH IF THERE ARE.
                TAY                                 ;ACC STILL CONTAINS MONTH, SAVE IN Y
FOR NOW.
                LSR         A
                ROR         A
                ROR         A
                ROR         A                       ;(HI BIT OF MONTH HELD IN CARRY)
                ORA         DAY
                STA         DATE                    ;SAVE LOW VALUE OF DATE.
                PAGE
                PHP                                 ;SAVE HI BIT OF MONTH FOR NOW.
                AND         #$1F                    ;ISOLATE DAY AGAIN.
; (WHEN MONTH >7 CARRY SET ACCOUNTED FOR IN FOLLOWING ADD)
                ADC         TDAYS-1,Y               ;REMEMBER THAT Y=MONTH.
                BCC         CNVRT3                  ;BRANCH NOT SEPT 13 THRU 30.
                ADC         #3                      ;ADJUST FOR MOD 7 WHEN DAY > 256.
CNVRT3          SEC
CNVRT4          SBC         #7
                BCS         CNVRT4                  ;LOOP UNTIL LESS THAN 0.
                ADC         #7                      ;NOW MAKE IT IN THE RANGE OF 0-6.
                SBC         WKDAY                   ; THE DELTA PROVIDES YEARS OFFSET.
                BCS         CNVRT5                  ;BRANCH IF POSITIVE.
                ADC         #7                      ;ELSE MAKE IT POSITIVE AGAIN.
CNVRT5          TAY                                 ;LOOK UP YEAR!
                LDA         YRADJ,Y
                PLP                                 ;LASTLY, COMBINE WITH HI BIT OF MONTH.
                ROL         A
                STA         DATE+1                  ;AND SAVE IT.
                LDA         HOUR
                STA         TIME+1                  ;MOVE HOUR AND MINUTE TO PRODOS
GLOBALS.
                LDA         MINUTE
                STA         TIME
                PLA
                LDX         CLKSLT                  ;RESTORE PREVIOUS MODE.
                STA         CLKMODE,X
                RTS                                 ;ALL DONE...
                SKP         2
TDAYS           DFB         $0,$1F,$3B,$5A
                DFB         $78,$97,$B5,$D3
```

```
               DFB          $F2,$14,$33,$51
               SKP          1
*
********** See Rev Note #51 **********
*
yradj          dfb          90,89,88,88               ; New year table.
               dfb          87,86,91                  ; Good thru 1991.
*YRADJ DFB $54,$54,$53,$52
* DFB $57,$56,$55
****************************************

; ###############################################################################################
; #   END OF FILE:  L.TCLOCK
; #   LINES      :  122
; #   CHARACTERS :  6140
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT L.XRW.pretty
================================================================================

; ##############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  L.XRW
; ##############################################################################################

                LST          ON,ASYM,VSYM,GEN
                X65816
                XREFSLOT     6
                SEG          $00
************************************************
*
*          PRODOS 8 DISK II DRIVER (RWTS)
*
*         COPYRIGHT APPLE COMPUTER INC., 1980-1986
*
*             ALL RIGHTS RESERVED
*
*          REVISED 11/8/82 BY J.R.H.
*
************************************************
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
                ifeq         os-ProDOS
                org          $d000
                fin
                ifeq         os-EdNet
                org          $d400
                fin
                INCLUDE      MLI.SRC/XRW1
                INCLUDE      MLI.SRC/XRW2

; ##############################################################################################
; #   END OF FILE:  L.XRW
; #   LINES      :  27
; #   CHARACTERS :  757
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################################
```

```
================================================================================
DOCUMENT MAS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  MAS
; ###############################################################################################

* PRODOS KERNEL 1.0 UNIVERSAL BOOT AND
* VIRGIN DIRECTORY
* 04-APR-83
               INCLUDE      MLI.SRC/BOOT.SRC
               INCLUDE      MLI.SRC/PRODIR

; ###############################################################################################
; #   END OF FILE:  MAS
; #   LINES      :  5
; #   CHARACTERS :  155
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MASTER.BUILD.hex
================================================================================

File ............. "MASTER.BUILD"
Fork ............. DATA
Size (bytes) ..... 662 (0KB) / $00000296
Created .......... Wednesday, April 12, 2006 -- 9:20:28 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:28 AM

D/000000: 1C080A00 4424D0E7 2834293A 4224D044 [....D$..(4):B$.D]
D/000010: 24C82242 4C4F4144 20220030 081400BA [$."BLOAD.".0....]
D/000020: 44243B22 50524546 49582C44 3122005A [D$;"PREFIX,D1".Z]
D/000030: 081E0097 3ABA2220 20202020 20202020 [....:."........]
D/000040: 20202050 524F444F 53203031 2D4A554E [...PRODOS.01-JUN]
D/000050: 2D383322 3AA23130 007C0828 00BA222D [-83":.10.|.(.."-]
D/000060: 3E425549 4C44494E 47202F50 524F444F [>BUILDING./PRODO]
D/000070: 53204D41 53544552 223B0099 083200BA [S.MASTER";...2..]
D/000080: 44243B22 424C4F41 44204D41 532E312C [D$;"BLOAD.MAS.1,]
D/000090: 41243230 30302200 B1083C00 BA44243B [A$2000"...<..D$;]
D/0000A0: 22425255 4E204D41 532E494E 49542200 ["BRUN.MAS.INIT".]
D/0000B0: C3086400 903A913A 893AA231 303A9632 [..d..:.:.:.10:.2]
D/0000C0: 3600CD08 6900BA22 2E223B00 E4086E00 [6...i.."."";...n.]
D/0000D0: BA42243B 224C4452 2E312C41 24323030 [.B$;"LDR.1,A$200]
D/0000E0: 302200EE 087300BA 222E223B 00050978 [0"...s.."."";...x]
D/0000F0: 00BA4224 3B225241 4D2E322C 41243238 [..B$;"RAM.2,A$28]
D/000100: 30302200 0F097D00 BA222E22 3B002609 [00"...}.."."";.&.]
D/000110: 8200BA42 243B2252 414D2E31 2C412432 [...B$;"RAM.1,A$2]
D/000120: 39303022 00300987 00BA222E 223B0047 [900".0...."".;.G]
D/000130: 098C00BA 42243B22 52414D2E 332C4124 [....B$;"RAM.3,A$]
D/000140: 32423030 22005109 9100BA22 2E223B00 [2B00".Q...."".;.]
D/000150: 68099600 BA42243B 224D4C49 2E332C41 [h....B$;"MLI.3,A]
D/000160: 24324330 30220072 099B00BA 222E223B [$2C00".r...."".;]
D/000170: 008909A0 00BA4224 3B224D4C 492E312C [......B$;"MLI.1,]
D/000180: 41243444 30302200 9309A500 BA222E22 [A$4D00"......"."]
D/000190: 3B00AA09 AA00BA42 243B224D 4C492E32 [;......B$;"MLI.2]
D/0001A0: 2C412434 45303022 00B409AF 00BA222E [,A$4E00"......".]
D/0001B0: 223B00CE 09B400BA 42243B22 54434C4F [";......B$;"TCLO]
D/0001C0: 434B2E31 2C412434 46303022 00D809B9 [CK.1,A$4F00"....]
D/0001D0: 00BA222E 223B00EF 09BE00BA 42243B22 [.."."";......B$;"]
D/0001E0: 524F4D2E 312C4124 34463942 2200F909 [ROM.1,A$4F9B"...]
D/0001F0: C300BA22 2E223B00 100AC800 BA42243B [..."."";......B$;]
D/000200: 22585257 2E312C41 24353130 3022001A ["XRW.1,A$5100"..]
D/000210: 0ACD00BA 2221223B 003F0AD2 00BA4424 [...."!";.?....D$]
D/000220: 3B224352 45415445 202F5052 4F444F53 [;"CREATE./PRODOS]
D/000230: 2F50524F 444F532C 54244646 2200490A [/PRODOS,T$FF".I.]
D/000240: D700BA22 21223B00 7B0ADC00 BA44243B [..."!";.{....D$;]
D/000250: 22425341 5645202F 50524F44 4F532F50 ["BSAVE./PRODOS/P]
D/000260: 524F444F 532C5424 46462C41 24323030 [RODOS,T$FF,A$200]
D/000270: 302C4C24 33383030 22008E0A 0401BA3A [0,L$3800"......:]
D/000280: BA22414C 4C20444F 4E452122 00940A0E [."ALL.DONE!"....]
D/000290: 01800000 0011                      [......          ]

File ............. "MASTER.BUILD"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

FINIS
```

```
================================================================================
DOCUMENT MASTER.INIT.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  MASTER.INIT
; ###############################################################################################

                ORG         $4000
MLI             EQU         $BF00
DEVNUM          EQU         $BF30
CIERR           EQU         $BE09
                SKP         1
ENTRY           JSR         MLI
                DFB         $C4                     ;GET FILE INFO
                DW          GFI
                BCS         ERROR
                LDA         DEVNUM
                STA         UNIT
                LDY         #$20
WRITE           STY         ADR+1
                JSR         MLI                     ;WRITE BLOCKS 0-6
                DFB         $81
                DW          WRT
                BCS         ERROR
                INC         BLOK
                INY
                INY
                CPY         #$2E
                BCC         WRITE
                CLC
                RTS
                SKP         1
ERROR           LDA         #6                      ;I/O ERROR
                JMP         CIERR
                SKP         1
GFI             DFB         $A
                DW          PROPATH
                DS          15,0
PROPATH         DFB         $7
                ASC         '/PRODOS'
                SKP         1
WRT             DFB         $3
UNIT            DFB         $00
ADR             DW          $0000
BLOK            DW          $0000
                DS          $40FF-*,0

; ###############################################################################################
; #   END OF FILE:  MASTER.INIT
; #   LINES      :  39
; #   CHARACTERS :  1341
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  MLI
; ###############################################################################################

                LST          OFF,NOGEN,NOASYM,NOVSYM
                X65816
                XREFSLOT     1
                SEG          $00
*************************************************************
*                                                           *
*        PRODOS 8 - PROFESSIONAL DISK OPERATING SYSTEM      *
*                                                           *
*            COPYRIGHT APPLE COMPUTER, INC., 1983-87        *
*                                                           *
*                     ALL RIGHTS RESERVED                   *
*                                                           *
*                      February 3, 1987                     *
*                                                           *
*************************************************************
*
* Note that the "os" variable must be set to the system you
* wish to build.
*
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
                INCLUDE      MLI.SRC/REVISIONS
                ORG          $2000
                INCLUDE      MLI.SRC/EQUATES
                INCLUDE      MLI.SRC/PROLDR
                page
                INCLUDE      MLI.SRC/DEVSRCH
                INCLUDE      MLI.SRC/RELOC
                MSB          OFF
                INCLUDE      MLI.SRC/GLOBALS
                INCLUDE      MLI.SRC/XDOSMLI
                INCLUDE      MLI.SRC/BFMGR
                INCLUDE      MLI.SRC/CREATE
                INCLUDE      MLI.SRC/FNDFIL
                INCLUDE      MLI.SRC/NEWFNDVOL
                INCLUDE      MLI.SRC/ALLOC
                INCLUDE      MLI.SRC/POSN.OPEN
                INCLUDE      MLI.SRC/READ.WRITE
                INCLUDE      MLI.SRC/CLOSE.EOF
                INCLUDE      MLI.SRC/DESTROY
                INCLUDE      MLI.SRC/DETREE
                INCLUDE      MLI.SRC/MEMMGR
                INCLUDE      MLI.SRC/DATATBLS
                INCLUDE      MLI.SRC/WRKSPACE
                ifeq         os-EdNet
                include      mli.src/doatalk
                fin
                INCLUDE      MLI.SRC/ROM

; ###############################################################################################
; #   END OF FILE:  MLI
```

```
; #   LINES      :  50
; #   CHARACTERS :  2077
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.ALLOC.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  ALLOC
; ###############################################################################################

               page
*
dealloc        stx         bmcnt                    ;save high order address of block to be
freed.
               pha                                  ;save it
               ldx         vcbptr                   ; while the bitmap
               lda         vcb+vcbtblk+1,x          ; disk address is checked
               cmp         bmcnt                    ; to see if it makes sense
               pla                                  ; restore
               bcc         dealerr1                 ; branch if impossible
               tax
               and         #$7                      ;get the bit to be or-ed in.
               tay
               lda         whichbit,y               ;(shifting takes 7 bytes, but is
slower)
               sta         nofree                   ;save bit pattern
               txa                                  ;get low block address again.
               lsr         bmcnt
               ror         a                        ;get pointer to byte in bitmap that
represents
               lsr         bmcnt                    ; the block address.
               ror         a
               lsr         bmcnt
               ror         a
               sta         bmptr                    ;save pointer.
               lsr         bmcnt                    ;now transfer bit which specifies which
page of bitmap.
               rol         half
               jsr         fndbmap                  ;make absolutely sure we've got the
right device.
               bcs         dealerr                  ;return any errors.
               lda         bmacmap                  ; what is the current map
               cmp         bmcnt                    ;is in-core bit map the one we want?
               beq         deall1                   ;branch if in-core is correct.
               jsr         upbmap                   ;put current map away.
               bcs         dealerr                  ;pass back any error.
               lda         bmcnt                    ;get desired map number.
               ldx         vcbptr
               sta         vcb+vcbcmap,x            ;and make it current.
               lda         bmadev
               jsr         gtbmap                   ;read it into the buffer,
               bcs         dealerr
deall1         ldy         bmptr                    ;index to byte.
               lsr         half
               lda         nofree                   ;(get individual bit)
               bcc         deall2                   ;branch if on page one of bitmap.
               ora         bmbuf+$100,y
               sta         bmbuf+$100,y
               bcs         deall3                   ;branch always taken
deall2         equ         *
bmbufhi        ora         bmbuf,y
               sta         bmbuf,y
deall3         lda         #$80                     ;mark bitmap as modified.
```

```
                ora         bmastat
                sta         bmastat
                inc         deblock                 ;bump count of blocks deallocated
                bne         deall4
                inc         deblock+1
deall4          clc
dealerr         rts
dealerr1        lda         #bitmapadr              ; bit map block number impossible
                sec                                 ; say bit map disk address wrong
                rts                                 ;   (probably data masquerading as
index block)
                page
*
alc1blk         jsr         fndbmap                 ;get address of bit map in 'bmadr'
                bcs         erralc1                 ;branch if error encountered
srchfre         ldy         #0                      ;start search at beginning of bit map
block
                sty         half                    ;indicate which half (page) we're
searching.
getbits1        lda         bmbuf,y
                bne         bitfound                ;free blocks are indicated by 'on' bits
                iny
                bne         getbits1                ;check all of 'em in first page.
*
                inc         half                    ;indicate search has progressed to page
2
                inc         basval                  ;base value= base address/2048
getbits2        lda         bmbuf+$100,y            ;search second half for free block
                bne         bitfound
                iny
                bne         getbits2
*
                inc         basval                  ; add 2048 offset for next page
                jsr         nxtbmap                 ;get next bitmap (if it exists) and
update vcb.
                bcc         srchfre                 ;branch if no error encountered.
erralc1         rts                                 ;return error.
                page
*
bitfound        sty         bmptr                   ;save indx pointer to valid bit group
                lda         basval                  ;set up for block address calculation
                sta         scrtch+1
                tya                                 ;get address of bit pattern
                asl         a                       ;multiply this and basval by 8
                rol         scrtch+1
                asl         a
                rol         scrtch+1
                asl         a
                rol         scrtch+1
                tax                                 ;now x= low address within 7 of actual
address.
                sec
                lda         half
                beq         page1alc                ;branch if allocating from 1st half
                lda         bmbuf+$100,y            ;get pattern from second page
                bcs         adcalc                  ;branch always
page1alc        lda         bmbuf,y                 ;get bit pattern from first page
adcalc          rol         a                       ;find left most 'on' bit
                bcs         bounce                  ;branch if found.
                inx                                 ;adjust low address
                bne         adcalc                  ;branch always
bounce          lsr         a                       ;restore all but left most bit to
original position
```

```
                bcc         bounce                  ;loop until mark (set above) moves into
carry
                stx         scrtch                  ;save low address.
                ldx         half                    ;which half of bit map?
                bne         page2alc
                sta         bmbuf,y
                beq         drtybmap                ;branch always
*
page2alc        sta         bmbuf+$100,y            ;update bitmap to show allocated block
in use.
drtybmap        lda         #$80                    ;indicate map has been modified
                ora         bmastat
                sta         bmastat                 ; by setting dirty bit.
                ldy         vcbptr                  ;subtract 1 from total free
                lda         vcb+vcbtfre,y           ; blocks in vcb to account for newly
                sbc         #1                      ; allocated block (carry is set from
'bounce')
                sta         vcb+vcbtfre,y
                bcs         ret1blk                 ; branch if hi free count doesn't need
adjustment.
*
                lda         vcb+vcbtfre+1,y         ;adjust high count.
                sbc         #0                      ;(carry is clear, so acc=acc-1)
                sta         vcb+vcbtfre+1,y
ret1blk         clc                                 ;indicate no error encountered
                lda         scrtch                  ;get address low in acc.
                ldy         scrtch+1                ;and high address in y
                rts                                 ;return address of newly allocated
block.
*
                page
nxtbmap         ldy         vcbptr                  ;before bumping to next map,
                lda         vcb+vcbtblk+1,y         ;check to be sure there is
                lsr         a                       ; indeed a next map!
                lsr         a
                lsr         a
                lsr         a
                cmp         vcb+vcbcmap,y           ;are there more maps?
                beq         nomorbit                ;branch if no more to look at.
                lda         vcb+vcbcmap,y           ;add 1 to current map
                clc
                adc         #1
                sta         vcb+vcbcmap,y
                jsr         upbmap
*
fndbmap         ldy         vcbptr                  ;get device number
                lda         vcb+vcbdev,y
                cmp         bmadev
                beq         freshmap
                jsr         upbmap                  ;save out other volumes bitmap, and
                bcs         nogo
                ldy         vcbptr
                lda         vcb+vcbdev,y
                sta         bmadev                  ; read in fresh bitmap for this device.
freshmap        equ         *
                ldy         bmastat                 ;is this one already modified?
                bmi         bmfound                 ;yes, return pointer in 'bmadr'
                jsr         gtbmap                  ;otherwise read in fresh bit map
                bcs         nogo                    ;branch if unsuccessful.
*
bmfound         equ         *
                ldy         vcbptr
                lda         vcb+vcbcmap,y
```

```
                asl         a
                sta         basval
                clc                                 ;indicate all is valid and good!
nogo            rts
*
nomorbit        lda         #ovrerr                 ;indicate request can't be filled.
                sec                                 ;indicate error
                rts
*
upbmap          clc                                 ;anticipate nothing to do
                lda         bmastat                 ;is current map dirty?
                bpl         nogo                    ;no need to do anything
                jsr         wrtbmap                 ;it is dirty, update device!
                bcs         nogo                    ; error encountered on writing
                lda         #0
                sta         bmastat                 ;mark bm buffer as free
                rts                                 ;all done!
*
*
gtbmap          sta         bmadev                  ;read bitmap specified by dev & vcb
                ldy         vcbptr                  ;get lowest map number with free blocks
in it.
                lda         vcb+vcbcmap,y
                sta         bmacmap                 ; associate the offset with the bitmap
control block
                clc                                 ;add this number to the base
                adc         vcb+vcbdmap,y           ; address of first bit map
                sta         bmadadr                 ;save low address of bit map to be
used.
                lda         vcb+vcbdmap+1,y         ;now get high disk address of map
                adc         #0                      ;add to this the state of the carry
                sta         bmadadr+1               ;save high disk address too.
*
                lda         #rdcmd                  ;(x contains an index to determine
which buffer)
dobmap          sta         dhpcmd                  ;save device command
                lda         devnum                  ; preserve current devnum.
                pha
                lda         bmadev                  ;get bitmap's device number.
                sta         devnum
                lda         bmadadr                 ;and map's disk address
                sta         bloknml
                lda         bmadadr+1
                sta         bloknmh
                lda         bmbufhi+2               ;lastly get the address of the buffer
                jsr         dobitmap                ;(note: low address is fixed to zero as
this is a buffer)
                tax                                 ;preserve error code, if any.
                pla                                 ; restore
                sta         devnum                  ; the devnum we came in with!
                bcc         dobmap1                 ;return devnum if no error.
                txa                                 ;return any errors.
dobmap1         rts
*
rdblk           equ         *
                sta         bloknml
                stx         bloknmh
                jsr         rdgbuf
                rts
*
wrtbmap         lda         #wrtcmd                 ;write bit map.
                bne         dobmap                  ;branch always
*
```

```
wrtgbuf         lda          #wrtcmd                  ;set call for write.
                bne          svgcmd                   ;branch always.
rdgbuf          lda          #rdcmd                   ;set call for read.
svgcmd          sta          dhpcmd                   ;passed to device handler.
                lda          #<gbuf                   ;get high address of general buffer
dobitmap        php                                   ;no interupts allowed
                sei
                sta          dbufph
                lda          #0                       ;general purpose buffers always
                sta          dbufpl                   ; start on a page boundary.
                sta          serr                     ; clear global error value
                lda          #$ff                     ;also,
                sta          ioaccess                 ; set to indicate reg call made to dev
handler.
                lda          devnum                   ;transfer the device number for
dispatcher to convert to unit number.
                sta          unitnum
                jsr          dmgr                     ; call the driver
                bcs          fioerr1                  ;branch if error
                plp                                   ;restore interupts.
                clc
                rts
fioerr1         plp                                   ;restore interupts
                sec
                rts
*

; ###############################################################################################
; #   END OF FILE:  ALLOC
; #   LINES      :  244
; #   CHARACTERS :  14399
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.ATPREPLY.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  ATPREPLY
; ###############################################################################################

;
;
SdRsp.ssckt     equ         4
SdRsp.dnet      equ         5
SdRsp.dnode     equ         7
SdRsp.dsckt     equ         8
SdRsp.flag      equ         9
SdRsp.bitmap    equ         10
SdRsp.tid       equ         11
SdRsp.bdsadr    equ         13
SdRsp.bdssiz    equ         15
SdRsp.timer     equ         16
;
GetRq.dsckt     equ         4
GetRq.snet      equ         5
GetRq.snode     equ         7
GetRq.ssckt     equ         8
GetRq.flag      equ         9
GetRq.bitmap    equ         10
GetRq.tid       equ         11
GetRq.bdsadr    equ         13
GetRq.nxtptr    equ         16
;
;
ATPXObit        equ         $20
ATPEOMbit       equ         $10
ATPSTSbit       equ         $08
ATPchksum       equ         $01
;
OpenATPSckt     equ         *
                ldx         #>atprplisten
                ldy         #<atprplisten
                jmp         getnscktno
;
;
;
AtpSndResponse  equ         *
                php
                sei                                 ; non-rentrant code does not allow
interrupt
                ldy         #SdRsp.ssckt
                lda         (cmndlist),y
                jsr         chkvalidsckt            ; is it valid ?
                bne         retasynflag             ; bad socket error
                ldy         #GetRq.bitmap
                lda         (cmndlist),y            ; get buffer to be sent
                jsr         sendNrsp                ; send the N response
                pha                                 ; save error flag
                jsr         FindRspCBIndex          ; see if RspCB exists
                bpl         Asndrsp.9               ; yes, wait for release
                pla                                 ; not XO, get back error flag
                jmp         retasynflag             ; and return it
Asndrsp.9       pla                                 ; pop error flag and ignore it
```

```
                ldy         #SdRsp.timer
                lda         #120                    ; 30 sec
                sta         (cmndlist),y            ; also reset timer
waitdone        lda         #$80                    ; first indicates it is busy
retasynflag     ldy         #1
                sta         (cmndlist),y
                ora         #0                      ; Get status.
                bpl         procdone                ; if error or request complete call io
complete routine.
                plp                                 ; re-enable interrupt, otherwise wait
for ever
                dey                                 ; check for async
                lda         (cmndlist),y
                bmi         waitdn.9                ; Branch if async done.
                iny
waitdn.1        equ         *
                lda         (cmndlist),y
                bmi         waitdn.1
                sta         status
waitdn.9        equ         *
                rts
;
;
procdone        plp                                 ; re-enable interrupts.
chkiocmp        ldy         #0
                lda         (cmndlist),y
                bpl         waitdn.9                ; no need if not asyn
                iny
                lda         (cmndlist),y
                bmi         waitdn.9                ; not yet completed
                iny
                lda         (cmndlist),y
                sta         execadr
                iny
                lda         (cmndlist),y
                sta         execadr+1
                beq         waitdn.9                ; no completition routine
                IFNE        OPERS-PRODOS
                ldx         cmndlist
                ldy         cmndlist+1
                FIN
; fall thru to doexecute
;
; since execadr is in zero page, it will be saved and restore during interrupt
; and it can be resued after the jmp indirect
doexecute       jmp         (execadr)
;
;
AGetRequest     equ         *
                jsr         killRspCB               ; not needed, only as a caution against
user mistake
                php
                sei
                ldy         #GetRq.dsckt
                lda         (cmndlist),y            ; get socket number
                jsr         chkvalidsckt            ; is it valid ?
                bne         retasynflag             ; bad socket error
                ldy         #GetRq.nxtptr+1
                lda         #0
                sta         (cmndlist),y            ; zero it to make sure it is end of
queue
                txa
                asl         a
```

```
                tax
                lda         atprptable+1,x
                beq         getreq.3
                sta         atprqptr+1
                lda         atprptable,x
                sta         atprqptr
                DO          SAFECLOSE
getreq.1        jsr         nextinqueue             ; find end of queue
                bne         getreq.1                ; bra because y <> 0
                ELSE
getreq.1        ldy         #GetRq.nxtptr+1
                lda         (atprqptr),y
                beq         getreq.2
                tax
                dey
                lda         (atprqptr),y
                sta         atprqptr
                stx         atprqptr+1
                jmp         getreq.1
                FIN
getreq.2        lda         cmndlist+1
                sta         (atprqptr),y
                lda         cmndlist
                dey                                 ; also clear Z flag
                sta         (atprqptr),y
                bne         waitdone                ; BRA
getreq.3        lda         cmndlist
                sta         atprptable,x
                lda         cmndlist+1
                sta         atprptable+1,x
                bne         waitdone                ; BRA
;
                DO          SAFECLOSE
nextinqueue     ldy         #GetRq.nxtptr+1
                lda         (atprqptr),y
                beq         nxtq.9
                tax
                dey
                lda         (atprqptr),y
                sta         atprqptr
                stx         atprqptr+1
                iny                                 ; point back to low byte, also clear Z
flag
nxtq.9          rts
                FIN
;
;
FindRspCBIndex  equ         *                       ; return index into RspCBTable
                ldx         #RspCBend               ; check for RspCB
                lda         #RspCBbegin
FindTable       sta         fRsp.2+1
fRsp.1          dex
                dex
fRsp.2          cpx         #0                      ; self modifying code
                bmi         fRsp.9
                lda         cmndlist
                cmp         processtable,x
                bne         fRsp.1
                lda         cmndlist+1
                cmp         processtable+1,x
                bne         fRsp.1
fRsp.9          rts                                 ;
;
```

```
atprplisten     equ         *                       ; listener for reply
                jsr         rdatpheader             ; read header and calculate checksum
                bne         fRsp.9                  ; error, return with carry flag clear
                lda         atpheader
                tax
                asl         a                       ; check function
                bpl         secrts                  ; request, ignore it
                bcs         atprpls.3               ; release ; release XO
                txa
                and         #ATPXObit
                beq         atprpls.4               ; not XO
                jsr         LookRspCB               ; check for duplicate
                sta         atprpls.4a+1
                bmi         atprpls.4               ; no, do it the normal way
                ldy         #1
                lda         (cmndlist),y            ; are we ready ?
                bpl         secrts                  ; no
; potential bug, we may need to check more
                lda         atpheader+1             ; any request ?
                beq         secrts                  ; no, ignore it
                ldy         #SdRsp.bitmap
                and         (cmndlist),y            ; find repsonse needed
                beq         *+5
                jmp         sendNrsp
                lda         #requestfail            ; none of the request is available
                bne         retasynsec              ; BRA to return as error
atprpls.3       jsr         LookRspCB               ; is it in table ?
                bmi         secrts                  ; no, ignore it
                tax
                lda         #0                      ; also set error to 0
                sta         processtable+1,x        ; remove item from table
retasynsec      ldy         #1                      ; return asyn result byte and also set
carry
                sta         (cmndlist),y
                jsr         chkiocmp
secrts          sec
                rts
atprpls.4       lda         ddphead+ld.dsckt        ; address to which socket
                jsr         srchscktable
;bne secrts ; discard it, actually this is not possible
                txa
                asl         a
                tax
                lda         atprptable+1,x          ; anything queue up ?
                beq         secrts                  ; no discard it
                sta         cmndlist+1
                lda         atprptable,x
                sta         cmndlist
                stx         atprpls.5+1             ; save x reg for use later
                lda         #0                      ; it is the first bds
                ldy         #GetRq.bdsadr
                jsr         readltdsize
                bne         atprpls.8               ; checksum error, do not use it
                lda         atpheader
                and         #ATPXObit               ; is it XO ?
                beq         atprpls.5               ; then no need to generate Rsp CB
atprpls.4a      lda         #0                      ; self modifying code, do we have one
already
;
; note this is OK now, but if we ever change the process table
;   then we may have a bug
;
                bne         atprpls.5               ; we already have a Rsp CB
```

```
                ldy         #RspCBSize*2
atprpls.4b      dey
                dey
                bmi         atprpls.8               ; don't do it unless we have room for
Rsp CB
                lda         RspCBTable+1,y
                bne         atprpls.4b
                lda         cmndlist                ; create a RspCB
                sta         RspCBTable,y
                lda         cmndlist+1
                sta         RspCBTable+1,y
atprpls.5       ldx         #0                      ; self modifying code
                ldy         #GetRq.nxtptr           ; remove front item from queue
                lda         (cmndlist),y
                sta         atprptable,x
                iny
                lda         (cmndlist),y
                sta         atprptable+1,x
                ldy         #GetRq.snet
atprpls.5a      ldx         RspCmptable+1-GetRq.snet,y
                beq         atprpls.5b
                lda         dataarea,x
                sta         (cmndlist),y
                iny
                bne         atprpls.5a              ; BRA
atprpls.5b      ldy         #GetRq.flag
atprpls.5c      lda         atpheader-GetRq.flag,y
                sta         (cmndlist),y
                iny
                cpy         #GetRq.tid+2
                bne         atprpls.5c
                lda         #0
                ldy         #1
                sta         (cmndlist),y
                jsr         chkiocmp
atprpls.8       clc                                 ; indicate we have processed it
                rts
;
LookRspCB       equ         *
                lda         #RspCBend
                ldx         #RspCBbegin
                ldy         #ddphead+ld.snet-dataarea ; must check network number
                sty         RspCmptable+1
                iny
                sty         RspCmptable+2
Looktable       pha                                 ; save index
                stx         lkrsp.2+1
lkrsp.1         pla
                tax
                dex
                dex
                txa
lkrsp.2         cpx         #0
                bmi         lkrsp.5                 ; return with N set
                pha
                lda         processtable,x
                sta         CmndList
                lda         processtable+1,x
                sta         CmndList+1
                beq         lkrsp.1                 ; empty, try next
                ldy         #SdRsp.ssckt-1          ; coming from right address ?
lkrsp.3         iny
                ldx         RspCmptable-SdRsp.ssckt,y
```

```
                beq         lkrsp.3                 ; don't care
                bmi         lkrsp.4                 ; end of list marker, this is it
                lda         (CmndList),y
                cmp         dataarea,x
                beq         lkrsp.3                 ; agree, then check more
                bne         lkrsp.1                 ; no, try again
lkrsp.4         pla                                 ; since acc is always positive, return
with N clear
lkrsp.5         rts
;
RspCmptable     equ         *
                dfb         ddphead+ld.dsckt-dataarea
                dfb         ddphead+ld.snet-dataarea
                dfb         ddphead+ld.snet+1-dataarea
                dfb         ddphead+ld.snode-dataarea
                dfb         ddphead+ld.ssckt-dataarea
                dfb         0,0
                dfb         atpheader+2-dataarea
                dfb         atpheader+3-dataarea
                dfb         -1
;
;
;
;
sendNrsp        equ         *
                pha                                 ; save the bit map to be sent
                lda         #$80                    ; response packet
                sta         atpheader
                lda         #$FF
                sta         atpheader+1             ; start with response -1
sdrsp.1         pla                                 ; get back bit map to be sent
                beq         sdrsp.9                 ; done
                inc         atpheader+1
                lsr         a                       ; move bit to carry
                pha
                bcc         sdrsp.1                 ; this buffer not needed
                bne         sdrsp.2                 ; not last buffer in sequence
                ldy         #SdRsp.flag
                lda         (cmndlist),y
                and         #ATPSTSbit
                ora         atpheader
                sta         atpheader
sdrsp.2         lda         atpheader+1
                tax
                adc         #0                      ; since carry is set (else bcc
sdrsp.1), we increment acc by 1
                ldy         #SdRsp.bdssiz
                cmp         (cmndlist),y            ; is this EOM
                bcc         sdrsp.3                 ; not last, just send it
                bne         sdrsp.1                 ; not smaller or equal, then too big
                ldy         #SdRsp.flag             ; equal to last, then may need EOM
                lda         (cmndlist),y
                and         #ATPEOMbit
                ora         atpheader
                sta         atpheader
sdrsp.3         txa
                asl         a                       ; * 2
                asl         a                       ; * 4
                adc         atpheader+1             ; *5
                asl         a                       ; * 10
                jsr         watppckt                ; send an ATP packet
                beq         sdrsp.1                 ; no error
                tax                                 ; save error
```

```
                pla                                     ; pop stack
                txa                                     ; get back error
sdrsp.9         sec                                     ; always return with carry set for
cortland compatabliity
                rts
;
;
;
killRspCB       php
                sei
                jsr         FindRspCBIndex              ; see if RspCB exists
                bmi         *+7
                lda         #0
                sta         processtable+1,x            ; remove item from table
                plp
                rts

; ##############################################################################################
; #   END OF FILE:  ATPREPLY
; #   LINES      :  372
; #   CHARACTERS :  16566
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.ATPREQUEST.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  ATPREQUEST
; ###############################################################################################

curatptid       dw          0
;
bdsptr          equ         cmndlist+2
;
atprq.result    equ         1
atprq.dymsckt   equ         4
atprq.dnet      equ         5
atprq.dnode     equ         7
atprq.dsckt     equ         8
atprq.flag      equ         9
atprq.bitmap    equ         10
atprq.tid       equ         11
atprq.sndbds    equ         13
atprq.rbdsad    equ         15
atprq.rbnum     equ         17
atprq.response  equ         18
atprq.timeout   equ         19
atprq.rtrycnt   equ         20
atprq.timleft   equ         21
atprq.tryleft   equ         22
;
                do          savespace
;
decindy         sec
                lda         (cmndlist),y
                sbc         #1
                sta         (cmndlist),y
                rts
;
incindy         clc
                lda         (cmndlist),y
                adc         #1
                sta         (cmndlist),y
                rts
                fin
;
;
;
sndatprq        equ         *
                php
                sei
                ldx         #atprqtblsiz+atprqtblsiz-2 ; end of table
sndatpq.1       lda         atprqtable+1,x             ; search for available slot
                beq         sndatpq.5                  ; found
                dex
                dex
                bpl         sndatpq.1                  ; continue if more
toomanyproc     lda         #toomanyreq                ; otherwise we cannot do it yet
                jmp         retasynflag
sndatpq.5       stx         sndatpq.6+1                ; save x
                ldy         #atprq.response
                lda         #0
                sta         (cmndlist),y               ; no response yet
```

```
                ldx         #>atplisten
                ldy         #<atplisten
                jsr         opensckt                ; try to open a dynamic socket
                beq         toomanyproc             ; open failed
                ldy         #atprq.dymsckt
                sta         (cmndlist),y
sndatpq.6       ldx         #0                      ; get back x
                lda         cmndlist
                sta         atprqtable,x
                lda         cmndlist+1
                sta         atprqtable+1,x
                ldy         #atprq.tid+1
                inc         curatptid+1
                lda         curatptid+1             ; generate new transaction id
                sta         (cmndlist),y
                bne         *+5
                inc         curatptid
                lda         curatptid
                dey
                sta         (cmndlist),y
                ldy         #atprq.rtrycnt
                lda         (cmndlist),y            ; retry count
                ldy         #atprq.tryleft
                sta         (cmndlist),y
                jsr         sndatrqpckt
                jmp         waitdone
;
;
atptmpindex     equ         cmndlist+2
;
atplisten       jsr         rdatpheader             ; read the 8 byte atp header, and
calculate checksum
                bne         atpl.rts                ; if error, do nothing, carry flag
already set
                lda         atpheader               ; check function code
                and         #$c0
                cmp         #$80                    ; is it a reply ?
                bne         noatp
                ldx         #atprqbegin
                lda         #atprqend
                ldy         #0                      ; ignore network number so that 0 and
local net is equivalent
                sty         rspcmptable+1
                sty         rspcmptable+2
                jsr         looktable               ; search table for occupied slot
                bmi         noatp                   ; not found
                lda         atpheader+1
                ldy         #atprq.rbnum
                cmp         (cmndlist),y            ; compare with nubmer of buffers
                bcc         atls.0                  ; ok
noatp           sec                                 ; discard data
atpl.rts        rts
atls.0          tax
                lda         onebitmask,x
                ldy         #atprq.bitmap
                and         (cmndlist),y            ; do we want this one ?
                beq         noatp                   ; no, throw it away
                txa
                asl         a                       ; * 2
                asl         a                       ; * 4, also clear carry
                adc         atpheader+1             ; result is * 5
                asl         a                       ; * 10
                ldy         #atprq.rbdsad
```

```
                jsr         readltdsize             ; read it, but no more can buffer can
accomdate
                bne         atpclc
                ldy         #atprq.response
                do          savespace
                jsr         incindy
                else
                clc
                lda         #1
                adc         (cmndlist),y
                sta         (cmndlist),y            ; increment receive count
                fin
                ldy         #atprq.bitmap
                ldx         atpheader+1
                lda         atpheader
                and         #atpeombit              ; is it eom ?
                beq         atls.8                  ; no
                lda         lowbitmask,x
                and         (cmndlist),y
                jmp         atls.9
atls.8          lda         onebitmask,x
                eor         (cmndlist),y
atls.9          sta         (cmndlist),y
                php
                lda         atpheader
                and         #atpstsbit              ; is it sts
                beq         atls.10                 ; no
                jsr         sndatrqpckt             ; send new bit map, do not consume
retry
atls.10         plp
                bne         atpclc
cancelatpreq    lda         #0
killrequest     php
                sei
                ldy         #1
                sta         (cmndlist),y
                ldx         #atprqend
                lda         #atprqbegin
                jsr         findtable               ; see if it is in table
                bmi         kilreq.9                ; not in table, ignore it
                lda         #0
                sta         processtable+1,x
                ldy         #atprq.dymsckt
                lda         (cmndlist),y
                jsr         closeddpsckt            ; remove the dynamic socket
                ldy         #atprq.flag
                lda         (cmndlist),y
                and         #atpxobit               ; is it xo
                beq         kilreq.9                ; no
                lda         #$c0                    ; else release it
                jsr         sndatppckt              ; send release packet
kilreq.9        plp
                jsr         chkiocmp                ; do i/o completetion routine
atpclc          clc
                rts
ratprest        dfb         5
                dw          0,0
onebitmask      dfb         $01,$02,$04,$08,$10,$20,$40,$80
lowbitmask      dfb         $00,$01,$03,$07,$0f,$1f,$3f,$7f
;
;
readltdsize     equ         *                       ; read with size restriction
                do          savespace
```

```
                jsr         calcbdsadr
                else
                clc
                adc         (cmndlist),y            ; get bds
                sta         bdsptr
                iny
                lda         (cmndlist),y
                adc         #0
                sta         bdsptr+1
                fin
                ldy         #6                      ; copy four user bytes
rdltd.1         lda         atpheader+4-6,y
                sta         (bdsptr),y
                iny
                cpy         #10
                bne         rdltd.1
                ldy         #3                      ; get address of buffer
rdltd.2         lda         (bdsptr),y
                sta         ratprest-1,y
                dey
                cpy         #1
                bne         rdltd.2
rdltd.2a        lda         (bdsptr),y
                sta         ratprest+3,y
                dey
                bpl         rdltd.2a
                clc                                 ; find the length of the rest of packet
                lda         #-13
                ldx         laptype
                dex                                 ; is it short ddp
                beq         *+4
                lda         #-21                    ; long ddp + atp header bytes
                adc         ddphead+ld.length+1
                tax                                 ; save low byte in x
                ldy         #4
                sta         (bdsptr),y
                lda         ddphead+ld.length
                and         #3
                adc         #$ff
                iny                                 ; y = 5
                sta         (bdsptr),y
                ldy         #3
                cmp         (bdsptr),y
                bcc         rdltd.5                 ; small packet
                bne         rdltd.7                 ; big packet
                dey                                 ; same size, must compare low byte  y =
2
                txa                                 ; get back low byte of packet size
                cmp         (bdsptr),y
                bcc         rdltd.6                 ; small packet
                bcs         rdltd.7                 ; exact size or too big
; packet is smaller than the buffer
rdltd.5         sta         ratprest+4              ; high byte of length
rdltd.6         stx         ratprest+3              ; low byte of length
rdltd.7         ldy         #<ratprest
                ldx         #>ratprest
                jsr         readhdrsum              ; read that many byte into buffer
                bne         rdltd.8                 ; should never happen, can be throw
away if space is tight
                jsr         discard
                lda         chksmf
                beq         rdltd.9
                lda         curcksum                ; check if checksum is correct
```

```
                cmp         ddphead+ld.chksum
                bne         rdltd.8
                lda         curcksum+1
                cmp         ddphead+ld.chksum+1
                beq         rdltd.9
rdltd.8         lda         #0
                ldy         #4
                sta         (bdsptr),y
                iny
                sta         (bdsptr),y              ; note that z was clear
rdltd.9         rts
;
;
;
atptimeout      equ         *
                lda         #atprqend
                pha
atpqtim.1       pla                                 ; find a slot that has atp operation
                tax
                dex
                dex
                txa
                cmp         #rspcbbegin
                bmi         rdltd.9
                pha
                lda         processtable+1,x
                beq         atpqtim.1
                sta         cmndlist+1
                lda         processtable,x
                sta         cmndlist
                cpx         #atprqbegin             ; waiting for reply ?
                bcc         atpqtim.5               ; no, then rspcb
                do          savespace
                ldy         #atprq.timleft
                jsr         decindy
                else
                ldy         #atprq.timleft
                lda         (cmndlist),y
                sec
                sbc         #1
                sta         (cmndlist),y
                fin
                bne         atpqtim.1
                ldy         #atprq.tryleft
                lda         (cmndlist),y
                beq         atpqtim.4
                cmp         #$ff                    ; infinite retry
                beq         atpqtim.3               ; then do not decrement count
                do          savespace
                jsr         decindy
                else
                lda         (cmndlist),y
                sec
                sbc         #1
                sta         (cmndlist),y
                fin
atpqtim.3       jsr         sndatrqpckt             ; send a atp request packet
                jmp         atpqtim.1
atpqtim.4       lda         #requestfail
                jsr         killrequest
                jmp         atpqtim.1
atpqtim.5       equ         *                       ; check to find rspcb to release
                ldy         #sdrsp.timer
```

```
                do          savespace
                jsr         decindy                 ; decrement timer
                else
                sec
                lda         (cmndlist),y
                sbc         #1
                sta         (cmndlist),y
                fin
                bne         atpqtim.1               ; timer still going
                lda         #0
                sta         processtable+1,x        ; else remove the item
                ldy         #1
                sta         (cmndlist),y            ; release the response command
                jsr         chkiocmp                ; do i/o completetion routine
                jmp         atpqtim.1
;
; send an atp request packet
;
sndatrqpckt     equ         *
                ldy         #atprq.timeout
                lda         (cmndlist),y
                ldy         #atprq.timleft
                sta         (cmndlist),y
                ldy         #atprq.flag
                lda         (cmndlist),y            ; get flags
                and         #atpxobit
                ora         #$40
; jsr sndatppckt ; fall through to send generic atp packet
; rts
;
; send an atp packet
;
sndatppckt      sta         atpheader
                ldy         #atprq.bitmap
                lda         (cmndlist),y
                sta         atpheader+1
                lda         #0
                jsr         watppckt                ; write it out
                beq         sndatp.rts              ; no error
                tax
                ldy         #atprq.tryleft
                lda         (cmndlist),y            ; any more try left
                bne         sndatp.rts              ; yes, then ignore error
                txa
                jmp         killrequest
sndatp.rts      rts
;
;
;
watppckt        equ         *
                pha
                lda         #3
                sta         ddphead+ld.protocol     ; atp is protocol 3
                ldy         #atprq.dymsckt
                lda         (cmndlist),y
                jsr         setdestadr              ; use parmeter list to set dest addr
                pla                                 ; get back bds displacement
                ldy         #atprq.sndbds
                do          savespace
                jsr         calcbdsadr
                else
                clc
                adc         (cmndlist),y
```

```
                sta         bdsptr
                iny
                lda         #0
                adc         (cmndlist),y
                sta         bdsptr+1
                fin
                ldy         #3
sdrsp.4         lda         (bdsptr),y
                sta         atprqwds+8,y
                dey
                bpl         sdrsp.4
                ldy         #6
sdrsp.5         lda         (bdsptr),y
                sta         atpheader+4-6,y
                iny
                cpy         #10
                bcc         sdrsp.5
                ldy         #atprq.tid
                lda         (cmndlist),y
                sta         atpheader+2
                iny
                lda         (cmndlist),y
                sta         atpheader+3
                ldx         #>atprqwds
                ldy         #<atprqwds
                jmp         dowrtddp                ; if we want to save 3 bytes, we can
fall thru to dowrtddp
;
                do          savespace
calcbdsadr      clc
                adc         (cmndlist),y
                sta         bdsptr
                iny
                lda         #0
                adc         (cmndlist),y
                sta         bdsptr+1
                rts
                fin
;
atprqwds        ds          4,0
                dw          8,atpheader
                ds          4,0
                dw          $ffff
;
ratpparm        dfb         5
                dw          atpheader
                dw          8

; ###########################################################################################
; #   END OF FILE:  ATPREQUEST
; #   LINES      :  407
; #   CHARACTERS :  16583
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.ATREPLY.pretty
================================================================================

; ##############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  ATREPLY
; ##############################################################################################

;
;
sdrsp.ssckt     equ         4
sdrsp.dnet      equ         5
sdrsp.dnode     equ         7
sdrsp.dsckt     equ         8
sdrsp.flag      equ         9
sdrsp.bitmap    equ         10
sdrsp.tid       equ         11
sdrsp.bdsadr    equ         13
sdrsp.bdssiz    equ         15
sdrsp.timer     equ         16
;
getrq.dsckt     equ         4
getrq.snet      equ         5
getrq.snode     equ         7
getrq.ssckt     equ         8
getrq.flag      equ         9
getrq.bitmap    equ         10
getrq.tid       equ         11
getrq.bdsadr    equ         13
getrq.nxtptr    equ         16
;
;
atpxobit        equ         $20
atpeombit       equ         $10
atpstsbit       equ         $08
atpchksum       equ         $01
;
openatpsckt     equ         *
                ldx         #>atprplisten
                ldy         #<atprplisten
                jmp         getnscktno
;
;
;
atpsndresponse  equ         *
                php
                sei                                 ; non-rentrant code does not allow
interrupt
                ldy         #sdrsp.ssckt
                lda         (cmndlist),y
                jsr         chkvalidsckt            ; is it valid ?
                bne         retasynflag             ; bad socket error
                ldy         #getrq.bitmap
                lda         (cmndlist),y            ; get buffer to be sent
                jsr         sendnrsp                ; send the n response
                pha                                 ; save error flag
                jsr         findrspcbindex          ; see if rspcb exists
                bpl         asndrsp.9               ; yes, wait for release
                pla                                 ; not xo, get back error flag
                jmp         retasynflag             ; and return it
asndrsp.9       pla                                 ; pop error flag and ignore it
```

```
                ldy         #sdrsp.timer
                lda         #120                    ; 30 sec
                sta         (cmndlist),y            ; also reset timer
waitdone        lda         #$80                    ; first indicates it is busy
retasynflag     ldy         #1
                sta         (cmndlist),y
                ora         #0                      ; get status.
                bpl         procdone                ; if error or request complete call io
complete routine.
                plp                                 ; re-enable interrupt, otherwise wait
for ever
                dey                                 ; check for async
                lda         (cmndlist),y
                bmi         waitdn.9                ; branch if async done.
                iny
waitdn.1        equ         *
                lda         (cmndlist),y
                bmi         waitdn.1
                sta         status
waitdn.9        equ         *
                rts
;
;
procdone        plp                                 ; re-enable interrupts.
chkiocmp        ldy         #0
                lda         (cmndlist),y
                bpl         waitdn.9                ; no need if not asyn
                iny
                lda         (cmndlist),y
                bmi         waitdn.9                ; not yet completed
                iny
                lda         (cmndlist),y
                sta         execadr
                iny
                lda         (cmndlist),y
                sta         execadr+1
                beq         waitdn.9                ; no completition routine
                ifne        opers-prodos
                ldx         cmndlist
                ldy         cmndlist+1
                fin
; fall thru to doexecute
;
; since execadr is in zero page, it will be saved and restore during interrupt
; and it can be resued after the jmp indirect
doexecute       jmp         (execadr)
;
;
agetrequest     equ         *
                jsr         killrspcb               ; not needed, only as a caution against
user mistake
                php
                sei
                ldy         #getrq.dsckt
                lda         (cmndlist),y            ; get socket number
                jsr         chkvalidsckt            ; is it valid ?
                bne         retasynflag             ; bad socket error
                ldy         #getrq.nxtptr+1
                lda         #0
                sta         (cmndlist),y            ; zero it to make sure it is end of
queue
                txa
                asl         a
```

```
                tax
                lda         atprptable+1,x
                beq         getreq.3
                sta         atprqptr+1
                lda         atprptable,x
                sta         atprqptr
                do          safeclose
getreq.1        jsr         nextinqueue             ; find end of queue
                bne         getreq.1                ; bra because y <> 0
                else
getreq.1        ldy         #getrq.nxtptr+1
                lda         (atprqptr),y
                beq         getreq.2
                tax
                dey
                lda         (atprqptr),y
                sta         atprqptr
                stx         atprqptr+1
                jmp         getreq.1
                fin
getreq.2        lda         cmndlist+1
                sta         (atprqptr),y
                lda         cmndlist
                dey                                 ; also clear z flag
                sta         (atprqptr),y
                bne         waitdone                ; bra
getreq.3        lda         cmndlist
                sta         atprptable,x
                lda         cmndlist+1
                sta         atprptable+1,x
                bne         waitdone                ; bra
;
                do          safeclose
nextinqueue     ldy         #getrq.nxtptr+1
                lda         (atprqptr),y
                beq         nxtq.9
                tax
                dey
                lda         (atprqptr),y
                sta         atprqptr
                stx         atprqptr+1
                iny                                 ; point back to low byte, also clear z
flag
nxtq.9          rts
                fin
;
;
findrspcbindex  equ         *                       ; return index into rspcbtable
                ldx         #rspcbend               ; check for rspcb
                lda         #rspcbbegin
findtable       sta         frsp.2+1
frsp.1          dex
                dex
frsp.2          cpx         #0                      ; self modifying code
                bmi         frsp.9
                lda         cmndlist
                cmp         processtable,x
                bne         frsp.1
                lda         cmndlist+1
                cmp         processtable+1,x
                bne         frsp.1
frsp.9          rts                                 ;
;
```

```
atprplisten     equ          *                        ; listener for reply
                jsr          rdatpheader              ; read header and calculate checksum
                bne          frsp.9                   ; error, return with carry flag clear
                lda          atpheader
                tax
                asl          a                        ; check function
                bpl          secrts                   ; request, ignore it
                bcs          atprpls.3                ; release ; release xo
                txa
                and          #atpxobit
                beq          atprpls.4                ; not xo
                jsr          lookrspcb                ; check for duplicate
                sta          atprpls.4a+1
                bmi          atprpls.4                ; no, do it the normal way
                ldy          #1
                lda          (cmndlist),y             ; are we ready ?
                bpl          secrts                   ; no
; potential bug, we may need to check more
                lda          atpheader+1              ; any request ?
                beq          secrts                   ; no, ignore it
                ldy          #sdrsp.bitmap
                and          (cmndlist),y             ; find repsonse needed
                beq          *+5
                jmp          sendnrsp
                lda          #requestfail             ; none of the request is available
                bne          retasynsec               ; bra to return as error
atprpls.3       jsr          lookrspcb                ; is it in table ?
                bmi          secrts                   ; no, ignore it
                tax
                lda          #0                       ; also set error to 0
                sta          processtable+1,x         ; remove item from table
retasynsec      ldy          #1                       ; return asyn result byte and also set
carry
                sta          (cmndlist),y
                jsr          chkiocmp
secrts          sec
                rts
atprpls.4       lda          ddphead+ld.dsckt         ; address to which socket
                jsr          srchscktable
;bne secrts ; discard it, actually this is not possible
                txa
                asl          a
                tax
                lda          atprptable+1,x           ; anything queue up ?
                beq          secrts                   ; no discard it
                sta          cmndlist+1
                lda          atprptable,x
                sta          cmndlist
                stx          atprpls.5+1              ; save x reg for use later
                lda          #0                       ; it is the first bds
                ldy          #getrq.bdsadr
                jsr          readltdsize
                bne          atprpls.8                ; checksum error, do not use it
                lda          atpheader
                and          #atpxobit                ; is it xo ?
                beq          atprpls.5                ; then no need to generate rsp cb
atprpls.4a      lda          #0                       ; self modifying code, do we have one
already
;
; note this is ok now, but if we ever change the process table
;   then we may have a bug
;
                bne          atprpls.5                ; we already have a rsp cb
```

```
                ldy         #rspcbsize*2
atprpls.4b      dey
                dey
                bmi         atprpls.8               ; don't do it unless we have room for
rsp cb
                lda         rspcbtable+1,y
                bne         atprpls.4b
                lda         cmndlist                ; create a rspcb
                sta         rspcbtable,y
                lda         cmndlist+1
                sta         rspcbtable+1,y
atprpls.5       ldx         #0                      ; self modifying code
                ldy         #getrq.nxtptr           ; remove front item from queue
                lda         (cmndlist),y
                sta         atprptable,x
                iny
                lda         (cmndlist),y
                sta         atprptable+1,x
                ldy         #getrq.snet
atprpls.5a      ldx         rspcmptable+1-getrq.snet,y
                beq         atprpls.5b
                lda         dataarea,x
                sta         (cmndlist),y
                iny
                bne         atprpls.5a              ; bra
atprpls.5b      ldy         #getrq.flag
atprpls.5c      lda         atpheader-getrq.flag,y
                sta         (cmndlist),y
                iny
                cpy         #getrq.tid+2
                bne         atprpls.5c
                lda         #0
                ldy         #1
                sta         (cmndlist),y
                jsr         chkiocmp
atprpls.8       clc                                 ; indicate we have processed it
                rts
;
lookrspcb       equ         *
                lda         #rspcbend
                ldx         #rspcbbegin
                ldy         #ddphead+ld.snet-dataarea ; must check network number
                sty         rspcmptable+1
                iny
                sty         rspcmptable+2
looktable       pha                                 ; save index
                stx         lkrsp.2+1
lkrsp.1         pla
                tax
                dex
                dex
                txa
lkrsp.2         cpx         #0
                bmi         lkrsp.5                 ; return with n set
                pha
                lda         processtable,x
                sta         cmndlist
                lda         processtable+1,x
                sta         cmndlist+1
                beq         lkrsp.1                 ; empty, try next
                ldy         #sdrsp.ssckt-1          ; coming from right address ?
lkrsp.3         iny
                ldx         rspcmptable-sdrsp.ssckt,y
```

```
                beq         lkrsp.3                 ; don't care
                bmi         lkrsp.4                 ; end of list marker, this is it
                lda         (cmndlist),y
                cmp         dataarea,x
                beq         lkrsp.3                 ; agree, then check more
                bne         lkrsp.1                 ; no, try again
lkrsp.4         pla                                 ; since acc is always positive, return
with n clear
lkrsp.5         rts
;
rspcmptable     equ         *
                dfb         ddphead+ld.dsckt-dataarea
                dfb         ddphead+ld.snet-dataarea
                dfb         ddphead+ld.snet+1-dataarea
                dfb         ddphead+ld.snode-dataarea
                dfb         ddphead+ld.ssckt-dataarea
                dfb         0,0
                dfb         atpheader+2-dataarea
                dfb         atpheader+3-dataarea
                dfb         -1
;
;
;
;
sendnrsp        equ         *
                pha                                 ; save the bit map to be sent
                lda         #$80                    ; response packet
                sta         atpheader
                lda         #$ff
                sta         atpheader+1             ; start with response -1
sdrsp.1         pla                                 ; get back bit map to be sent
                beq         sdrsp.9                 ; done
                inc         atpheader+1
                lsr         a                       ; move bit to carry
                pha
                bcc         sdrsp.1                 ; this buffer not needed
                bne         sdrsp.2                 ; not last buffer in sequence
                ldy         #sdrsp.flag
                lda         (cmndlist),y
                and         #atpstsbit
                ora         atpheader
                sta         atpheader
sdrsp.2         lda         atpheader+1
                tax
                adc         #0                      ; since carry is set (else bcc
sdrsp.1), we increment acc by 1
                ldy         #sdrsp.bdssiz
                cmp         (cmndlist),y            ; is this eom
                bcc         sdrsp.3                 ; not last, just send it
                bne         sdrsp.1                 ; not smaller or equal, then too big
                ldy         #sdrsp.flag             ; equal to last, then may need eom
                lda         (cmndlist),y
                and         #atpeombit
                ora         atpheader
                sta         atpheader
sdrsp.3         txa
                asl         a                       ; * 2
                asl         a                       ; * 4
                adc         atpheader+1             ; *5
                asl         a                       ; * 10
                jsr         watppckt                ; send an atp packet
                beq         sdrsp.1                 ; no error
                tax                                 ; save error
```

```
                pla                                     ; pop stack
                txa                                     ; get back error
sdrsp.9         sec                                     ; always return with carry set for
cortland compatabliity
                rts
;
;
;
killrspcb       php
                sei
                jsr         findrspcbindex              ; see if rspcb exists
                bmi         *+7
                lda         #0
                sta         processtable+1,x            ; remove item from table
                plp
                rts

; ###############################################################################################
; #   END OF FILE:  ATREPLY
; #   LINES      :  372
; #   CHARACTERS :  16566
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.BFMGR.ONE.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  BFMGR.ONE
; ###############################################################################################

                SBTL        'ProDOS Block File Manager'
BFMGR           STX         COMMAND                 ;WHAT CALL?
                LDA         DISPTCH,X               ;TRANSLATE INTO COMMAND ADDRESS
                ASL         A                       ;(BIT 7 INDICATES A PATHNAME TO
PREPROCESS)
                STA         CMDTEMP
                AND         #$3F                    ;(BIT 6 IS REFNUM PREPROCESS, 5 IS FOR
TIME, SO STRIP EM.)
                TAX
                LDA         CMDTABLE,X              ;MOVE ADDRESS FOR INDIRECT JUMP.
                STA         GOADR
                LDA         CMDTABLE+1,X            ;(HIGH BYTE)
                STA         GOADR+1
                LDA         #$20                    ; INIT "BACKUP BIT FLAG"
                STA         BKBITFLG                ; TO SAY "FILE MODIFIED"
                BCC         NOPATH
                JSR         SETPATH                 ;GO PROCESS PATHNAME BEFORE CALLING
COMMAND
                BCS         ERRORSYS                ;BRANCH IF BAD NAME.
NOPATH          ASL         CMDTEMP                 ;TEST FOR REFNUM PREPROCESSING
                BCC         NOPREREF
                JSR         FINDFCB                 ;GO SET UP POINTERS TO FCB AND VCB OF
THIS FILE.
                BCS         ERRORSYS                ;BRANCH IF ANY ERRORS ARE ENCOUNTERED.
NOPREREF        ASL         CMDTEMP                 ;LASTLY CHECK FOR NECESSITY OF TIME
STAMP.
                BCC         EXECUTE
                JSR         DATETIME                ;(NO ERROR POSIBLE)
EXECUTE         JSR         GOCMD                   ;EXECUTE COMMAND
                BCC         GOODOP                  ;BRANCH IF SUCCESSFUL
*
ERRORSYS        JSR         SYSERR
GOODOP          RTS                                 ; GOOD RETURN
*
GOCMD           JMP         (GOADR)
                PAGE
SETPATH         EQU         *
                LDY         #C.PATH                 ;INDEX TO PATHNAME POINTER
                LDA         (PAR),Y                 ;GET LOW POINTER ADDR.
                STA         TPATH
                INY
                LDA         (PAR),Y                 ; AND HI POINTER ADDR.
                STA         TPATH+1
*
SYNPATH         EQU         *                       ;ENTRY USED BY RENAME FOR SECOND
PATHNAME.
                LDX         #0                      ;X REGISTER IS USED AS INDEX TO PATHBUF
                LDY         #0                      ;Y REGISTER IS INDEX TO INPUT PATHNAME.
                STX         PRFXFLG                 ;ASSUME PREFIX IS IN USE.
                STX         PATHBUF                 ;MARK PATHBUF TO INDICATE NOTHING
PROCESSED.
                LDA         (TPATH),Y               ;VALIDATE PATHNAME LENGTH>0, AND <65
                BEQ         ERRSYN
```

```
                CMP         #$41
                BCS         ERRSYN
                STA         PATHCNT                 ;THIS IS USED TO COMPARE FOR
                INC         PATHCNT                 ; END OF PATHNAME PROCESSING.
                INY                                 ;NOW CHECK FOR FULL PATHNAME...
                LDA         (TPATH),Y               ;(FULL NAME IF STARTS WITH "/".)
                ORA         #$80
                CMP         #$AF
                BNE         SPATH2                  ;BRANCH IF PREFIX APPENDED
                STA         PRFXFLG                 ;SET PREFIX FLAG TO INDICATE PREFIX NOT
USED.
                INY                                 ;INDEX TO FIRST CHARCTER OF PATHNAME.
*
SPATH2          EQU         *
                LDA         #$FF                    ;SET CURRENT POSITION OF PATHBUF
                STA         PATHBUF,X               ; TO INDICATE END OF PATHNAME.
                STA         NAMCNT                  ;ALSO INDICATE NO CHARACTERS PROCESSED
IN LOCAL NAME.
                STX         NAMPTR                  ;PRESERVE POINTER TO LOCAL NAME LENGTH
BYTE.
SPATH3          EQU         *
                CPY         PATHCNT                 ;DONE WITH PATHNAME PROCESSING?
                BCS         ENDPATH
                LDA         (TPATH),Y               ;GET CHARCTER
                AND         #$7F                    ;WE'RE NOT INTERESTED IN HIGH ORDER
BIT.
                INX                                 ;PREPARE FOR NEXT CHARACTER.
                INY
                CMP         #$2F                    ;IS IT DELIMITER "/"?
                BEQ         ENDNAME                 ;BRANCH IF IT IS.
                CMP         #$61                    ;IS IT LOWER CASE CHARACTER?
                BCC         NOTLOWER                ;BRANCH IF NOT.
                AND         #$5F                    ;UPSHIFT TO UPPER CASE
NOTLOWER        STA         PATHBUF,X               ;STORE CHARCTER.
                INC         NAMCNT                  ;IS IT THE FIRST OF A LOCAL NAME?
                BNE         NOTFRST                 ;BRANCH IF NOT.
                INC         NAMCNT                  ;KICK COUNT TO 1
                BNE         TSTALFA                 ;FIRST CHAR. MUST BE ALPHA (BRANCH
ALWAYS TAKEN).
*
NOTFRST         CMP         #$2E                    ;IS IT "."?
                BEQ         SPATH3                  ;IT'S OK IF IT IS, DO NEXT CHAR.
                CMP         #$30                    ;IS IT AT LEAST "0"?
                BCC         ERRSYN                  ;REPORT SYNTAX ERROR IF NOT.
                CMP         #$3A                    ;IS IT NUMERIC?
                BCC         SPATH3                  ;OK IF IT IS, DO NEXT CHARACTER.
TSTALFA         EQU         *
                CMP         #$41                    ;IS IT AT LEAST AN "A"?
                BCC         ERRSYN                  ;REPORT ERR IF NOT.
                CMP         #$5B                    ;IS IT G.T. "Z"?
                BCC         SPATH3                  ;GET NEXT CHAR IF VALID ALPHA.
*
ERRSYN          SEC                                 ;MAKE SURE CARRY SET.
                LDA         #BADPATH
                RTS                                 ;REPORT ERROR.
*
ENDPATH         LDA         #0                      ;END PATHNAME WITH 0.
                BIT         NAMCNT                  ;ALSO MAKE SURE NAME COUNT IS POSITIVE.
                BPL         EPATH2
                STA         NAMCNT
                DEX
EPATH2          INX
                STA         PATHBUF,X
```

```
                BEQ         ERRSYN                  ;REPORT ERROR IF "/" ONLY
                STX         PATHCNT                 ;SAVE TRUE LENGTH OF PATHNAME.
                TAX                                 ;X=0 CAUSES END OF PROCESS, AFTER
ENDNAME.
*
ENDNAME         LDA         NAMCNT                  ;VALIDATE LOCAL NAME <16
                CMP         #$10
                BCS         ERRSYN
                STX         TEMPX                   ;SAVE CURRENT POINTER.
                LDX         NAMPTR                  ;GET INDEX TO BEGINNING OF LOCAL NAME.
                STA         PATHBUF,X               ;SAVE LOCAL NAME'S LENGTH.
                LDX         TEMPX                   ;RESTORE X
                BNE         SPATH2                  ;BRANCH IF MORE NAMES TO PROCESS.
                CLC                                 ;INDICATE SUCCESS!
                LDA         PRFXFLG                 ;BUT MAKE SURE ALL PATHNAMES ARE
                BNE         EN1                     ; PREFIXED OR BEGIN WITH A "/".
                LDA         PFIXPTR                 ;MUST BE NON ZERO
                BEQ         ERRSYN
EN1             RTS
*
                PAGE
SETPREFX        JSR         SETPATH                 ;CALL IS MADE HERE SO A 'NUL' PATH MAY
BE DETECTED.
                BCC         SETPRFX1                ;BRANCH IF PATHNAME OK
                LDY         PATHBUF                 ;WAS IT A NUL PATHNAME?
                BNE         PFXERR                  ;BRANCH IF TRUE SYNTAX ERROR.
                STY         PFIXPTR                 ;INDICATE NUL PREFIX
                CLC                                 ;NO ERROR.
                RTS
*
SETPRFX1        EQU         *
                JSR         FINDFILE                ;GO FIND SPECIFIED PREFIX DIRECTORY.
                BCC         SETPRFX2                ;BRANCH IF NO ERROR
                CMP         #BADPATH
                BNE         PFXERR                  ;BRANCH IF ERROR IS REAL (NOT ROOT DIR)
SETPRFX2        LDA         DFIL+D.STOR             ;MAKE SURE LAST LOCAL NAME IS DIRECTORY
TYPE.
                AND         #$D0                    ;(EITHER ROOT OR SUB)
                EOR         #$D0                    ;IS IT A DIRECTORY?
                BNE         PTYPERR                 ;REPORT WRONG TYPE.
                LDY         PRFXFLG                 ;NEW OR APPENDED PREFIX?
                BNE         SETPRFX3
                LDA         PFIXPTR                 ;APPEND NEW PREFIX TO OLD.
SETPRFX3        TAY
                SEC                                 ;FIND NEW BEGINNING OF PREFIX.
                SBC         PATHCNT
                CMP         #$C0                    ;TOO LONG?
                BCC         ERRSYN                  ;REPORT IT IF SO.
                TAX
                STA         PFIXPTR
                LDA         D.DEV                   ;SAVE DEVICE NUMBER.
                STA         P.DEV
                LDA         DFIL+D.FRST             ; AND ADDR OF FIRST BLOCK.
                STA         P.BLOK
                LDA         DFIL+D.FRST+1
                STA         P.BLOK+1
MOVPRFX         LDA         PATHBUF,Y
                STA         PATHBUF,X
                INY
                INX
                BNE         MOVPRFX
                CLC                                 ;INDICATE GOOD PREFIX.
                RTS
```

```
*
PTYPERR         LDA         #TYPERR                 ;REPORT NOT A DIRECTORY.
PFXERR          SEC                                 ;INDICATE ERROR
                RTS
*
*
                PAGE
*
GETPREFX        CLC                                 ;CALCULATE HOW BIG A BUFFER IS NEEDED
TO
                LDY         #C.PATH                 ;GET INDEX TO USERS PATHNAME BUFFER
                LDA         (PAR),Y
                STA         USRBUF
                INY
                LDA         (PAR),Y
                STA         USRBUF+1
                LDA         #0                      ;SET BUF LENGTH AT MAX.
                STA         CBYTES+1
                LDA         #$40                    ;(64 CHARACTERS MAX).
                STA         CBYTES
                JSR         VALDBUF                 ;GO VALIDATE PREFIX BUFFER ADDR.
                BCS         PFXERR
                LDY         #0                      ;Y IS INDIRECT INDEX TO USER BUFFER.
                LDA         PFIXPTR                 ;GET ADDRESS OF BEGINNING OF PREFIX
                TAX
                BEQ         NULPRFX                 ;BRANCH IF NUL PREFIX.
                EOR         #$FF                    ;GET TOTAL LENGTH OF PREFIX
                ADC         #2                      ;ADD 2 FOR LEADING AND TRAILING
SLASHES.
NULPRFX         STA         (USRBUF),Y              ;STORE LENGTH IN USERS BUFFER.
                BEQ         GOTPRFX                 ;BRANCH IF NUL PREFIX.
SENDPRFX        INY                                 ;BUMP TO NEXT USER BUF LOC.
                LDA         PATHBUF,X               ;GET NEXT CHAR OF PREFIX.
SNDLIMIT        STA         (USRBUF),Y              ;GIVE CHARACTER TO USER.
                AND         #$F0                    ;CHECK FOR LENGTH DESCRIPTOR
                BNE         SNDPFX1                 ;BRANCH IF REGULAR CHARACTER
                LDA         #$2F                    ;OTHERWISE, SUBSTITUTE A SLASH "/".
                BNE         SNDLIMIT                ;BRANCH ALWAYS.
*
SNDPFX1         INX
                BNE         SENDPRFX                ;BRANCH IF MORE TO SEND.
                INY
                LDA         #$2F                    ;END WITH "/"
                STA         (USRBUF),Y
GOTPRFX         CLC                                 ;INDICATE NO ERROR.
                RTS
                PAGE
FINDFCB         LDY         #C.REFNUM               ;INDEX TO REFERENCE NUMBER
                LDA         (PAR),Y                 ;IS IT A VALID FILE NUMBER?
                BEQ         REEFER                  ;MUST NOT BE 0!
                CMP         #9                      ;MUST BE 1 TO 8 ONLY.
                BCS         REEFER                  ;USER MUST BE STONED...
                PHA
                SBC         #0                      ;(ACTUALLY SUBTRACTS 1).
                LSR         A                       ;SHIFT LOW 3 BITS TO HIGH BITS.
                ROR         A
                ROR         A
                ROR         A                       ;EFFECTIVE MULTIPLY BY 32.
                STA         FCBPTR                  ;LATTER USED AS AN INDEX TO FCB.
                TAY                                 ; LIKE NOW.
                PLA                                 ;RESTORE REFNUM IN ACC.
                CMP         FCB+FCBREFN,Y           ;IS IT AN OPEN REFERENCE?
                BNE         ERRNOREF                ;BRANCH IF NOT.
```

```
*
FNDFCBUF        LDA         FCB+FCBFBUF,Y           ;GET PAGE ADDR. OF FILE BUFFER.
                JSR         GETBUFADR               ;GET FILES ADDRESS INTO BUFADDRL&H
                LDX         BUFADDRH                ;(Y=FCBPTR PRESERVED)
                BEQ         FCBDEAD                 ;REPORT FCB SCREWED UP!!!
                STX         DATPTR+1                ;SAVE POINTER TO DATA AREA OF BUFFER.
                INX
                INX                                 ;INDEX BLOCK ALWAYS 2 PAGES AFTER DATA.
                STX         TINDX+1
                LDA         FCB+FCBDEVN,Y           ;ALSO SET UP DEVICE NUMBER.
                STA         DEVNUM
                LDA         BUFADDRL
                STA         DATPTR
                STA         TINDX                   ;INDEX AND DATA BUFFERS ALWAYS ON PAGE
BOUNDARIES.
*
FNDFVOL         TAX                                 ;SEARCH FOR ASSOCIATED VCB.
                LDA         VCB+VCBDEV,X
                CMP         FCB+FCBDEVN,Y           ;IS THIS VCB THE SAME DEVICE?
                BEQ         TSTVOPEN                ;IF IT IS, MAKE SURE VOLUME IS ACTIVE.
*
NXTFVOL         TXA                                 ;ADJUST INDEX TO NEXT VCB.
                CLC
                ADC         #VCBSIZE
                BCC         FNDFVOL                 ;LOOP UNTIL VOLUME FOUND.
                LDA         #VCBERR                 ;REPORT OPEN FILE HAS NO VOLUME...
                JSR         SYSDEATH                ; AND KILL THE SYSTEM.
*
FCBDEAD         LDA         #FCBERR                 ;REPORT FCB TRASHED!
                JSR         SYSDEATH                ; AND KILL THE SYSTEM.
*
TSTVOPEN        LDA         VCB,X                   ;MAKE SURE THIS VCB IS OPEN
                BEQ         NXTFVOL                 ;BRANCH IF IT IS NOT ACTIVE.
                STX         VCBPTR                  ;SAVE POINTER TO GOOD VCB.
                CLC                                 ;INDICATE ALL IS WELL.
                RTS
*
ERRNOREF        LDA         #0                      ;DROP A ZERO INTO THIS FCB TO
                STA         FCBPTR+FCBREFN,Y        ; SHOW FREE FCB
*
REEFER          LDA         #BADREFNUM              ;TELL USER THAT REQUESTED REFNUM
                SEC                                 ; IS ILLEGAL (OUT OF RANGE) FOR THIS
CALL.
                RTS
*
*
ONLINE          JSR         MVDBUFR                 ;MOVE USER SPECIFIED BUFFER POINTER TO
USRBUF.
                LDA         #0                      ;FIGURE OUT HOW BIG BUFFER HAS TO BE.
                STA         CBYTES
                STA         CBYTES+1
                LDY         #C.DEVNUM
                LDA         (PAR),Y                 ;IF ZERO THEN CBYTES=$100, ELSE =$010
FOR ONE DEVICE.
                BEQ         OLIN1                   ;BRANCH IF ALL DEVICES.
                LDA         #$10
                STA         CBYTES
                BNE         OLIN2
OLIN1           LDA         #1                      ;ALLOW FOR UP TO 16 DEVICES.
                STA         CBYTES+1
OLIN2           JSR         VALDBUF                 ;GO VALIDATE BUFFER RANGE AGAINST
ALLOCATED MEMORY.
                BCS         ONLINERR
```

```
                LDA         #0                      ;ZERO OUT USER BUFFER SPACE.
                LDY         CBYTES
OLIN3           DEY
                STA         (USRBUF),Y              ;ZERO EITHER 16 OR 256 BYTES.
                BNE         OLIN3                   ;BRANCH IF MORE TO ZERO.
                STA         NAMPTR                  ;USE NAMPTR AS POINTER TO USER BUFFER.
                LDY         #C.DEVNUM               ;GET DEVICE NUMBER AGAIN.
                LDA         (PAR),Y
                AND         #$F0
                BNE         ONLINE1                 ;BRANCH IF ONLY 1 DEVICE TO PROCESS.
                JSR         MVDEVNUMS               ;GET LIST OF CURRENTLY RECOGNIZED
DEVICES.
OLIN4           STX         TEMPX                   ;SAVE INDEX TO LAST ITEM ON LIST.
                LDA         LOKLST,X                ;GET NEXT DEVICE #
                JSR         ONLINE1                 ;LOG THIS VOLUME AND RETURN IT'S NAME
TO USER.
OLIN5           LDA         NAMPTR                  ;BUMP POINTER FOR NEXT DEVICE.
                CLC
                ADC         #$10
                STA         NAMPTR
                LDX         TEMPX                   ;GET INDEX TO DEVICE LIST.
                DEX                                 ;INDEX TO NEXT DEVICE
                BPL         OLIN4                   ;BRANCH IF THERE IS ANOTHER DEVICE.
                LDA         #0                      ;NO ERRORS FOR MULIPLE ON-LINE.
                CLC                                 ;INDICATE GOOD ON ALL VOLUMES.
ONLINERR        RTS
*
                PAGE
ONLINE1         STA         DEVNUM                  ;SAVE DESIRED DEVICE TO LOOK AT.
                JSR         FNDDVCB                 ;SEE IF IT HAS ALREADY BEEN LOGGED IN.
                BCS         OLINERR1                ;BRANCH IF VCB IS FULL
                LDX         #0                      ;READ IN ROOT (VOLUME) DIRECTORY.
                LDA         #2
                JSR         RDBLK                   ;READ IT INTO GENERAL PURPOSE BUFFER.
*------------------- See Rev Note #28 ----------------------------
                bcc         volfound                ; branch if the read was OK.
                pha                                 ; save error value on stack.
                ldx         vcbptr                  ; use vcbptr as an index.
                lda         vcb+vcbstat,x           ; don't zero VCB if there
                bmi         rtrnerr                 ; active files present!
                ldy         #vcbsize                ; zero the VCB entry just incase the
                lda         #0                      ; volume was removed from the device
zeroit          sta         vcb,x                   ; list by the user.
                inx                                 ;
                dey                                 ;
                bne         zeroit                  ; branch if not done zeroing VCB.
rtrnerr         pla                                 ; get error code back from stack.
*-----------------------------------------------------------------
                BCS         OLINERR1                ;BRANCH IF UNABLE TO READ.
*------------------- See Rev Note #28 ----------------------------
volfound        equ         *
*-----------------------------------------------------------------
                LDX         VCBPTR                  ;NOW DETERMINE IF IT'S AN ACTIVE
VOLUME.
                LDA         VCB,X                   ;HAS IT BEEN LOGGED IN BEFORE?
                BEQ         OLIN11                  ;BRANCH IF NOT.
                LDA         VCB+VCBSTAT,X           ;IT HAS, ARE THERE ACTIVE FILES?
                BMI         OLIN12                  ;BRANCH IF THE VOLUME IS CURRENTLY
BUSY.
OLIN11          JSR         LOGVCB1                 ;GO LOG IT IN.
                BCS         OLINERR1                ;BRANCH IF THERE IS SOME PROBLEM (LIKE
NOTSOS).
```

```
                LDA         #DUPVOL                 ;ANTICIPATE A DUPLICATE ACTIVE VOLUME
EXISTS.
                BIT         DUPLFLAG
                BMI         OLINERR1                ;BRANCH IF WE GUESSED RIGHT.
OLIN12          LDX         VCBPTR                  ;RESTORE VCBPTR JUST IN CASE WE LOST
IT.
                JSR         CMPVCB                  ;DOES READ IN VOLUME COMPARE WITH
LOGGED VOLUME?
                LDA         #DSWTCHED               ;ANTICIPATE WRONG VOLUME MOUNTED IN
ACTIVE DEVICE.
                BCC         OLIN13                  ;BRANCH IF NO PROBLEM!
OLINERR1        PHA                                 ;SAVE ERROR CODE.
                JSR         SVDEVN                  ;TELL USER WHAT DEVICE WE LOOKED AT.
                PLA                                 ;GET ERROR CODE AGAIN
                INY                                 ;TELL USER WHAT ERROR WAS ENCOUNTERED
ON THIS DEVICE
                STA         (USRBUF),Y
                CMP         #DUPVOL                 ;WAS IT A DUPLICATE VOLUME ERROR?
                BNE         OLINERR2                ;BRANCH IF NOT,
                INY                                 ;OTHERWISE TELL USER WHICH OTHER DEVICE
HAS SAME NAME.
                LDX         VCBENTRY
                LDA         VCB+VCBDEV,X
                STA         (USRBUF),Y
                LDA         #0                      ;CLEAR DUPLICATE FLAG.
                STA         DUPLFLAG
                LDA         #DUPVOL                 ;RESTORE ERROR CODE.
OLINERR2        SEC                                 ;INDICATE ERROR.
                RTS
*
OLIN13          LDA         VCB,X                   ;GET VOLUME NAME COUNT
                STA         NAMCNT
                LDY         NAMPTR                  ;INDEX TO USER'S BUFFER
OLIN14          LDA         VCB,X                   ;MOVE NAME TO USER'S BUFFER
                STA         (USRBUF),Y
                INX
                INY
                DEC         NAMCNT                  ;LOOP UNTIL ALL CHARACTERS MOVED.
                BPL         OLIN14
SVDEVN          LDY         NAMPTR                  ;INDEX TO FIRST BYTE OF THIS ENTRY.
                LDA         DEVNUM                  ;PUT DEVICE NUMBER IN UPPER NIBBLE OF
THIS BYTE.
                ORA         (USRBUF),Y              ;LOWER NIBBLE IS NAME LENGTH.
                STA         (USRBUF),Y
                CLC                                 ;INDICATE NO ERRORS.
                RTS
*

; ###############################################################################################
; #   END OF FILE:  BFMGR.ONE
; #   LINES      :  380
; #   CHARACTERS :  23259
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.BFMGR.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  BFMGR
; ###############################################################################################

                sbtl        'prodos block file manager'
bfmgr           stx         command                 ;what call?
                lda         disptch,x               ;translate into command address
                asl         a                       ;(bit 7 indicates a pathname to
preprocess)
                sta         cmdtemp
                and         #$3f                    ;(bit 6 is refnum preprocess, 5 is for
time, so strip em.)
                tax
                lda         cmdtable,x              ;move address for indirect jump.
                sta         goadr
                lda         cmdtable+1,x            ;(high byte)
                sta         goadr+1
                lda         #$20                    ; init "backup bit flag"
                sta         bkbitflg                ; to say "file modified"
                bcc         nopath
                jsr         setpath                 ;go process pathname before calling
command
                ifeq        os-prodos
                bcs         errorsys                ; branch if bad name.
                fin
*
******************** see rev note #en2 ******************
*
                ifeq        os-ednet
                bcc         checkatalk              ; check to see if the path is for
appletalk.
                bcs         errorsys                ; report error if bad name.
checkatalk      equ         *
                lda         prfxflg                 ; is it a full pathname?
                bne         full                    ; yes, so branch.
                lda         atpfxflg                ; is prefix for /atalk?
                beq         notatalk                ; no, so branch.
                bne         doatalk                 ; go do /atalk command.
full            ldx         #0                      ; see if volume name is 5 chars long.
                ldy         pathbuf,x               ; get length of volume name.
                cpy         #5                      ; is it 5 chars long like /atalk?
                bne         notatalk                ; no, branch and continue as normal.
chkatalk        lda         pathbuf,y               ; now see if volume name is 'atalk'.
                cmp         atalkstr-1,y            ;
                bne         notatalk                ;
                dey                                 ;
                bne         chkatalk                ;
doatalk         equ         *
                ldx         cmdtemp                 ; get command table offset back.
                lda         atcmdtable,x            ; move atalk command address
                sta         goadr                   ; for indirect jump.
                lda         atcmdtable+1,x          ;
                sta         goadr+1                 ;
                bne         execute                 ; branch always, and do atalk cmd!
notatalk        equ         *
                fin
************************************************************
```

```
nopath          asl         cmdtemp                 ;test for refnum preprocessing
                bcc         nopreref
                jsr         findfcb                 ;go set up pointers to fcb and vcb of
this file.
                bcs         errorsys                ;branch if any errors are encountered.
nopreref        asl         cmdtemp                 ;lastly check for necessity of time
stamp.
                bcc         execute
                jsr         datetime                ;(no error posible)
execute         jsr         gocmd                   ;execute command
                bcc         goodop                  ;branch if successful
*
errorsys        jsr         syserr
goodop          rts                                 ; good return
*
gocmd           jmp         (goadr)
                page
setpath         equ         *
                ldy         #c.path                 ;index to pathname pointer
                lda         (par),y                 ;get low pointer addr.
                sta         tpath
                iny
                lda         (par),y                 ; and hi pointer addr.
                sta         tpath+1
*
synpath         equ         *                       ;entry used by rename for second
pathname.
                ldx         #0                      ;x register is used as index to pathbuf
                ldy         #0                      ;y register is index to input pathname.
                stx         prfxflg                 ;assume prefix is in use.
                stx         pathbuf                 ;mark pathbuf to indicate nothing
processed.
                lda         (tpath),y               ;validate pathname length>0, and <65
                beq         errsyn
                cmp         #$41
                bcs         errsyn
                sta         pathcnt                 ;this is used to compare for
                inc         pathcnt                 ; end of pathname processing.
                iny                                 ;now check for full pathname...
                lda         (tpath),y               ;(full name if starts with "/".)
                ora         #$80
                cmp         #$af
                bne         spath2                  ;branch if prefix appended
                sta         prfxflg                 ;set prefix flag to indicate prefix not
used.
                iny                                 ;index to first charcter of pathname.
*
spath2          equ         *
                lda         #$ff                    ;set current position of pathbuf
                sta         pathbuf,x               ; to indicate end of pathname.
                sta         namcnt                  ;also indicate no characters processed
in local name.
                stx         namptr                  ;preserve pointer to local name length
byte.
spath3          equ         *
                cpy         pathcnt                 ;done with pathname processing?
                bcs         endpath
                lda         (tpath),y               ;get charcter
                and         #$7f                    ;we're not interested in high order
bit.
                inx                                 ;prepare for next character.
                iny
                cmp         #$2f                    ;is it delimiter "/"?
```

```
                beq         endname                 ;branch if it is.
                cmp         #$61                    ;is it lower case character?
                bcc         notlower                ;branch if not.
                and         #$5f                    ;upshift to upper case
notlower        sta         pathbuf,x               ;store charcter.
                inc         namcnt                  ;is it the first of a local name?
                bne         notfrst                 ;branch if not.
                inc         namcnt                  ;kick count to 1
                bne         tstalfa                 ;first char. must be alpha (branch
always taken).
*
notfrst         cmp         #$2e                    ;is it "."?
                beq         spath3                  ;it's ok if it is, do next char.
                cmp         #$30                    ;is it at least "0"?
                bcc         errsyn                  ;report syntax error if not.
                cmp         #$3a                    ;is it numeric?
                bcc         spath3                  ;ok if it is, do next character.
tstalfa         equ         *
                cmp         #$41                    ;is it at least an "a"?
                bcc         errsyn                  ;report err if not.
                cmp         #$5b                    ;is it g.t. "z"?
                bcc         spath3                  ;get next char if valid alpha.
*
errsyn          sec                                 ;make sure carry set.
                lda         #badpath
                rts                                 ;report error.
*
endpath         lda         #0                      ;end pathname with 0.
                bit         namcnt                  ;also make sure name count is positive.
                bpl         epath2
                sta         namcnt
                dex
epath2          inx
                sta         pathbuf,x
                beq         errsyn                  ;report error if "/" only
                stx         pathcnt                 ;save true length of pathname.
                tax                                 ;x=0 causes end of process, after
endname.
*
endname         lda         namcnt                  ;validate local name <16
                cmp         #$10
                bcs         errsyn
                stx         tempx                   ;save current pointer.
                ldx         namptr                  ;get index to beginning of local name.
                sta         pathbuf,x               ;save local name's length.
                ldx         tempx                   ;restore x
                bne         spath2                  ;branch if more names to process.
                clc                                 ;indicate success!
                lda         prfxflg                 ;but make sure all pathnames are
                bne         en1                     ; prefixed or begin with a "/".
                lda         newpfxptr               ; must be non zero
                beq         errsyn
en1             rts
*
                page
setprefx        jsr         setpath                 ;call is made here so a 'nul' path may
be detected.
                bcc         setprfx1                ;branch if pathname ok
                ldy         pathbuf                 ;was it a nul pathname?
                bne         pfxerr                  ;branch if true syntax error.
                jsr         stypfx                  ;indicate nul prefix
                clc                                 ;no error.
                rts
```

```
*
setprfx1        equ         *
                jsr         findfile                ;go find specified prefix directory.
                bcc         setprfx2                ;branch if no error
                cmp         #badpath
                bne         pfxerr                  ;branch if error is real (not root dir)
setprfx2        lda         dfil+d.stor             ;make sure last local name is directory
type.
                and         #$d0                    ;(either root or sub)
                eor         #$d0                    ;is it a directory?
                bne         ptyperr                 ;report wrong type.
                ldy         prfxflg                 ;new or appended prefix?
                bne         setprfx3
                lda         newpfxptr               ;append new prefix to old.
setprfx3        tay
                sec                                 ;find new beginning of prefix.
                sbc         pathcnt
                cmp         #$c0                    ;too long?
                bcc         errsyn                  ;report it if so.
                tax
                jsr         stapfx
                lda         d.dev                   ;save device number.
                sta         p.dev
                lda         dfil+d.frst             ; and addr of first block.
                sta         p.blok
                lda         dfil+d.frst+1
                sta         p.blok+1
movprfx         lda         pathbuf,y
                sta         pathbuf,x
                iny
                inx
                bne         movprfx
                clc                                 ;indicate good prefix.
                rts
*
ptyperr         lda         #typerr                 ;report not a directory.
pfxerr          sec                                 ;indicate error
                rts
*
*
                page
*
getprefx        clc                                 ;calculate how big a buffer is needed
to
                ldy         #c.path                 ;get index to users pathname buffer
                lda         (par),y
                sta         usrbuf
                iny
                lda         (par),y
                sta         usrbuf+1
                lda         #0                      ;set buf length at max.
                sta         cbytes+1
                lda         #$40                    ;(64 characters max).
                sta         cbytes
                jsr         valdbuf                 ;go validate prefix buffer addr.
                bcs         pfxerr
                ldy         #0                      ;y is indirect index to user buffer.
                lda         newpfxptr               ;get address of beginning of prefix
                tax
                beq         nulprfx                 ;branch if nul prefix.
                eor         #$ff                    ;get total length of prefix
                adc         #2                      ;add 2 for leading and trailing
slashes.
```

```
nulprfx         sta         (usrbuf),y              ;store length in users buffer.
                beq         gotprfx                 ;branch if nul prefix.
sendprfx        iny                                 ;bump to next user buf loc.
                lda         pathbuf,x               ;get next char of prefix.
sndlimit        sta         (usrbuf),y              ;give character to user.
                and         #$f0                    ;check for length descriptor
                bne         sndpfx1                 ;branch if regular character
                lda         #$2f                    ;otherwise, substitute a slash "/".
                bne         sndlimit                ;branch always.
*
sndpfx1         inx
                bne         sendprfx                ;branch if more to send.
                iny
                lda         #$2f                    ;end with "/"
                sta         (usrbuf),y
gotprfx         clc                                 ;indicate no error.
                rts
                page
findfcb         ldy         #c.refnum               ;index to reference number
                lda         (par),y                 ;is it a valid file number?
                beq         reefer                  ;must not be 0!
                cmp         #9                      ;must be 1 to 8 only.
                bcs         reefer                  ;user must be stoned...
                pha
                sbc         #0                      ;(actually subtracts 1).
                lsr         a                       ;shift low 3 bits to high bits.
                ror         a
                ror         a
                ror         a                       ;effective multiply by 32.
                sta         fcbptr                  ;latter used as an index to fcb.
                tay                                 ; like now.
                pla                                 ;restore refnum in acc.
                cmp         fcb+fcbrefn,y           ;is it an open reference?
                bne         errnoref                ;branch if not.
*
fndfcbuf        lda         fcb+fcbfbuf,y           ;get page addr. of file buffer.
                jsr         getbufadr               ;get files address into bufaddrl&h
                ldx         bufaddrh                ;(y=fcbptr preserved)
                beq         fcbdead                 ;report fcb screwed up!!!
                stx         datptr+1                ;save pointer to data area of buffer.
                inx
                inx                                 ;index block always 2 pages after data.
                stx         tindx+1
                lda         fcb+fcbdevn,y           ;also set up device number.
                sta         devnum
                lda         bufaddrl
                sta         datptr
                sta         tindx                   ;index and data buffers always on page
boundaries.
*
fndfvol         tax                                 ;search for associated vcb.
                lda         vcb+vcbdev,x
                cmp         fcb+fcbdevn,y           ;is this vcb the same device?
                beq         tstvopen                ;if it is, make sure volume is active.
*
nxtfvol         txa                                 ;adjust index to next vcb.
                clc
                adc         #vcbsize
                bcc         fndfvol                 ;loop until volume found.
                lda         #vcberr                 ;report open file has no volume...
                jsr         sysdeath                ; and kill the system.
*
fcbdead         lda         #fcberr                 ;report fcb trashed!
```

```
                jsr         sysdeath                ; and kill the system.
*
tstvopen        lda         vcb,x                   ;make sure this vcb is open
                beq         nxtfvol                 ;branch if it is not active.
                stx         vcbptr                  ;save pointer to good vcb.
                clc                                 ;indicate all is well.
                rts
*
errnoref        lda         #0                      ;drop a zero into this fcb to
                sta         fcbptr+fcbrefn,y        ; show free fcb
*
reefer          lda         #badrefnum              ;tell user that requested refnum
                sec                                 ; is illegal (out of range) for this
call.
                rts
*
*
online          jsr         mvdbufr                 ;move user specified buffer pointer to
usrbuf.
                lda         #0                      ;figure out how big buffer has to be.
                sta         cbytes
                sta         cbytes+1
                ldy         #c.devnum
                lda         (par),y                 ;if zero then cbytes=$100, else =$010
for one device.
                beq         olin1                   ;branch if all devices.
                lda         #$10
                sta         cbytes
                bne         olin2
olin1           lda         #1                      ;allow for up to 16 devices.
                sta         cbytes+1
olin2           jsr         valdbuf                 ;go validate buffer range against
allocated memory.
                bcs         onlinerr
                lda         #0                      ;zero out user buffer space.
                ldy         cbytes
olin3           dey
                sta         (usrbuf),y              ;zero either 16 or 256 bytes.
                bne         olin3                   ;branch if more to zero.
                sta         namptr                  ;use namptr as pointer to user buffer.
                ldy         #c.devnum               ;get device number again.
                lda         (par),y
                and         #$f0
                bne         online1                 ;branch if only 1 device to process.
                jsr         mvdevnums               ;get list of currently recognized
devices.
olin4           stx         tempx                   ;save index to last item on list.
                lda         loklst,x                ;get next device #
                jsr         online1                 ;log this volume and return it's name
to user.
olin5           lda         namptr                  ;bump pointer for next device.
                clc
                adc         #$10
                sta         namptr
                ldx         tempx                   ;get index to device list.
                dex                                 ;index to next device
                bpl         olin4                   ;branch if there is another device.
                lda         #0                      ;no errors for muliple on-line.
                clc                                 ;indicate good on all volumes.
onlinerr        rts
*
                page
online1         sta         devnum                  ;save desired device to look at.
```

```
                jsr         fnddvcb                 ;see if it has already been logged in.
                bcs         olinerr1                ;branch if vcb is full
                ldx         #0                      ;read in root (volume) directory.
                lda         #2
                jsr         rdblk                   ;read it into general purpose buffer.
                ldx         vcbptr                  ; use x as an index to the vcb entry.
*------------------- see rev note #28 ----------------------------
*
* this fix is to remove vcb entries that correspond to devices that
* are no longer in the device list (i.e. removed by the user).
*
                bcc         volfound                ; branch if the read was ok.
                tay                                 ; save off error value in y.
                lda         vcb+vcbstat,x           ; don't take the vcb off line if
                bne         rtrnerr                 ; there are active files present!
                sta         vcb,x                   ; now take the volume off line.
                sta         vcb+vcbdev,x            ;
rtrnerr         tya                                 ; now return error to accumulator.
                bcs         olinerr1                ;branch if unable to read.
volfound        equ         *
*-------------------------------------------------------------
                lda         vcb,x                   ;has it been logged in before?
                beq         olin11                  ;branch if not.
                lda         vcb+vcbstat,x           ;it has, are there active files?
                bmi         olin12                  ;branch if the volume is currently
busy.
olin11          jsr         logvcb1                 ;go log it in.
                bcs         olinerr1                ;branch if there is some problem (like
notsos).
                lda         #dupvol                 ;anticipate a duplicate active volume
exists.
                bit         duplflag
                bmi         olinerr1                ;branch if we guessed right.
olin12          ldx         vcbptr                  ;restore vcbptr just in case we lost
it.
                jsr         cmpvcb                  ;does read in volume compare with
logged volume?
                lda         #dswtched               ;anticipate wrong volume mounted in
active device.
                bcc         olin13                  ;branch if no problem!
olinerr1        pha                                 ;save error code.
                jsr         svdevn                  ;tell user what device we looked at.
                pla                                 ;get error code again
                iny                                 ;tell user what error was encountered
on this device
                sta         (usrbuf),y
                cmp         #dupvol                 ;was it a duplicate volume error?
                bne         olinerr2                ;branch if not,
                iny                                 ;otherwise tell user which other device
has same name.
                ldx         vcbentry
                lda         vcb+vcbdev,x
                sta         (usrbuf),y
                lda         #0                      ;clear duplicate flag.
                sta         duplflag
                lda         #dupvol                 ;restore error code.
olinerr2        sec                                 ;indicate error.
                rts
*
olin13          lda         vcb,x                   ;get volume name count
                sta         namcnt
                ldy         namptr                  ;index to user's buffer
olin14          lda         vcb,x                   ;move name to user's buffer
```

```
               sta          (usrbuf),y
               inx
               iny
               dec          namcnt                    ;loop until all characters moved.
               bpl          olin14
svdevn         ldy          namptr                    ;index to first byte of this entry.
               lda          devnum                    ;put device number in upper nibble of
this byte.
               ora          (usrbuf),y                ;lower nibble is name length.
               sta          (usrbuf),y
               clc                                    ;indicate no errors.
               rts
*

; ###############################################################################################
; #   END OF FILE:  BFMGR
; #   LINES      :  410
; #   CHARACTERS :  24879
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.BOOT.SRC.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  BOOT.SRC
; ###############################################################################################

*<esc>e
                sbtl         'universal boot loader - stage 2'
                page
                rep          60
* prodos universal boot loader.  this is the second stage boot
*   for all apple manufactured apple ii disk drives.
* it is located at block zero (0) of a prodos or sos formatted
*   disk(ette).  if booted in apple /// native mode, the regular
*   sos boot will be attempted.
                rep          60
                skp          1
                org          $800
dcmd            equ          $42                        ;disk command (=1 for read)
unit            equ          $43                        ;(16*slot)+(128*(drive-1))
buff            equ          $44                        ;ram address
blok            equ          $46                        ;disk address
                skp          1
dent            equ          $48                        ;device call entry address.
idxl            equ          $4a                        ;pointer to low page of index block
idxh            equ          $4c                        ;pointer to high page of index block
idxp            equ          $4e                        ;index byte pointer.
iobuff          equ          $60
                skp          1
* the following are for disk ii only:
                skp          1
dbuf            equ          $26
slotz           equ          $2b
oddbits         equ          $3c
sector          equ          $3d
trktmp          equ          $40
track           equ          $41
prior           equ          $50
trkn            equ          $51
rtrycnt         equ          $52
curtrk          equ          $53
trkcnt          equ          $54
*
q6l             equ          $c08c
motoron         equ          $c089
motoroff        equ          $c088
phaseoff        equ          $c080
nbuf1           equ          $300
dnib            equ          $2d6
                skp          1
* directory dependent stuff...
clrscrn         equ          $fc58
scrn            equ          $5ae
dostyp          equ          $ff
sosid           equ          $c00
entlen          equ          sosid+$23
kernel          equ          $2000
                page
xboot           dfb          $01                        ;(prodos boot id)
```

```
entry           sec                             ;(apple iii enters xboot 'ora $38')
                bcs         entry1              ;branch if not apple iii native mode.
                jmp         goapl3              ;go do apple iii boot!
                skp         1
entry1          stx         unit                ;save unit number.
                cmp         #$03                ;for disk ii.
                php                             ;save result, it may be irrelevent.
                txa                             ;find out if disk ii.
                and         #$70                ;strip drive # if any.
                lsr         a
                lsr         a                   ;get slot address.
                lsr         a
                lsr         a
                ora         #$c0
                sta         dent+1
                ldy         #$ff                ;look at last byte.
                sty         dent
                plp                             ;restore carry (if disk ii & sect 0&2
read carry set).
                iny                             ;make y=0
                lda         (dent),y            ;get device entry addr.
                bne         ndsk2               ;branch if not disk ii (16 sector).
                bcs         isdsk2              ;branch if it is disk ii, but block 0
read.
                lda         #3                  ;make rom read only sector 2
                sta         xboot               ;to complete block 0
                inc         sector              ;(was = 1)
                lda         dent+1              ;do rts to re-enter rom.
                pha
                lda         #$5b
                pha
                rts                             ;go read sector 2 into $900.
                skp         1
isdsk2          sta         trktmp              ;make sure previous track =0
                sta         dent                ;and dent points at beginning of slot
                ldy         #$63                ;move code from card to ram
mvboot          lda         (dent),y
                sta         zzstart-$5e,y
                iny
                cpy         #$eb                ;have we moved enough?
                bne         mvboot
                ldx         #6                  ;now modify code to handle errors.
modboot         ldy         mods,x
                lda         chgs,x
                sta         zzstart,y
                lda         endcode,x
                sta         zzzend,x
                dex
                bpl         modboot
                lda         #<d2io              ;reset device entry
                sta         dent+1              ; to point at disk ii routines.
                lda         #>d2io              ;get low addr (must be <$80)
ndsk2           ldy         #0                  ;make sure y=0 again.
                cmp         #$f9
                bcs         bterr1              ;branch if not bootable device.
                sta         dent                ;save low adr of device call entry.
                sty         iobuff
                sty         idxl
                sty         idxh                ;(y=0)
                sty         idxp
                sty         blok+1
                iny
                sty         dcmd                ;set read command.
```

```
                iny
                sty     blok                    ; to read directory blocks
                lda     #$c                     ; 2-5 at $c00
                sta     iobuff+1
                sta     idxl+1
                page
rddir           jsr     goread                  ;call read block routine.
                bcs     bterr2                  ;give up on error.
                inc     iobuff+1
                inc     iobuff+1
                inc     blok
                lda     blok                    ;have all directory blocks been read?
                cmp     #6
                bcc     rddir                   ;loop if not.
                skp     1
                lda     sosid                   ;is it a prodos (sos) directory?
                ora     sosid+1
bterr1          bne     booterr                 ;branch if not.
                lda     #4                      ;begin look-up with first entry past
header.
                bne     nxdent1                 ;branch always
nxdent          lda     idxl
nxdent1         clc
                adc     entlen                  ;bump to next directory entry.
                tay                             ;save in y for now.
                bcc     nxdent2                 ;branch if not a page cross.
                inc     idxl+1
                lda     idxl+1                  ;check for new block.
                lsr     a                       ;if even then new block.
                bcs     nxdent2
                cmp     #$a                     ;have all file names been compared?
                beq     nopro                   ;branch if no pro.kernel.
                ldy     #4                      ;else, begin at block beginning.
nxdent2         sty     idxl                    ;note: this method treats garbage at
                lda     sysname                 ; the end of the dir block as an entry.
                and     #$f                     ;get target name length.
                tay
lookpro         lda     (idxl),y                ;look for matching name.
                cmp     sysname,y               ;last to first method.
                bne     nxdent                  ;branch if no match.
                dey                             ;else check all characters.
                bpl     lookpro                 ;including length/storage type.
                and     #$f0                    ;make sure storage type is a tree!
                cmp     #$20
                bne     booterr                 ;branch if not.
                ldy     #$10                    ;get file type & index block addr.
                lda     (idxl),y
                cmp     #dostyp                 ;is it a system file?
                bne     booterr
                iny
                lda     (idxl),y
                sta     blok
                iny
                lda     (idxl),y
                sta     blok+1
                lda     #0                      ;now set up to read kernel.
                sta     idxl
                ldy     #$1e                    ;read index block at $1e00 and
                sty     idxl+1                  ; kernel at $2000
                sty     iobuff+1
                iny
                sty     idxh+1
rdkernl         jsr     goread                  ;read index block.
```

```
bterr2          bcs         booterr
                inc         iobuff+1
                inc         iobuff+1
                ldy         idxp                    ;get index pointer
                inc         idxp                    ;bump for next time.
                lda         (idxl),y
                sta         blok
                lda         (idxh),y                ;high disk addr.
                sta         blok+1
                ora         (idxl),y                ;if both=0 then done.
                bne         rdkernl                 ;branch if more to read.
                skp         1
                jmp         kernel                  ;go execute kernel code.
                skp         1
nopro           equ         *
booterr         equ         *
                jmp         quitmes
                skp         1
                msb         off
sysname         dfb         $26
                asc         "prodos         " "
                skp         1
goread          lda         iobuff
                sta         buff
                lda         iobuff+1
                sta         buff+1
                jmp         (dent)
*
mods            dfb         mod1,mod2,mod3,mod4
                dfb         mod5,mod6,mod7
*
chgs            dfb         chg1,chg2,chg3,chg4
                dfb         chg5,chg6,chg7
*
endcode         equ         *
                ldx         slotz
                clc
                rts
                jmp         seek
*
goapl3          equ         *+$9800
                lda         #$9f                    ;make apple iii boot using block 1.
                pha                                 ;(the return address is $a000)
                lda         #$ff
                pha
                lda         #1                      ;read block 1.
                ldx         #0
                jmp         $f479
                skp         2
quitmes         jsr         clrscrn                 ;clear video.
                ldy         #meslen                 ;print message centered on screen.
prmess          lda         errmess,y
                sta         scrn,y
                dey
                bpl         prmess
hang            jmp         hang
*
                msb         on
meslen          equ         28
errmess         asc         "*** unable to load prodos ***"
*
                page
setphase        lda         curtrk                  ;get current track
```

```
clrphase        and         #3                      ;mask for 1 of 4 phases
                rol         a                       ;double for phaseon/off index
                ora         slotz
                tax
                lda         phaseoff,x              ;turn on/off one phase
                lda         #$2c
************************
*                      *
*   mswait subroutine  *
*                      *
************************
mswait          ldx         #$11
msw1            dex                                 ;delay 86 usec.
                bne         msw1
                sbc         #$1                     ;done 'n' intervals?
                bne         mswait                  ;(a-reg counts)
                ldx         slotz                   ;restore x-reg
                rts
*
*
d2io            lda         blok                    ;figure out track & sector.
                and         #7                      ;strip track for now.
                cmp         #4
                and         #3
                php
                asl         a
                plp
                rol         a                       ;now we have the first sector of block.
                sta         sector
                lda         blok+1                  ;get high block #
                lsr         a                       ;shift hi addr to carry.
                lda         blok                    ;now figure track #
                ror         a
                lsr         a
                lsr         a
                sta         track
                asl         a
                sta         trkn
                lda         buff+1
                sta         dbuf+1
                ldx         slotz
                lda         motoron,x
                jsr         rdsector                ;go read sector.
                inc         dbuf+1                  ;bump address
                inc         sector
                inc         sector                  ;and sector #
                bcs         quitrd                  ;branch if error.
                jsr         rdsector
quitrd          ldy         motoroff,x
erretrn         rts                                 ;return error status in carry.
*
rdsector        equ         *                       ;do seek then read sector.
*
seek            lda         trktmp                  ;get track we're on.
                asl         a
                sta         curtrk
                lda         #$0
                sta         trkcnt                  ;halftrack count.
seek2           lda         curtrk                  ;save curtrk for
                sta         prior                   ;delayed turnoff.
                sec
                sbc         trkn                    ;delta-tracks.
                beq         seekend                 ;br if curtrk=destination
```

```
                bcs         out                     ;(move out, not in)
                inc         curtrk                  ;incr current track (in).
                bcc         skin                    ;(always taken)
out             dec         curtrk                  ;decr current track (out).
skin            sec
step2           jsr         setphase
                lda         prior
                clc                                 ;for phaseoff
                jsr         clrphase                ;de-energize previous phase.
                bne         seek2                   ;(always taken)
seekend         equ         *
*
rdsect1         ldy         #$7f                    ;allow 127 mistakes.
                sty         rtrycnt
                php
*
tryread         plp                                 ;fix stack.
                skp         1
rdhead          sec                                 ;anticipate error.
                dec         rtrycnt                 ;if = 0 then give up!
                beq         erretrn                 ;branch if can't fine/read sector.
                clc                                 ;indicate reading header.
rddata          php                                 ;carry set if reading sector.
rd0             dey                                 ;every time y=0 decrement find count.
                beq         tryread
                skp         1
zzstart         equ         *
* from zzstart to zzend code is moved from
* rom and modified to match this code...
                skp         1
rd1             lda         q6l,x                   ;read a byet from the state machine.
                bpl         rd1                     ;loop until ready.
                dsect
                org         zzstart+5               ;equivalent to org *
rd1a            eor         #$d5                    ;mark 1?
mod1            equ         *-zzstart+1
                bne         rd0                     ;branch if not.
chg1            equ         rd0-*
rd2             lda         q6l,x
                bpl         rd2
                cmp         #$aa                    ;mark 2?
                bne         rd1a
                nop                                 ;waste a little time.
rd3             lda         q6l,x
                bpl         rd3
                cmp         #$96                    ;header mark 3?
                beq         rdhd1                   ;branch if it is.
                plp                                 ;were we looking for data mark 3?
mod2            equ         *-zzstart+1
                bcc         rdhead                  ;branch if not.
chg2            equ         rdhead-*
                eor         #$ad                    ;data mark 3?
                beq         rddt1                   ;go read data field if true...
mod3            equ         *-zzstart+1
rdhd0           bne         rdhead                  ;otherwise, start over.
chg3            equ         rdhead-*
rdhd1           ldy         #3                      ;read in trk,sect,&volume #.
rdhd2           sta         trktmp                  ;save last result in temp
rdhd3           lda         q6l,x
                bpl         rdhd3
                rol         a
                sta         oddbits                 ;save odd bits (7,5,3,1)
rdhd4           lda         q6l,x
```

```
                bpl         rdhd4
                and         oddbits                 ;combine even and odd to form value.
                dey
                bne         rdhd2                   ;read in next pair.
                plp
                cmp         sector                  ;last byte formed is sector#
mod4            equ         *-zzstart+1
                bne         rdhead                  ;branch if target sector not found.
chg4            equ         rdhead-*
                skp         1
                lda         trktmp                  ;previous result is track #
                cmp         track                   ;is desired track found?
mod5            equ         *-zzstart+1
                bne         goseek                  ;re-seek if mismatch.
chg5a           equ         *
mod6            equ         *-zzstart+1
                bcs         rddata                  ;branch if proper track always.
chg6            equ         rddata-*
                skp         1
rddt1           ldy         #$56                    ;read 2 bit groupings first.
rddt1a          sty         oddbits
rddt2           ldy         q6l,x
                bpl         rddt2
                eor         dnib,y                  ;denibblize using table left from boot
rom.
                ldy         oddbits                 ;save in nbuf1
                dey
                sta         nbuf1,y
                bne         rddt1a                  ;loop until all 86 groups are read.
                skp         1
rddt3           sty         oddbits                 ;now count up for 6-bit groups.
rddt4           ldy         q6l,x
                bpl         rddt4
                eor         dnib,y
                ldy         oddbits                 ;save result to specified buffer.
                sta         (dbuf),y
                iny
                bne         rddt3                   ;loop for 256 bytes.
rdchk           ldy         q6l,x                   ;now verify checksum...
                bpl         rdchk
                eor         dnib,y                  ;must be equal...
mod7            equ         *-zzstart+1
                bne         rdhd0                   ;branch if error.
chg7            equ         rdhd0-*
                ldy         #0                      ;make y=0
nxttwo          ldx         #$56                    ;now combine 2-bit group with 6 bit
group
twobit          dex                                 ;all done with this group?
                bmi         nxttwo                  ;branch if so.
                lda         (dbuf),y
                lsr         nbuf1,x
                rol         a
                lsr         nbuf1,x
                rol         a
                sta         (dbuf),y
                iny
                bne         twobit
                skp         1
zzzend          equ         *
                ldx         slotz
                clc                                 ;indicate good read.
                rts
chg5            equ         *-chg5a
```

```
goseek          jmp          seek
                dend
                ds           $a00-zzstart-5,0
* the following is the apple /// sos boot loader.
                msb          off
                sbtl         "sos system boot 1.1"
*****************************************************************
*
* sos system boot
*
* the code resides on blocks 0 and 1 of every sos diskette.
* its job is to locate the file named 'sos.kernel' on the
* boot diskette (drive 1), load the entire file into memory
* and then transfer control to the second stage boot,
* (sos loader).
*
* this first stage boot is designed to have minimal knowledge
* of both the rom code and the operating system including
* its associated drivers.
*
* assumptions:
*
*   1.  screen is cleared and 40 column b&w mode is selected.
*
*   2.  blockio routine is in rom with the entry pt at $f479.
*
*   3.  hardware:  see equates
*
*   4.  sos directory format
*
*   5.  file 'sos.kernel' format
*
* potential problems:
*
*   1.  this code disregards the address/count information
*       affixed to the front of the sos loader module.
*
*   2.  if code grows beyond current size, the padding at end of the code
*       needs to be modified.  (the code currently resides in less than a
*       single block; thus two padding statements necessary to have total
*       be two blocks on diskette.)
*
*
*****************************************************************
                page
*
* hardware addresses
*
e.reg           equ          $ffdf
b.reg           equ          $ffef
kybdstrb        equ          $c010
*
* monitor data and addresses
*
ibcmd           equ          $87
ibbufp          equ          $85
blockio         equ          $f479
*
* zero page storage (z reg = $03)
*
zpage           equ          $e0
blknum          equ          zpage+0                  ; & 1
*
```

```
begin           equ         zpage+2                 ; & 3
end             equ         zpage+4                 ; & 5
blk.ctr         equ         zpage+6
temp            equ         zpage+7
*
sosldr          equ         zpage+8                 ; & 9
*
* equates
*
dirblk0         equ         $a400
entry0          equ         dirblk0+4               ; loc of first file entry in directory
entry.len       equ         entry0+$1f              ; loc of entry length in directory
storage         equ         0                       ; file's storage type
sapling         equ         $20                     ; storage type = tree index file w/one
index block
rootdir         equ         $f0                     ; storage type = root directory
nextdblk        equ         2                       ; loc of next directory block
*
k.xblk          equ         $c00                    ;start loc of sos.kernel's index block
xblk            equ         $11                     ; loc of index block address in file
entry
k.file          equ         $1e00                   ; start loc of sos.kernel file
k.label         equ         k.file+0                ; loc of label in file "sos.kernel"
k.hdr.cnt       equ         k.file+8                ;   "    header       "
****************************************************************
*
* sos system boot - entry point
*
****************************************************************
*
bootinfo        equ         *
asmbase         equ         *                       ;assembly base address
runbase         equ         $a000                   ;execution base address
                jmp         boot+runbase-asmbase
                asc         "sos boot  1.1 "  ; sos boot identification "stamp"
                page
****************************************************************
*
* local data storage
*
****************************************************************
*
namlen          dfb         10
name            asc         "sos.kernel     "
name2           asc         "sos krnl"
name2.len       equ         *-name2
*
* messages
*
msg             equ         *                       ; message table
*
msg0            asc         "i/o error"
msg0l           equ         *-msg0
xmsg0           dw          *-msg-1
*
msg1            asc         "file 'sos.kernel' not found"
msg1l           equ         *-msg1
xmsg1           dw          *-msg-1
*
msg2            asc         "invalid kernel file"
msg2l           equ         *-msg2
xmsg2           dw          *-msg-1
*
```

```
* -- these dw's get around tla's hatred of hibyte/lobyte stuff
*
xk.xblk         dw          k.xblk
xk.file         dw          k.file
xk.hdr.cnt      dw          k.hdr.cnt+6             ;includes 6-byte offset
xentry0         dw          entry0
                page
****************************************************************
*
* sos system boot - main code body
*
****************************************************************
*
* turn off interrupts & decimal mode
*
boot            sei
                cld
*
* set up environment register and init stack
*
                lda         #$77                    ;1mhz dsbl
*                              i/o enbl
*                              primary stack enbl
*                              reset/nmi enbl
*                              write prot. dsbl
*                              primary stack enbl
*                              rom1 enbl
*                              rom enbl
                sta         e.reg
                ldx         #$fb
                txs
                bit         kybdstrb                ; turns off kybd
                lda         #$40                    ; "rti" instruction
                sta         $ffca                   ; prevents reboot w/keyboard nmi
*
* find highest memory bank in system and set bank reg to it
* - max memsize = 256k.
*
                lda         #7
                sta         b.reg
                ldx         #0
boot005         dec         b.reg
                stx         $2000
                lda         $2000
                bne         boot005
*
* read in blocks 1 thru n (rest of boot and all of root dir.)
*
                lda         #1
                sta         blknum
                lda         #0
                sta         blknum+1
*
                lda         #0
                sta         ibbufp
                lda         #$a2
                sta         ibbufp+1
*
                jsr         read.blk+runbase-asmbase  ; rest of boot (block 1)
*
                inc         blknum                  ; first root directory block (block 2)
                lda         #0
                sta         blk.ctr
```

```
rd.dir          inc         ibbufp+1
                inc         ibbufp+1
                inc         blk.ctr
*
                jsr         read.blk+runbase-asmbase  ; root directory
*
                ldy         #nextdblk                 ; if nextdir field = 0 then done
                lda         (ibbufp),y
                sta         blknum
                iny
                lda         (ibbufp),y
                sta         blknum+1
                bne         rd.dir
                lda         blknum
                bne         rd.dir
                page
*
* search directory for file 'sos.kernel'
*
                lda         xentry0+runbase-asmbase   ;get lo byte of address
                sta         begin
                lda         xentry0+1+runbase-asmbase
                sta         begin+1
*
search          clc                                   ; end:=begin+512-entry.len
                lda         begin+1
                adc         #2
                sta         end+1
                sec
                lda         begin
                sbc         entry.len
                sta         end
                lda         end+1
                sbc         #0
                sta         end+1
*
srch020         ldy         #0                        ; does count match?
                lda         (begin),y
                and         #$f
                cmp         namlen+runbase-asmbase
                bne         srch040                   ; no match
*
                tay
srch030         lda         (begin),y                 ; do chars match?
                cmp         name-1+runbase-asmbase,y
                bne         srch040                   ; no match
                dey
                bne         srch030
*
                ldy         #storage                  ;test storage type
                lda         (begin),y                 ;must be sapling
                and         #$f0
                cmp         #sapling
                beq         match
                cmp         #rootdir                  ;skip if stg type=rootdir
                beq         srch040
*
                ldx         xmsg2+runbase-asmbase     ;err,invalid kernel file
                ldy         #msg2l
                jmp         prnt.msg+runbase-asmbase
*
srch040         clc
                lda         begin
```

```
                adc         entry.len
                sta         begin
                lda         begin+1
                adc         #0
                sta         begin+1
                lda         end
                cmp         begin                   ;is begin <=end?
                lda         end+1
                sbc         begin+1
                bcs         srch020                 ;yes,search next field in current block
*
                clc                                 ;begin :=end+entry.len
                lda         end
                adc         entry.len
                sta         begin
                lda         end+1
                adc         #0
                sta         begin+1
*
                dec         blk.ctr
                bne         search                  ;search the next dir block
*
                ldx         xmsg1+runbase-asmbase   ;err, can't find 'sos.kernel'
                ldy         #msg1l
                jmp         prnt.msg+runbase-asmbase
                page
*
* file entry 'sos.kernel' found
* read in its index block ($c00) and first data block ($1e00)
*
match           ldy         #xblk
                lda         (begin),y
                sta         blknum
                iny
                lda         (begin),y
                sta         blknum+1
                lda         xk.xblk+runbase-asmbase ;get lo byte of address
                sta         ibbufp
                lda         xk.xblk+1+runbase-asmbase ;get hi byte of address
                sta         ibbufp+1
                jsr         read.blk+runbase-asmbase  ; index block
*
                lda         xk.file+runbase-asmbase ;get lo byte of address
                sta         ibbufp
                lda         xk.file+1+runbase-asmbase
                sta         ibbufp+1
                lda         k.xblk
                sta         blknum
                lda         k.xblk+$100
                sta         blknum+1
                jsr         read.blk+runbase-asmbase  ; first data block
*
* check the label, should be 'sos krnl'
*
                ldx         #name2.len-1
chk010          lda         k.label,x
                cmp         name2+runbase-asmbase,x
                beq         chk020
                ldx         xmsg2+runbase-asmbase   ; err, invalid kernel file
                ldy         #msg2l
                jmp         prnt.msg+runbase-asmbase
chk020          dex
                bpl         chk010
```

```
*
* read in the rest of the data blocks in file "sos.kernel"
*
                lda          #0
                sta          temp
*
data010         inc          temp
                inc          ibbufp+1
                inc          ibbufp+1
*
                ldx          temp                       ; get block address of next data block
                lda          k.xblk,x
                sta          blknum
                lda          k.xblk+$100,x
                sta          blknum+1
*
                lda          blknum                     ; is next block address = 0 ?
                bne          data020
                lda          blknum+1
                beq          entry.a3                   ; yes, stop reading
*
data020         jsr          read.blk+runbase-asmbase   ; read data block
                jmp          data010+runbase-asmbase    ;  and repeat
*
* build sos loader entry point address
*
entry.a3        clc                                     ; sosldr:=k.hdr.cnt+(k.hdr.cnt)
                lda          xk.hdr.cnt+runbase-asmbase
                adc          k.hdr.cnt
                sta          sosldr
                lda          xk.hdr.cnt+1+runbase-asmbase
                adc          k.hdr.cnt+1
                sta          sosldr+1
*
* now jump to sos loader (secondary bootstrap)
*
                jmp          (sosldr)
*
****************************************************************
*
* finished !!
*
* state of registers:
*
* b reg = highest 32k bank
* e reg = $77
* z reg = $03
*
* file "sos.kernel":
*
* index block is at $c00...$fff
* data block 0 is at $2200..$23ff
* data block 1 is at $2400..$25ff
*  " "
* data block n "
*
****************************************************************
                page
****************************************************************
*
* read block routine
*
* input: blknum & ibbufp
```

```
*
*****************************************************************
*
read.blk        equ         *
                lda         #1
                sta         ibcmd
*
                lda         blknum
                ldx         blknum+1
                jsr         blockio
                bcs         rd.err
                rts                                 ; normal exit
*
rd.err          ldx         xmsg0+runbase-asmbase   ;err, i/o error
                ldy         #msg0l
                jmp         prnt.msg+runbase-asmbase
                page
*****************************************************************
*
* print message
*
* input: msg index  (x)
*  msg length (y)
*****************************************************************
msgline         equ         $5a8                    ; prnt.msg routine
*
prnt.msg        equ         *
                sty         temp                    ; center msg (y:=40-len/2+len)
                sec
                lda         #40
                sbc         temp
                lsr         a
                clc
                adc         temp
                tay
*
prnt010         lda         msg+runbase-asmbase,x
                sta         msgline-1,y
                dex
                dey
                dec         temp
                bne         prnt010
*
                lda         $c040                   ; sound bell
                jmp         *+runbase-asmbase       ; hang until reboot (ctrl/reset)
*
*************************************************************
*
* padding to end of two blocks. modify if code length increases
* beyond one block.
*
*************************************************************
*
pad             equ         *-asmbase
                ds          512-pad,0               ;pad to end of block
zzend           equ         *
*<esc>e

; ##############################################################################################
; #   END OF FILE:  BOOT.SRC
; #   LINES      :  859
; #   CHARACTERS :  36129
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
```

```
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.CLOSE.EOF.pretty
================================================================================

; ###############################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  CLOSE.EOF
; ###############################################################################

               page
close          ldy          #c.refnum               ;close all?
               lda          (par),y
               bne          close1                  ;no, just one of 'em
               sta          cferr                   ; clear global close error
               lda          #0                      ;begin at the beginning.
clsall1        sta          fcbptr                  ;save current low byte of pointer
               tay                                  ; fetch the level at which
               lda          fcb+fcblevl,y           ; file was opened
               cmp          level                   ; test against current global level
               bcc          nxtclos                 ; dont close if files level is < global
level
               lda          fcb+fcbrefn,y           ;is this reference file open?
               beq          nxtclos                 ;no, try next.
               jsr          flush2                  ;clean it out...
               bcs          closerr                 ; return flush errors
               jsr          close2                  ;update fcb & vcb
               ldy          #c.refnum
               lda          (par),y
               beq          nxtclos                 ; no err if close all
               bcs          closerr
nxtclos        lda          fcbptr                  ;bump pointer to next file control
block.
               clc
               adc          #$20
               bcc          clsall1                 ;branch if within same page.
               lda          cferr                   ; on final close of close all report
logged errors
               beq          closend                 ; branch if errors
               rts                                  ;carry already set (see bcc)
*
close1         jsr          flush1                  ;flush file first (including updating
bit map)
               bcs          closerr                 ;report errors immediately!!
close2         ldy          fcbptr
               lda          fcb+fcbfbuf,y           ;release file buffer
               jsr          relbuffr
               bcs          closerr
               lda          #0
               ldy          fcbptr
               sta          fcb+fcbrefn,y           ;free file control block too.,
               lda          fcb+fcbdevn,y
               sta          devnum                  ;go look for associated vcb.
               jsr          fnddvcb
               ldx          vcbptr                  ;get vcbptr
               dec          vcb+vcbopnc,x           ;indicate one less file open.
               bne          closend                 ;branch if that wasn't the last...
               lda          vcb+vcbstat,x
               and          #$7f                    ;strip 'files open' bit
               sta          vcb+vcbstat,x
closend        clc
               rts
closerr        bcs          flsherr                 ; don't report closall err now
```

```
               page
flush          ldy          #c.refnum                ;flush all?
               lda          (par),y
               bne          flush1                   ;no, just one of 'em
               sta          cferr                    ; clear global flush error
               lda          #0                       ;begin at the beginning.
flshal1        sta          fcbptr                   ;save current low byte of pointer
               tay                                   ;index to reference number
               lda          fcb+fcbrefn,y            ;is this reference file open?
               beq          nxflush                  ;no, try next.
               jsr          flush2                   ;clean it out...
               bcs          flsherr                  ;return any errors
*
nxflush        lda          fcbptr                   ;bump pointer to next file control
block.
               clc
               adc          #$20
               bcc          flshal1                  ;branch if within same page.
flushend       clc
               lda          cferr                    ; on last flush of a flush(0)
               beq          f3                       ; branch if no logged errors
               sec                                   ; report error now
f3             rts
*
flush2         jsr          fndfcbuf                 ;must set up associated vcb an buffer
locations first.
               bcc          flush2a                  ;branch if no error encountered.
flsherr        jmp          glberr                   ; check for close or flush all
*
flush1         lda          #0                       ; clear
               sta          cferr                    ; global error for normal refnum flush
               jsr          findfcb                  ;set up pointer to fcb user references
               bcs          flsherr                  ;return any errors
flush2a        equ          *                        ;test to see if file is
               lda          fcb+fcbattr,y            ;modified. first test write enabled.
               and          #writen
               beq          flushend                 ;branch if 'read only'
               lda          fcb+fcbdirty,y           ; see if eof has been modified
               bmi          flush2b                  ; branch if it has
               jsr          gfcbstat                 ;now test for data modified.
               and          #usemod+eofmod+datmod    ; was written to while it's been open?
               beq          flushend                 ;branch if file not modified.
flush2b        jsr          gfcbstat                 ;now test for data modified.
               and          #datmod                  ;does current data buffer need to be
               beq          flush3                   ; written? branch if not.
               jsr          wfcbdat                  ;if so, go write it stupid!
               bcs          flsherr
flush3         jsr          gfcbstat                 ;check to see if the index block (tree
files only)
               and          #idxmod                  ; needs to be written.
               beq          flush4                   ;branch if not...
               jsr          wfcbidx
               bcs          flsherr                  ;return any errors.
               page
*
flush4         lda          #fcbentn                 ;now prepare to update directory
               tax
               ora          fcbptr
               tay
ownrmov        lda          fcb,y                    ;note: this code depends on the
               sta          d.dev-1,x                ; defined order of the file control
               dey                                   ; block and the temporary directory
area in 'workspc'! *************
```

```
                dex
                bne         ownrmov
                sta         devnum
                lda         d.head                  ;read the directory header for this
file
                ldx         d.head+1
                jsr         rdblk                   ;read it into the general purpose
buffer
                bcs         flsherr                 ;branch if error.
                jsr         movhed0                 ;move header info.
                lda         d.entblk                ;get address of directory block that
                ldy         d.entblk+1              ; contains the file entry.
                cmp         d.head                  ;test to see if it's the same block
that
                bne         flsheblk                ; the header is in. branch if not.
                cpy         d.head+1
                beq         flush5                  ;branch if header block = entry block.
flsheblk        sta         bloknml
                sty         bloknmh
                jsr         rdgbuf                  ;get block with file entry in general
buffer.
flush5          jsr         entcalc                 ;set up pointer to entry
                jsr         moventry                ;move entry to temp entry buffer in
'workspc'
                ldy         fcbptr                  ;update 'blocks used' count.
                lda         fcb+fcbuse,y
                sta         dfil+d.usage
                lda         fcb+fcbuse+1,y
                sta         dfil+d.usage+1          ;hi byte too...
                ldx         #0                      ;and move in end of file mark whether
eofupdte        lda         fcb+fcbeof,y            ; we need to or not.
                sta         dfil+d.eof,x
                inx                                 ;move all three bytes.
                cpx         #3
                beq         eofup1
                lda         fcb+fcbfrst,y           ;also move in the address of
                sta         dfil+d.frst-1,x         ; the file's first block since
                iny                                 ; it might have changed since the file
                bne         eofupdte                ; first opened. branch always taken.
*
                page
eofup1          lda         fcb+fcbstyp-2,y         ;the last thing to update is storage
type (y=fcbptr+2).
                asl         a                       ;(shift it into the hi nibble)
                asl         a
                asl         a
                asl         a
                sta         scrtch
                lda         dfil+d.stor             ;get old type byte (it might be the
same)
                and         #$f                     ;strip off old type
                ora         scrtch                  ;add in the new type,
                sta         dfil+d.stor             ;and put it away.
                jsr         drevise                 ;go update directory!
                bcs         flusherr
                ldy         fcbptr                  ; mark
                lda         fcb+fcbdirty,y          ; fcb/directory
                and         #$ff-fcbmod             ; as
                sta         fcb+fcbdirty,y          ; undirty
                lda         d.dev                   ; see if bitmap should be written.
                cmp         bmadev                  ;is it in same as current file?
                bne         flshend1                ;yes, put it on the disk if necessary.
                jsr         upbmap                  ;go put it away.
```

```
                bcs         flusherr
flshend1        clc
                rts
flusherr        equ         *                       ; drop into glberr
*
glberr          equ         *                       ; report error immediately
* only if not a close all or flush all
                ldy         #c.refnum
                pha
                lda         (par),y
                bne         glberr1                 ; not an 'all' so report now
                clc
                pla
                sta         cferr                   ; save for later
                rts
glberr1         pla
                rts
*
gfcbstat        ldy         fcbptr                  ;index to fcb.
                lda         fcb+fcbstat,y           ;return status byte.
                rts                                 ;that is all...
*
seterr          lda         #accserr
                sec
eofretn         rts
                page
seteof          jsr         gfcbstyp                ;only know how to move eof of tree,
sapling, or seed.
                cmp         #tretyp+1
                bcs         seterr                  ;branch if other than tree
                asl         a
                asl         a
                asl         a
                asl         a
                sta         stortype                ;may be used later for trimming the
tree...
                lda         fcb+fcbattr,y           ;now check to insure write is enabled.
                and         #writen                 ;can we set new eof?
                beq         seterr                  ;nope, access error.
                jsr         tstwprot                ;find out if mod is posible (hardware
write protect)
                bcs         seterr
                ldy         fcbptr                  ; save old eof
                iny
                iny
                ldx         #2                      ; so it can be seen
setsave         lda         fcb+fcbeof,y            ; whether blocks need
                sta         oldeof,x                ; to be released
                dey                                 ; upon
                dex                                 ; contraction
                bpl         setsave                 ; all three bytes of the eof
                ldy         #c.eof+2
                ldx         #2
neofpos         lda         (par),y                 ;position mark to new eof
                sta         tposll,x
                dey
                dex
                bpl         neofpos
                ldx         #2                      ; point to third byte
purtest         lda         oldeof,x                ; see if eof moved backwards
                cmp         tposll,x                ; so blocks can
                bcc         eofset                  ; be released (branch if not)
                bne         purge                   ; branch if blocks to be released
```

```
                dex
                bpl         purtest                 ; all three bytes
eofset          ldy         #c.eof+2
                ldx         fcbptr                  ;place new end of file into fcb.
                inx
                inx
eofset1         lda         (par),y
                sta         fcb+fcbeof,x
                dex
                dey
                cpy         #c.eof                  ;all three bytes moved?
                bcs         eofset1                 ;branch if not...
                jmp         fcbused                 ;mark fcb as dirty... all done.
*
purge           jsr         flush1                  ;make sure file is current.
                bcs         eofretn
                ldx         datptr+1                ;restore pointer to index block
                inx
                inx                                 ;(zero page conflict with dirptr)
                stx         tindx+1
                ldx         datptr
                stx         tindx
                ldy         fcbptr                  ;find out if eof < mark.
                iny
                iny
                ldx         #2
neoftst         lda         fcb+fcbmark,y
                cmp         tposll,x                ;compare until not equal or carry clear
                bcc         seteof1                 ;branch if eof>mark
                bne         seteof0                 ;branch if eof<mark
                dey
                dex
                bpl         neoftst                 ;loop on all three bytes
seteof0         ldy         fcbptr
                ldx         #0
fakeof          lda         tposll,x                ;fake position, correct position
                sta         fcb+fcbmark,y
                iny                                 ; will be made below...
                inx                                 ;move all three bytes
                cpx         #3
                bne         fakeof
*
*
seteof1         jsr         tkfrecnt                ; force proper free block count before
releasing blocks
                lda         tposll                  ;now prepare for purge of excess
blocks..
                sta         dseed                   ;all blocks and bytes beyond new eof
                lda         tposlh                  ; must be zeroed!
                sta         dsap
                and         #1
                sta         dseed+1
                lda         tposhi
                lsr         a
                sta         dtree
                ror         dsap                    ;pass position in terms of block &
bytes
                lda         dseed                   ;now adjust for boundarys of $200
                ora         dseed+1
                bne         seteof3                 ;branch if no adjustment necessary
                lda         dsap                    ;get correct block positions for sap
and
                sec                                 ; tree levels.
```

```
                sbc         #1
                sta         dsap                    ;deallocate for last (phantom) block.
                lda         #2                      ;and don't modify last data block.
                bcs         seteof2                 ;branch if tree level un affected.
                dec         dtree                   ;but if it is affected, make sure
neweof#0
                bpl         seteof2                 ;branch if new eof not zero
                lda         #0                      ;otherwise, just make a nul seed out of
it.
                sta         dtree
                sta         dsap
seteof2         sta         dseed+1
seteof3         ldy         fcbptr                  ;also must pass file's first block
addr.
                lda         fcb+fcbfrst,y
                sta         firstbl
                lda         fcb+fcbfrst+1,y
                sta         firstbh
                lda         #0                      ;lastly, number of blocks to be
                sta         deblock                 ; freed should be initialized.
                sta         deblock+1
                jsr         detree                  ;go defoliate...
                php                                 ;save any error status until
                pha                                 ;fcb is cleaned up!
                sec
                ldy         fcbptr
                ldx         #0
adjfcb          lda         firstbl,x
                sta         fcb+fcbfrst,y           ;move in posible new first file block
addr.
                lda         fcb+fcbuse,y            ;adjust usage count also
                sbc         deblock,x
                sta         fcb+fcbuse,y
                iny
                inx
                txa
                and         #1                      ;test for both bytes adjusted
                bne         adjfcb                  ; without disturbing carry.
                lda         stortype                ;get posibly modified storage type
                lsr         a
                lsr         a
                lsr         a
                lsr         a
                ldy         fcbptr                  ;save it in fcb
                sta         fcb+fcbstyp,y
                jsr         clrstats                ;make it look as though position has
                jsr         dvcbrev                 ; nothing allocated, update total
blocks in vcb.
                ldy         fcbptr                  ;now correct position stuff.
                iny
                iny
                ldx         #2
corctpos        lda         fcb+fcbmark,y           ;tell rdposn to go to correct
                sta         tposll,x
                eor         #$80                    ; position from incorrect place.
                sta         fcb+fcbmark,y
                dey
                dex
                bpl         corctpos
                jsr         rdposn                  ;go do it!!!
                bcc         purge1                  ;branch if no error.
                tax                                 ;otherwise report latest error.
                pla
```

```
                plp
                txa                                     ;restore latest error code to stack
                sec
                php
                pha                                     ;save new error.
purge1          equ             *                       ;mark file as in need of a flush and
                jsr             eofset                  ;update fcb with new end of file
                jsr             flush1                  ;now go do flush
                bcc             purge2                  ;branch if no error.
                tax                                     ;save latest error.
                pla                                     ;clean previous error off stack
                plp
                txa                                     ;restore latest error code to stack
                sec                                     ;set the carry to show error condition
                php                                     ;restore error status to stack
                pha                                     ;and the error code.
purge2          equ             *
                pla                                     ;report any errors that may have
cropped up.
                plp
                rts
*
*
geteof          ldx             fcbptr                  ;index to end of file mark
                ldy             #c.eof                  ;and index to users call parameters
outeof          lda             fcb+fcbeof,x
                sta             (par),y
                inx
                iny
                cpy             #c.eof+3
                bne             outeof                  ;loop until all three bytes are moved.
                clc                                     ;no errors
                rts
*

; ###############################################################################################
; #   END OF FILE:  CLOSE.EOF
; #   LINES      :  374
; #   CHARACTERS :  21464
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.COMMANDS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  COMMANDS
; ###############################################################################################

; $40-47 are already used by prodos so we can use them
serr            equ         $bf0f
zpsize          equ         8
                ifeq        opers-prodos
cmndlist        equ         $40
                else
cmndlist        equ         0
                fin
execadr         equ         cmndlist+2
*
************** see rev note #en3 ***************
*
* the following address table contains entries into the atp, pap
* fap etc. drivers.  there is room for 16 entry addresses.
*
                ifne        ednet
start.entries   equ         *
                dw          atalkprg                ; entry (0) to atp drivers.
                dw          $e500                   ; entry (1) to pap drivers.
                dw          $e000                   ; entry (2) to rpm driver.
                dw          $0000                   ; entry (3) to fap drivers.
                dw          inthandler              ; entry (4) to atp interrupt handler.
                dw          $0000                   ; entry (5) to rpm close routine.
                ds          start.entries+32-*      ; remaining space in entry table.
                fin
*
*****************************************************
*
                ifeq        opers-prodos+ednet
;
; entry point from mli
;
* note that the following code that increments the parameter list
* pointer should not be assembled for the ednet version of prodos.
* this is done by prodos itself, so it can then use x/y entry point.
*
frommli         equ         *
                inc         cmndlist
                bne         fmli.1
                inc         cmndlist+1
fmli.1          jsr         atlkprg2
                sta         serr
fmli.9          rts
;
                fin
;
                ifeq        opers-prodos
atalkprg        equ         *
                stx         cmndlist
                sty         cmndlist+1
                else                                ; pascal and kernel saves zero page
atalkprg        equ         *
                cld
```

```
; drcallct starts as $ff, first time it is called it becomes $00
; second times it becomes $01
                txa
                ldx         intrflag                ; are we in an interrupt
                bne         atlkprg1                ; yes, no need to save zero page
                ldx         indriver                ; did we save it before
                bne         atlkprg1                ; then no need to save zero page
                pha
                ldx         #zpsize-1               ; save zero page
                lda         cmndlist,x
                sta         savezp1,x
                dex
                bpl         *-6
                pla
atlkprg1        sta         cmndlist
                sty         cmndlist+1
                inc         indriver                ; mark it as in driver already
                fin
atlkprg2        ldy         #0                      ; point to command
                sty         status                  ; clear error flag
                lda         (cmndlist),y
                and         #$7f                    ; ignore high bit
                cmp         #maxcmnd
                bcs         bdcmnd
                bit         openflag                ; are we open already
                bmi         cmndok
                cmp         #1                      ; is it open command
                bne         ntopen                  ; if not, driver not open error
cmndok          asl         a                       ; * 2 for displacement into table
                tax
                lda         cmndtable,x
                sta         execadr
                lda         cmndtable+1,x
                sta         execadr+1
                iny
                lda         (cmndlist),y            ; prefetch first operand
                jsr         doexecute               ; execute the routine
                ifne        opers-prodos
cmnddone        dec         indriver                ; now we are existing from driver
                bne         atlkprg8                ; no need to restore if we are still in
it
                lda         intrflag                ; do we need to restore zero page
                bne         atlkprg8                ; not if we call from interrupt
                ldx         #zpsize-1
                lda         savezp1,x
                sta         cmndlist,x
                dex
                bpl         *-6
atlkprg8        equ         *
                fin
                lda         status                  ; was everything ok ?
                bne         err.rts
                clc                                 ; indicates ok
                rts
                ifeq        opers-prodos
bdcmnd          lda         #badcmnd
err.rts         sec                                 ; indicates error
                rts
ntopen          lda         #nodriver
                sec
                rts
                else
err.rts         sec
```

```
                rts
bdcmnd          lda         #badcmnd
                bne         ntopen+2
ntopen          lda         #nodriver
                sta         status
                bne         cmnddone                ; bra
                fin
;
openflag        db          0
status          db          0
                ifne        opers-prodos
savezp1         ds          zpsize,0
intrflag        dfb         0
indriver        dfb         0
                fin
;
toomanycollison equ         $60+8
badcmnd         equ         $60+11
nodriver        equ         $60+12
alreadyopen     equ         $60+13
nointcard       equ         $60+14
illegalvalue    equ         $60+15
badsocket       equ         $60+16
socketopen      equ         $60+17
scktnotopen     equ         $60+18
toomanyreq      equ         $60+19
bufferfull      equ         $60+20
notconfirmed    equ         $60+21
diffsocket      equ         $60+22
requestfail     equ         $60+23
reqabort        equ         $60+24
;
maxcmnd         equ         18+1
cmndtable       dw          bdcmnd
                dw          opendriver
                dw          closedriver
                do          nolap
                dw          bdcmnd
                else
                dw          writelap
                fin
                dw          sethook
                dw          openddpsckt
                dw          writeddp
                dw          closeddpsckt
                dw          lookupname
                dw          confirmname
                dw          bdcmnd
                dw          bdcmnd
                dw          sndatprq
                dw          openatpsckt
                dw          closeddpsckt
                dw          agetrequest
                dw          atpsndresponse
                dw          killrspcb
                dw          cancelatpreq
;
;
;
applebus        lda         cardslot
                jmp         $c712
; the next 4 bytes (6 for prodos) are to be copied back to opendriver parm
; and therefore must be group together in the exact order
```

```
; note the cardslot is also part of the routine applebus
cardslot        equ         applebus+5
dataarea        equ         *-1                     ; so that all displacement is non-zero
ournode         dfb         0
                dw          inthandler
                ifeq        opers-prodos
                dw          atalkprg
                fin
;
const0          dfb         0
const1          dfb         1
const2          dfb         2
const5          dfb         5
const8          dfb         8
const13         dfb         13
const16         dfb         16
chksmf          db          0
curcksum        dw          0
bridgetime      dw          0
wddpparm        db          3
wddptbl         dw          0
wddpsize        dw          0,lapdata
version         dfb         $ff
cardversion     dfb         0
thisnet         dw          0
                dfb         0                       ; this should contains value of 8 to
indicates 8 bits
a.bridge        db          0
outnbp          dfb         $21                     ; lkup with 1 tuple
outnbpid        dfb         0
nbpbuffer       ds          32,0
lapdata         equ         nbpbuffer+5
lapdest         equ         lapdata+0
lapsrce         equ         lapdata+1
laptype         equ         lapdata+2
ddphead         equ         lapdata+3
sd.length       equ         0                       ; use for parameter passing
sd.dsckt        equ         2                       ; use for parameter passing
sd.ssckt        equ         3                       ; use for parameter passing
sd.protocol     equ         4
ld.length       equ         0
ld.chksum       equ         2
ld.dnet         equ         4                       ; use for parameter passing
ld.snet         equ         6
ld.dnode        equ         8
ld.snode        equ         9
ld.dsckt        equ         10
ld.ssckt        equ         11
ld.protocol     equ         12                      ; use for parameter passing
atpheader       equ         ddphead+ld.protocol+1

; ###########################################################################################
; #   END OF FILE:  COMMANDS
; #   LINES      :  227
; #   CHARACTERS :  9098
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.CREATE.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  CREATE
; ###############################################################################################

               page
create         jsr         lookfile                ;check for duplicate / get free entry
               bcs         tstfnf                  ;error code in acc may be 'file not
found'
               lda         #duperr                 ;tell em a file of that name already
exists
crerr1         sec                                 ;indicate error encountered
               rts                                 ;return error in acc.
*
tstfnf         cmp         #fnferr                 ;'file not found' is what we want
               bne         crerr1                  ;pass back other error.
               ldy         #c.fkind                ;test for "tree" or directory file.
               lda         (par),y                 ;no other kinds are legal.
               cmp         #$4                     ;is it seed, sapling, or tree?
               bcc         tstdspc                 ;branch if it is.
               cmp         #$d
               bne         ctyperr                 ;report type error if not directory.
tstdspc        lda         devnum                  ;before proceeding, make sure
destination device
               jsr         twrprot1                ; is not write protected...
               bcs         crtn
               lda         nofree                  ;is there space in directory to add
this file?
               beq         xtndir                  ;branch if not.
               jmp         creat1                  ;otherwise, go create file.
*
ctyperr        lda         #typerr
               sec                                 ;indicate error
crtn           rts
*
xtndir         lda         own.blok                ;before extending directory,
               ora         own.blok+1              ; make sure it is a sub directory!!!
               bne         xtndir1
               lda         #dirfull                ;otherwise report directory full error.
               sec
               rts
*
xtndir1        lda         bloknml                 ;preserve disk address of current
(last)
               pha
               lda         bloknmh                 ; directory link, before allocating an
               pha                                 ; extend block.
               jsr         alc1blk                 ;allocate a block for extending
directory.
               tax                                 ;save acc for now.
               pla
               sta         bloknmh                 ;restore block addr of directory stuff
in gbuf.
               pla
               sta         bloknml
               txa                                 ;restore acc.
               bcs         crtn                    ;branch if unable to allocate.
```

```
                sta     gbuf+2                  ;save low block address in current
directory
                sty     gbuf+3                  ;and hi adr too
                jsr     wrtgbuf                 ;go update dir. block with new link.
                bcs     crtn                    ;(report any errors.)
                ldx     #1
swpbloks        lda     bloknml,x               ;now prepare new directory block.
                sta     gbuf,x                  ;use current block as back link.
                lda     gbuf+2,x
                sta     bloknml,x               ;and save new block as next to be
writen.
                dex
                bpl     swpbloks
                inx                             ;now x=0
                txa                             ; and a=0 too.
clrdir          sta     gbuf+2,x
                sta     gbuf+$100,x
                inx
                bne     clrdir
                jsr     wrtgbuf                 ;write prepared directory extention.
                bcs     crtn                    ;report errors.
*
                lda     own.blok
                ldx     own.blok+1
                jsr     rdblk                   ;read in 'parent' directory block.
                ldx     own.ent                 ;prepare to calculate entry address.
                lda     #gbuf/256
                sta     drbufph
                lda     #4
ocalc           clc
                dex                             ;has entry adr. been computed
                beq     ocalc1                  ;branch if yes.
                adc     own.len                 ;bump to next entry adr.
                bcc     ocalc
                inc     drbufph                 ;entry must be in second 256 of block.
                bcs     ocalc                   ;branc always.
ocalc1          sta     drbufpl
                ldy     #d.usage                ;index to block count.
ocalc2          lda     (drbufpl),y
                adc     dinctbl-d.usage,y       ;add 1 to block count and
                sta     (drbufpl),y
                iny
                tya                             ; $200 to the directory's end of file.
                eor     #d.eof+3                ;done with usage/eof update?
                bne     ocalc2                  ;branch if not.
                jsr     wrtgbuf                 ;go update parent.
                bcs     crerr2
                jmp     create
*
*
zgbuf           lda     #0                      ;zero out gbuf.
                tax
clrgbuf         sta     gbuf,x
                sta     gbuf+$100,x             ;first zero out data block of file.
                inx
                bne     clrgbuf                 ;loop until zipped!
crerr2          rts                             ;report errors...
*
creat1          equ     *
                jsr     zgbuf                   ;zero out gbuf.
                ldy     #c.time+1               ;move user specified date/time
cmvtime         lda     (par),y                 ; to directory entry.
                sta     dfil+d.credt-c.date,y
```

```
                txa                                     ;if all four bytes of date/time are
zero
                ora         (par),y                     ; then use built in date/time.
                tax
                dey                                     ;have all four bytes been moved and
tested?
                cpy         #c.fkind
                bne         cmvtime                     ;branch if not.
                txa                                     ;does user want default time?
                bne         cmvname                     ;branch if not.
                ldx         #3
mvdftime        lda         datelo,x                    ;move current default date/time.
                sta         dfil+d.credt,x
                dex
                bpl         mvdftime
*
cmvname         lda         (par),y                     ;(y is pointing at fkind.)
                cmp         #4
                lda         #$10                        ;assume tree type.
                bcc         csvfkind
                lda         #$d0                        ;it is directory since file kind has
already been verified.
csvfkind        ldx         namptr                      ;get index to 'local' name of pathname.
                ora         pathbuf,x                   ;combine file kind with name length.
                sta         dfil+d.stor                 ;(sos calls this 'storage type')
                and         #$f                         ;strip back to name length.
                tay                                     ;and use as count-down for move.
                clc
                adc         namptr                      ;calculate end of name.
                tax
*
crname          lda         pathbuf,x                   ;now move local name as filename.
                sta         dfil+d.stor,y
                dex
                dey                                     ;all characters transfered?
                bne         crname                      ;branch if not.
*
                ldy         #c.attr                     ;index to 'access' parameter.
                lda         (par),y
                sta         dfil+d.attr
                iny                                     ;also move 'file identification'
                lda         (par),y
                sta         dfil+d.filid
cmvauxid        iny                                     ; and finally, the auxillary
identifcation bytes.
                lda         (par),y
                sta         dfil+d.auxid-c.auxid,y
                cpy         #c.auxid+1
                bne         cmvauxid
                lda         xdosver                     ;save current xdos version number.
                sta         dfil+d.sosver
                lda         compat                      ;and backward compatiblity number.
                sta         dfil+d.comp
                lda         #1                          ;usage is always 1 block.
                sta         dfil+d.usage
                lda         d.head                      ;place back pointer to header block
                sta         dfil+d.dhdr
                lda         d.head+1
                sta         dfil+d.dhdr+1
                lda         dfil+d.stor                 ;get storage type again.
                and         #$e0                        ;is it a directory
                beq         cralcblk                    ;branch if seed file.
                ldx         #$1e                        ;move header to data block.
```

```
cmvheadr        lda         dfil+d.stor,x
                sta         gbuf+4,x
                dex
                bpl         cmvheadr
                eor         #$30                ;($dn->$en) last one is fkind/namlen,
                sta         gbuf+4              ; make it a directory header mark
                ldx         #7                  ;now overwrite password area and other
header info.
cmvpass         lda         pass,x
                sta         gbuf+4+$10,x
                lda         xdosver,x
                sta         gbuf+4+$1c,x
                dex
                bpl         cmvpass
*
                ldx         #2                  ;and include info about 'parent
directory.
                stx         dfil+d.eof+1
cmvparnt        lda         d.entblk,x
                sta         gbuf+4+$23,x
                dex
                bpl         cmvparnt
                lda         h.entln             ;lastly the length of parent's dir
entries
                sta         gbuf+4+$26
*
cralcblk        jsr         alc1blk             ;get address of file's data block.
                bcs         crerr3              ;branch if error encountered.
                sta         dfil+d.frst
                sty         dfil+d.frst+1
                sta         bloknml
                sty         bloknmh
                jsr         wrtgbuf             ;go write data block of file.
                bcs         crerr3
                inc         h.fcnt              ;add 1 to total # of files in this
directory
                bne         credone
                inc         h.fcnt+1
credone         jsr         drevise             ;go revise directories with new file.
                bcs         crerr3
                jmp         upbmap              ;lastly, update volume bitmap.
*
entcalc         lda         #gbuf/256           ;set high address of directory entry
index pointer
                sta         drbufph
                lda         #4                  ;calculate address of entry based
                ldx         d.entnum            ; on the entry number
ecalc0          clc
ecalc1          dex                             ;addr=gbuf+((entnum-1)*entlen)
                beq         ecalc2
                adc         h.entln
                bcc         ecalc1
                inc         drbufph             ;bump hi address
                bcs         ecalc0              ;branch always.
*
ecalc2          sta         drbufpl             ;save newly calculated low address
crerr3          equ         *                   ;return errors
derror2         rts
                page
drevise         lda         datelo              ; if no clock,
                beq         drevise1            ; then don't touch mod t/d
                ldx         #3                  ;move last modification date/time to
entry being updated.
```

```
modtime         lda         datelo,x
                sta         dfil+d.moddt,x
                dex
                bpl         modtime
*
drevise1        lda         dfil+d.attr             ; mark entry as backupable
                ora         bkbitflg                ; bit 5 = backup needed bit
                sta         dfil+d.attr
                lda         d.dev                   ;get device number of directory
                sta         devnum                  ; to be revised.
                lda         d.entblk                ;and address of directory block
                ldx         d.entblk+1
                jsr         rdblk                   ;read block into general purpose
buffer.
                bcs         crerr3
                jsr         entcalc                 ;fix up pointer to entry location
within gbuf.
                ldy         h.entln                 ;now move 'd.' stuff to directory.
                dey
mvdent          lda         dfil+d.stor,y
                sta         (drbufpl),y
                dey
                bpl         mvdent
                lda         d.head                  ;is the entry block the same as the
                cmp         bloknml                 ; entry's header block?
                bne         sventdir                ;no, save entry block
                lda         d.head+1                ;maybe, test high addresses
                cmp         bloknmh
                beq         uphead                  ;branch if they are the same block.
sventdir        jsr         wrtgbuf                 ;write updated directory block
                bcs         derror2                 ;return any error.
                lda         d.head                  ;get address of header block
                ldx         d.head+1
                jsr         rdblk                   ;read in header block for modification
                bcs         derror2
uphead          ldy         #1                      ;update current number of files in this
directory
uphed1          lda         h.fcnt,y
                sta         gbuf+hcent+4,y          ;(current entry count)
                dey
                bpl         uphed1
                lda         h.attr                  ;also update header's attributes.
                sta         gbuf+hattr+4
                jsr         wrtgbuf                 ;go write updated header
                bcs         derror1
                page
ripple          lda         gbuf+4                  ;test for 'root' directory
                and         #$f0                    ; if it is root, then directory
revision is complete.
                eor         #$f0                    ;(leaves carry clear)
                beq         drvisdne                ;branch if ripple done.
                lda         gbuf+hrent+4            ;get entry number
                sta         d.entnum
                lda         gbuf+hreln+4            ; and the length of entries in that dir
                sta         h.entln
                lda         gbuf+hrblk+4            ;get addr of parent entry's dir block.
                ldx         gbuf+hrblk+5
                jsr         rdblk                   ;read that sucker in.
                bcs         derror1
                jsr         entcalc                 ;get indirect pointer to parent entry
in gbuf
                lda         datelo                  ; don't touch mod
                beq         rupdate                 ; if no clock. . .
```

```
                ldx         #3                      ;now update the modification date and
time for this entry too.
                ldy         #d.moddt+3
riptime         lda         datelo,x
                sta         (drbufpl),y
                dey
                dex
                bpl         riptime                 ;move all for bytes...
rupdate         jsr         wrtgbuf                 ;write updated entry back to disk.
(assumes bloknm undisturbedd)
                bcs         derror1                 ;give up on any error.
                ldy         #d.dhdr                 ;now compare current block number to
this
                lda         (drbufpl),y             ; entry's header block
                iny
                cmp         bloknml                 ;are low addresses the same?
                sta         bloknml                 ;(save it in case it's not)
                bne         ripple2                 ;branch if entry does not reside in
same block as header.
                lda         (drbufpl),y             ;check high address just to be sure.
                cmp         bloknmh
                beq         ripple                  ;they are the same, continue ripple to
root directory.
ripple2         lda         (drbufpl),y             ;they aren't the same, read in this
directory's header.
                sta         bloknmh
                jsr         rdgbuf
                bcc         ripple                  ;continue if read was good.
derror1         equ         *
                rts
                page
*
tsterr          lda         #notsos                 ;not tree or directory- not a
recognized type!
                sec
                rts                                 ;do nothing.
*
*
tstsos          lda         gbuf                    ;test sos stamp
                cmp         sostmpl
                bne         tsterr
                lda         gbuf+1
                cmp         sostmph
                bne         tsterr
                lda         gbuf+4                  ;test for header
                and         #$e0
                cmp         #hedtyp*16
                bne         tsterr                  ;branch if not sos header (no error
number)
drvisdne        clc                                 ;indicate no error./
                rts
*

; ###############################################################################################
; #   END OF FILE:  CREATE
; #   LINES      :  324
; #   CHARACTERS :  18510
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.DATATBLS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DATATBLS
; ###############################################################################################

                page
* ---- added call $41 & its count - see rev note 20 --------
scnums          equ         *
                dfb         $d3,0,0,0                   ;(zeros are reserved for bfm)
                dfb         $40,$41,$00,0               ;(zero is reserved for interrupt calls)
                dfb         $80,$81,$82,$65
                dfb         $c0,$c1,$c2,$c3
                dfb         $c4,$c5,$c6,$c7
                dfb         $c8,$c9,$ca,$cb
                dfb         $cc,$cd,$ce,$cf
                dfb         $00,$d0,$d1,$d2             ;zero is non-existant.
*
pcntbl          equ         *                           ;parameter counts for the calls
                dfb         2,$ff,$ff,$ff
                dfb         2,1,$ff,$ff
                dfb         3,3,0,4
                dfb         7,1,2,7
                dfb         $a,2,1,1
                dfb         3,3,4,4
                dfb         1,1,2,2
                dfb         $ff,2,2,2
*
                page
*
************** see rev note #en2 *****************
*
                ifeq        os-ednet
atcmdtable      equ         *
                dw          atcreate
                dw          atdestroy
                dw          atrename
                dw          atsetinfo
                dw          atgetinfo
                dw          atonline
                dw          atsetprefx
                dw          atgetprefx
                dw          atopen
                dw          atnewline
                dw          atread
                dw          atwrite
                dw          atclose
                dw          atflush
                dw          atsetmark
                dw          atgetmark
                dw          atseteof
                dw          atgeteof
                dw          atsetbuf
                dw          atgetbuf
                fin
*
**********************************
*
cmdtable        equ         *
```

```
                dw          create
                dw          destroy
                dw          rename
                dw          setinfo
                dw          getinfo
                dw          online
                dw          setprefx
                dw          getprefx
                dw          open
                dw          newline
                dw          read
                dw          write
                dw          close
                dw          flush
                dw          setmark
                dw          getmark
                dw          seteof
                dw          geteof
                dw          setbuf
                dw          getbuf
*
disptch         equ         *
                dfb         prepath+pretime+0       ;create
                dfb         prepath+pretime+1       ;destroy
                dfb         prepath+pretime+2       ;rename
                dfb         prepath+pretime+3       ;setinfo
                dfb         prepath+4               ;getinfo
                dfb         5                       ;volume
                dfb         6                       ;setprefix, pathname moved to prefix
buffer
                dfb         7                       ;getprefix
                dfb         prepath+8               ;open
                dfb         preref+$9               ;newline
                dfb         preref+$a               ;read
                dfb         preref+$b               ;write
                dfb         pretime+$c              ;close
                dfb         pretime+$d              ;flush, refnum may be zero to flush
all.
                dfb         preref+$e               ;setmark
                dfb         preref+$f               ;getmark
                dfb         preref+$10              ;set eof
                dfb         preref+$11              ;get eof
                dfb         preref+$12              ;set buffer address (move)
                dfb         preref+$13              ;get buffer address
*
                page
dinctbl         dfb         1,0,0,2,0               ;table to increment directory usage/eof
counts.
*
*
pass            dfb         $75
;asc "huston!"
*
*
sostmpl         dfb         0
sostmph         dfb         0
*
xdosver         dfb         $0,0,$c3,$27,$d,0,0,0
compat          equ         xdosver+1
*
rootstuf        dfb         $f,2,0,4
                dfb         0,0,8,0
*
```

```
*
whichbit        dfb          $80,$40,$20,$10
                dfb          8,4,2,1
*
*
* the following table is used in the 'open5' loop (posn/open).
*
ofcbtbl         dfb          fcbfrst,fcbfrst+1,fcbuse,fcbuse+1
                dfb          fcbeof,fcbeof+1,fcbeof+2
*
* the following with $80+ are ignored by setinfo
*
inftabl         dfb          d.attr,d.filid,d.auxid,d.auxid+1
                dfb          $80+d.stor,$80+d.usage,$80+d.usage+1,d.moddt
                dfb          d.moddt+1,d.modtm,d.modtm+1,d.credt
                dfb          d.credt+1,d.cretm,d.cretm+1
*
                msb          on
*
********************* see rev note #36 *********************
*
death           asc          " RESTART SYSTEM    " "
************************************************************
                msb          off
*
******************* see rev note #en2 **********************
*
                ifeq         os-ednet
atalkstr        asc          'ATALK'
                fin
************************************************************

; ###############################################################################################
; #   END OF FILE:  DATATBLS
; #   LINES      :  144
; #   CHARACTERS :  5433
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DDPDRIVER
; ###############################################################################################

                do          nolap
                else
; can be further reduced if we change cmndlist instead
wlapparm        equ         cmndlist+2              ; 3 bytes as parameter to call the card
writelap        equ         *
                sta         wlapparm+1
                iny
                lda         (cmndlist),y
                sta         wlapparm+2
                iny                                 ; now y = 3
                sty         wlapparm
                ldy         #0
                ldx         #>wlapparm
                jsr         callapbus               ; call the card
                sta         status
                rts
                fin
;
wdsadr          equ         cmndlist+2
sizeinzp        equ         cmndlist+4
databuf         equ         cmndlist+6
;
writeddp        equ         *
                php
                sei                                 ; disable interrupt
                sta         ddphead+ld.protocol
                iny
                lda         (cmndlist),y
                beq         wddp.8a                 ; socket 0 is illegal
                jsr         srchscktable            ; is socket open ?
                beq         wddp.8b                 ; yes, proceed
wddp.8a         lda         #scktnotopen            ; else error
                bne         wddp.9                  ; bra
wddp.8b         jsr         setdestadr
                lda         (cmndlist),y            ; wds address
                tax
                iny
                lda         (cmndlist),y
                tay
                jsr         dowrtddp
wddp.9          plp
wddp.9a         sta         status
                rts
;
setdestadr      equ         *                       ; before input set y to 2 and acc to
srce socket
                sta         ddphead+sd.ssckt
                iny
                lda         (cmndlist),y            ; network high byte
                sta         ddphead+ld.dnet
                iny
                lda         (cmndlist),y
                sta         ddphead+ld.dnet+1
```

```
                iny
                lda         (cmndlist),y            ; destination node
                sta         lapdest
                iny
                lda         (cmndlist),y            ; get destination socket
                sta         ddphead+sd.dsckt
                iny
                lda         (cmndlist),y
                and         #$01
                sta         chksmf                  ; checksum flag
                iny
                rts
;
dowrtddp        equ         *                       ; make a ddp packet and write it out
                stx         wdsadr                  ; wds address
                stx         wddptbl
                sty         wdsadr+1
                sty         wddptbl+1
                lda         ddphead+ld.dnet         ; get network
                ldx         ddphead+ld.dnet+1
                bne         chknet
                tay                                 ; if low byte is 0, how about high byte
                beq         wrtshort                ; network = 0, do short ddp
chknet          cpx         thisnet+1               ; check to see if destination net is
local
                bne         wrtlong
                cmp         thisnet
                bne         wrtlong
                do          savespace
wrtshort        ldy         #move3-movetable
                jsr         movedata
                beq         wddp.2                  ; since movedata set z flag, bra
wrtlong         ldy         #move2-movetable
                jsr         movedata
                jsr         headersum               ; calcuate checksum of the header
                ldx         a.bridge                ; do we have a bridge
                bne         wddp.1a                 ; yes, then send packet to bridge
                lda         thisnet                 ; no bridge, then try it on same
network
                ora         thisnet+1               ; do we have a network number ?
                beq         wddp.2                  ; no, then send to same node else force
error
wddp.1a         stx         lapdest                 ; send to bridge or node 0 (force
error)
                else
wrtshort        lda         #1
                sta         laptype
                lda         ddphead+ld.protocol
                sta         ddphead+sd.protocol
                lda         #5
                sta         ddphead+sd.length+1
                lda         #8
                sta         wddpsize
                lda         #0
                sta         ddphead+sd.length
                sta         chksmf                  ; never do checksum on short ddp
                beq         wddp.2                  ; bra
wrtlong         lda         #2
                sta         laptype                 ; long ddp is protocol 2
                lda         ddphead+sd.ssckt
                sta         ddphead+ld.ssckt
                lda         ddphead+sd.dsckt
                sta         ddphead+ld.dsckt
```

```
                lda         #16                     ; lap + long atp header is 16 bytes
                sta         wddpsize
                lda         #13
                sta         ddphead+ld.length+1
                lda         lapdest
                sta         ddphead+ld.dnode
                lda         ournode
                sta         ddphead+ld.snode
                lda         thisnet
                sta         ddphead+ld.snet
                lda         thisnet+1
                sta         ddphead+ld.snet+1
                jsr         headersum               ; calcuate checksum of the header
                sta         ddphead+ld.length       ; headersum return acc = 0
                sta         ddphead+ld.chksum       ; for the case of no checksum
                sta         ddphead+ld.chksum+1
                ldx         a.bridge                ; do we have a bridge
                bne         wddp.1a                 ; yes, then send packet to bridge
                lda         thisnet                 ; no bridge, then try it on same
network
                ora         thisnet+1               ; do we have a network number ?
                beq         wddp.2                  ; no, then send to same node
                lda         #toomanycollison        ; else error, no bridge connected
                jmp         wddp.9a                 ; error return
wddp.1a         stx         lapdest                 ; send to bridge or node 0 (force
error)
                fin
wddp.2          ldy         #3                      ; copy our header addr to first 4 byte
of wds
wddp.3          lda         wddpsize,y
                sta         (wdsadr),y
                dey
                bpl         wddp.3
wddp.4          clc                                 ; first adjust for size of datagram
header
                lda         #4
                adc         wdsadr
                sta         wdsadr
                bcc         *+4
                inc         wdsadr+1
                ldy         #3
wddp.5          lda         (wdsadr),y
                sta         sizeinzp,y
                dey
                bpl         wddp.5
                cpy         sizeinzp+1              ; since y is $ff, check for end of list
                beq         wddp.6
                clc                                 ; note accumator is sizeinzp
                adc         ddphead+sd.length+1
                sta         ddphead+sd.length+1
                lda         sizeinzp+1
                adc         ddphead+sd.length
                sta         ddphead+sd.length
                jsr         dochecksum
                beq         wddp.4                  ; since dochecksum always set z flag,
bra
;
wddp.6          lda         chksmf
                beq         wddp.7
                lda         curcksum
                ldx         curcksum+1
                bne         *+7                     ; if low byt not 0, checksum non-zero
                tay                                 ; is high byte 0
```

```
                bne         *+4                     ; no, then checksum not zero
                dex                                 ; x = $ff
                txa                                 ; a = $ff
                sta         ddphead+ld.chksum
                stx         ddphead+ld.chksum+1
wddp.7          equ         *
                ifne        opers-pascal
                ldy         #<wddpparm
                ldx         #>wddpparm
                else
                ldy         adrwddpparm+1
                ldx         adrwddpparm
                fin
;
; jsr callapbus       ; replace by fall through
; rts
;
callapbus       jsr         applebus
                beq         callap.rts
                adc         #$5f                    ; if error, carry was set
callap.rts      rts
;
;
;
headersum       ldx         #4                      ; set up parameter to checksum the
header
hdrsum.1        lda         chksumhead-1,x
                sta         sizeinzp-1,x
                dex
                bne         hdrsum.1
                stx         curcksum                ; initial current checksum to 0
                stx         curcksum+1
;
; fall through to docheckсum
;
docheckсum      lda         chksmf                  ; do checksum ?
                bne         mustchksum              ; definitely do checksum
                rts                                 ; else return with z flag set
;
;
rdatpheader     equ         *                       ; read the atp header and calculate
checksum
                ldy         #<ratpparm
                ldx         #>ratpparm
;
; caution, readhdrsum destorys many zero page locations
;
readhdrsum      equ         *                       ; read the header and calculate the
checksum
                sty         databuf+1               ; save the parmlist pointer we are
using
                stx         databuf
                jsr         readheader
                bne         chksum.9
                lda         chksmf
                beq         chksum.9
                ldy         #4
                lda         (databuf),y
                sta         sizeinzp+1
                dey
                lda         (databuf),y
                sta         sizeinzp
                dey
```

```
                lda         (databuf),y
                tax
                dey
                lda         (databuf),y
                sta         databuf
                stx         databuf+1
;
mustchksum      ldx         sizeinzp                ; put low byte of byte count in x
chksum.0        ldy         #0
chksum.1        txa                                 ; look at low byte of byte count
                bne         chksum.2                ; then we have not reach 0, just dec
low byte
                dec         sizeinzp+1              ; else also decrement high byte
                bmi         chksum.8                ; if high byte was 0, done
chksum.2        dex
                clc
                lda         (databuf),y
                adc         curcksum+1
                sta         curcksum+1
                bcc         chksum.3
                inc         curcksum
chksum.3        rol         a                       ; high bit of low byte into carry
                rol         curcksum                ; high bit of low byte into low bit of
high byte
                rol         curcksum+1              ; high bit of high byte into low bit of
low byte
                iny
                bne         chksum.1
                inc         databuf+1
                jmp         chksum.0
chksum.8        txa                                 ; set z flag
chksum.9        rts
;
chksumhead      dw          9,ddphead+4
;
                ifeq        opers-pascal
adrwddpparm     dw          wddpparm
                fin

; ###############################################################################################
; #   END OF FILE:  DDPDRIVER
; #   LINES      :  263
; #   CHARACTERS :  12017
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.DESTROY.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DESTROY
; ###############################################################################################

                page
*
newline         ldy         #c.isnewl               ;adjust newline status for open file.
                lda         (par),y                 ;on or off?
                ldx         fcbptr                  ;it will be zero if off.
                sta         fcb+fcbnlmsk,x          ;set new line mask
                iny
                lda         (par),y                 ;and move in new 'new-line' byte.
                sta         fcb+fcbnewl,x
                clc
                rts                                 ;no error possible.
*
                page
*
getinfo         jsr         findfile                ;look for file they want ot know about.
                bcc         gtinfo1                 ;branch if no errors.
                cmp         #badpath                ;was it a root directory file?
                sec                                 ;(in case of no match)
                bne         ginfoerr
                lda         #$f0
                sta         dfil+d.stor             ;for get info, report proper storage
type
                lda         #0                      ;force a count of free blocks.
                sta         reql
                sta         reqh
                ldx         vcbptr
                jsr         tkfrecnt                ;take a fresh count of free blocks on
this volume.
                ldx         vcbptr
                lda         vcb+vcbtfre+1,x         ;return total blocks and total in use.
                sta         reqh                    ;first transfer 'free' blocks to zpage
for later subtract
                lda         vcb+vcbtfre,x           ; to determine the 'used' count
                sta         reql
                lda         vcb+vcbtblk+1,x         ;transfer to 'd.' table as aux i.d.
                sta         dfil+d.auxid+1          ;(total block count is considered aux
i.d. for the volume)
                pha
                lda         vcb+vcbtblk,x
                sta         dfil+d.auxid
                sec                                 ;now subtract and report the number of
blocks 'in use'
                sbc         reql
                sta         dfil+d.usage
                pla
                sbc         reqh
                sta         dfil+d.usage+1
gtinfo1         lda         dfil+d.stor             ;transfer bytes from there internal
order to call spec
                lsr         a                       ; via 'inftabl' translation table.
                lsr         a
                lsr         a                       ;but first change storage type to
                lsr         a                       ; external (low nibble) format.
```

```
                sta     dfil+d.stor
                ldy     #c.cretime+1            ;index to last of users spec table.
gtinfo2         lda     inftabl-3,y
                and     #$7f                    ;strip bit used by setinfo
                tax
                lda     dfil,x                  ;move directory info to call spec.
table.
                sta     (par),y
                dey
                cpy     #c.attr                 ;have all info bytes been sent?
                bcs     gtinfo2                 ;branch if not.
                clc                             ;indicate no errors
ginfoerr        rts
                page
*
setinfo         jsr     findfile                ;find what user wants...
                bcs     sinfoerr                ;return any failure.
                lda     bubit                   ;discover if backup bit can be cleared.
                eor     #$20
                and     dfil+d.attr
                and     #$20
                sta     bkbitflg                ;or preserve current...
                ldy     #c.modtime+1            ;init pointer to user supplied list.
setinf1         ldx     inftabl-3,y             ;get index into coresponding 'd.' table
                bmi     setinf2                 ;branch if we've got a non setable
parameter.
                lda     (par),y
                sta     dfil,x
setinf2         dey                             ;has user's request been satisfied?
                cpy     #c.attr
                bcs     setinf1                 ;no, move next byte.
*
                and     #$18                    ;make sure no illegal access bits were
set!
                beq     setinf3                 ;branch if legal access
                lda     #accserr                ;otherwise, refuse to do it.
                sec                             ;indicate error.
sinfoerr        rts
*
setinf3         ldy     #c.moddate+1
                lda     (par),y                 ;was clock nul input?
                beq     setinf4
                jmp     drevise1                ;end by updating directory.
*
setinf4         jmp     drevise                 ;update with clock also...
*
                page
*
rename          jsr     lookfile                ;look for source (original) file.
                bcc     rname0                  ;branch if found.
                cmp     #badpath                ;trying to rename a volume?
                bne     rnmeror                 ;no, return other error.
                jsr     renpath                 ;syntax new name.
                bcs     rnmeror
                ldy     pathbuf                 ;find out if only rootname for new name
                iny
                lda     pathbuf,y               ;must be $ff if v-name only
                bne     rnbadpth                ;branch if not single name.
                ldx     vcbptr                  ;text for open files before changing.
                lda     vcb+vcbstat,x
                bpl     rnamevol                ;branch if volume not busy
                lda     #filbusy
rnmeror         sec
```

```
                rts
rnamevol        ldy     #0                      ;get newname's length.
                lda     pathbuf,y
                ora     #$f0                    ;(root file storage type)
                jsr     mvrotnam                ;update root directory.
                bcs     rnamerr
                ldy     #0
                ldx     vcbptr                  ;update vcb also.
rnmevol         lda     pathbuf,y               ;move new name to vcb.
                beq     rnmevol1
                sta     vcb,x
                iny                             ;bump to next character.
                inx
                bne     rnmevol                 ;branch alwasy taken.
rnmevol1        clc                             ;no errors
                rts
*
rname0          jsr     getnamptr               ;set y to first char of path, x=0
rname1          lda     pathbuf,y               ;move original name to gbuf
                sta     gbuf,x                  ; for later comparison with new name.
                bmi     rname2                  ;branch if last character has been
moved.
                iny                             ;otherwise, get the next one.
                inx
                bne     rname1                  ;branch always taken.
*
                page
rname2          jsr     renpath                 ;get new name syntaxed.
                bcs     rnamerr
                jsr     getnamptr               ;set y to path, x to 0
                lda     pathbuf,y               ;now compare new name with old name
rname3          cmp     gbuf,x                  ; to make sure that they are in the
same directory.
                php                             ;save result of compare for now,
                and     #$f0                    ;was last char really a count?
                bne     rname4                  ;branch if not.
                sty     rnptr                   ;save pointer to next name, it might be
the last.
                stx     pnptr
rname4          plp                             ;what was the result of the compare?
                bne     rname5                  ;branch if different character or
count.
                inx                             ;bump pointers.
                iny
                lda     pathbuf,y               ;was that the last character?
                bne     rname3                  ;branch if not.
                clc                             ;no-operation, names were the same.
                rts
*
rname5          ldy     rnptr                   ;index to last name in the chains.
                lda     pathbuf,y               ;get last name length.
                sec
                adc     rnptr
                tay
                lda     pathbuf,y               ;this byte should be $00!
                bne     rnbadpth                ;branch if not.
                ldx     pnptr                   ;index to last of original name.
                lda     gbuf,x
                sec
                adc     pnptr
                tax
                lda     gbuf,x                  ;this byte should also be $00
                beq     rname6                  ;continue processing if it is.
```

```
*
rnbadpth        lda         #badpath
rnamerr         sec
                rts                                 ;report error.
*
rname6          jsr         lookfile                ;test for duplicate file name.
                bcs         rname7                  ;branch if file not found, which is
what we want!
                lda         #duperr                 ;new name already exists.
                sec                                 ;report duplicate.
                rts
                page
rname7          cmp         #fnferr                 ;was it a valid file not found?
                bne         rnamerr                 ;no, return other error code.
                jsr         setpath                 ;now syntax the pathname of the file to
be changed.
                jsr         findfile                ;get all the info on this one.
                bcs         rnamerr
                jsr         tstopen                 ;don't allow rename to occur if file is
in use.
                lda         #filbusy                ;anticipate error
                bcs         rnamerr
                lda         dfil+d.attr             ;test bit that says it's ok to rename
                and         #renamen
                bne         rname8                  ;branch if it's alright to rename.
                lda         #accserr                ;otherwise report illegal access.
rname10         sec
                rts
*
rname8          equ         *
                lda         dfil+d.stor             ;find out which storage type.
                and         #$f0                    ;strip off name length.
                cmp         #dirtyp*16              ;is it a directory?
                beq         rname11
                cmp         #tretyp+1*$10           ;is it a seed, sapling, or tree?
                bcc         rname11
                lda         #cpterr
                bne         rname10
rname11         jsr         renpath                 ;well... since both names would go into
the
                bcs         rnamerr                 ; directory, re-syntax the new name to
get local name address.
                ldy         rnptr                   ;(y contains index to local name
length)
                ldx         pathbuf,y               ;adjust y to last char of new name.
                tya
                adc         pathbuf,y
                tay
rname9          lda         pathbuf,y               ;move local name to dir entry
workspace.
                sta         dfil+d.stor,x
                dey
                dex
                bne         rname9
                lda         dfil+d.stor             ;preserve file storage type.
                and         #$f0                    ;strip off old name length.
                tax
                ora         pathbuf,y               ;add in new name's length
                sta         dfil+d.stor
                cpx         #dirtyp*16              ; that file must be changed also.
                bne         rnamdone                ;branch if not directory type.
                page
                lda         dfil+d.frst             ;read in 1st (header) block of sub-dir
```

```
                ldx         dfil+d.frst+1
                jsr         rdblk
                bcs         rnamerr                 ; report errors
                ldy         rnptr                   ;change the header's name to match the
owner's new name.
                lda         pathbuf,y               ; get local name length again
                ora         #hedtyp*16              ;assume it's a header.
                jsr         mvrotnam
                bcs         rnamerr
rnamdone        jmp         drevise1                ;end by updating all path directories
*
*
mvrotnam        ldx         #0
mvhednam        sta         gbuf+4,x
                inx
                iny
                lda         pathbuf,y
                bne         mvhednam
                jmp         wrtgbuf                 ;write changed header block.
*
*
renpath         ldy         #c.nwpath               ;get address to new pathname.
                lda         (par),y
                iny
                sta         tpath
                lda         (par),y                 ;set up for syntaxing routine
(synpath).
                sta         tpath+1
                jmp         synpath                 ;go syntax it. (returns last local name
length in y).
*
getnamptr       ldy         #0                      ;return pointer to first name of path.
                bit         prfxflg                 ;is this a prefixed name?
                bmi         getnptr                 ;branch if not.
                ldy         newpfxptr
getnptr         ldx         #0
                rts
*
                page
*
destroy         jsr         findfile                ;look for file to be wiped out.
                bcs         desterr                 ;pass back any error.
                jsr         tstopen                 ;is this file open?
                lda         totent
                bne         desterr1                ;branch if file open.
*
dstroy1         lda         #0                      ;force proper free count in volume.
                sta         reql                    ;(no disk access occurs if already
proper)
                sta         reqh
                jsr         tsfrblk
                bcc         dstroy2
                cmp         #ovrerr                 ;was it just a full disk?
                bne         desterr                 ;nope, report error.
*
dstroy2         lda         dfil+d.attr             ;make sure it's ok to destroy this
file.
                and         #dstroyen
                bne         dstroy3                 ;branch if ok.
                lda         #accserr                ;tell user it's not kosher.
                jsr         syserr                  ;(returns to caller of destory)
*
dstroy3         lda         devnum
```

```
                jsr         twrprot1                ;before going thru deallocation,
                bcs         desterr                 ; test for write protected hardware.
                lda         dfil+d.frst             ;"detree" needs first block addr.
                sta         firstbl
                lda         dfil+d.frst+1
                sta         firstbh
                lda         dfil+d.stor             ;find out which storage type.
                and         #$f0                    ;strip off name length.
                cmp         #tretyp+1*$10           ;is it a seed, sapling, or tree?
                bcc         dstree                  ;branch if it is.
                jmp         dstdir                  ;otherwise test for directory destroy.
*
desterr1        lda         #filbusy
desterr         sec                                 ;inform user that file can't be
destroyed at this time.
                rts
*
                page
dstree          equ         *                       ;destroy a tree file.
                sta         stortype                ;save storage type.
                ldx         #5
                lda         #0                      ;set "detree" input variables
dstre1          sta         stortype,x              ;variables must be
                dex                                 ; in order:deblock, dtree, dsap, dseed
                bne         dstre1                  ;loop until all set to zero.
                lda         #2
                sta         dseed+1                 ;this avoids an extra file i/o.
*
********************** see rev note #73 **********************
********************** see rev note #49 **********************
********************** see rev note #41 **********************
*
                inc         delflag                 ; don't allow detree to zero index
blocks.
                jsr         detree                  ;make trees and saplings into seeds
                dec         delflag                 ; reset flag.
************************************************************
                bcs         desterr                 ;(de-evolution).
dstlast         ldx         firstbh
                lda         firstbl                 ;now deallocate seed.
                jsr         dealloc
                bcs         desterr
*
                lda         #0                      ;update directory to free entry space.
                sta         dfil+d.stor
                cmp         h.fcnt                  ; file entry wrap?
                bne         dst1                    ; branch if no carry adjustment
                dec         h.fcnt+1                ; take carry from high byte of file
entries
dst1            dec         h.fcnt                  ; mark header with one less file
*
                jsr         upbmap
                bcs         desterr
                jsr         dvcbrev                 ;go update block count in vcb.
                jmp         drevise                 ;update directory last...
*
*
dvcbrev         ldy         vcbptr
                lda         deblock
                adc         vcb+vcbtfre,y
                sta         vcb+vcbtfre,y           ;update current free block count.
                lda         deblock+1
                adc         vcb+vcbtfre+1,y
```

```
                sta     vcb+vcbtfre+1,y
                lda     #0                      ;force rescan from first bitmap
                sta     vcb+vcbcmap,y
                rts
*
*
                page
*
dstdir          cmp     #dirtyp*16              ;is this a directory file?
                bne     drcpterr                ;no, report file incompatable.
*
dsdir1          jsr     fndbmap                 ;make sure a buffer is available for
the bitmap.
                bcs     dsdirerr
                lda     dfil+d.frst             ;read in first block of directory into
gbuf.
                sta     bloknml
                lda     dfil+d.frst+1
                sta     bloknmh
                jsr     rdgbuf
                bcs     dsdirerr
                lda     gbuf+hcent+4            ;find out if any files exist on this
directory.
                bne     dsdiracc                ;branch if any exist.
                lda     gbuf+hcent+5
                beq     dsdir1a
dsdiracc        lda     #accserr
                jsr     syserr
*
dsdir1a         sta     gbuf+4                  ;make it an invalid subdir.
                jsr     wrtgbuf
                bcs     dsdirerr
dsdir2          lda     gbuf+2                  ;get forward link.
                cmp     gbuf+3                  ;test for no link.
                bne     dsdir3
                cmp     #0
                beq     dstlast                 ;if no link, then finished.
dsdir3          ldx     gbuf+3
                jsr     dealloc                 ;free this block.
                bcs     dsdirerr
                lda     gbuf+2
                ldx     gbuf+3
                jsr     rdblk
                bcc     dsdir2                  ;loop until all are freed.
dsdirerr        rts
*
drcpterr        lda     #cpterr                 ;file is not compatable.
                jsr     syserr                  ;give up.
*
fcbused         equ     *                       ; mark as fcb as dirty so
* the directory will be flushed on 'flush'
                pha
                tya
                pha                             ; save regs
                ldy     fcbptr
                lda     fcb+fcbdirty,y          ; fetch current fcbdirty byte
                ora     #fcbmod                 ; mark fcb as dirty
                sta     fcb+fcbdirty,y          ; save it back
                pla
                tay                             ; and restore regs
                pla
                rts
```

```
; ###############################################################################################
; #   END OF FILE:  DESTROY
; #   LINES      :  399
; #   CHARACTERS :  21645
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.DETREE.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DETREE
; ###############################################################################################

               page
*
* "detree" deallocates blocks from tree files.
*
* it is assumed that the device preselected and the 'gbuf' may be used.
*
* on entry the following values must be set:
*   stortype = storage type in upper nibble, lower nibble is undisturbed.
*   firstbl & firstbh = first block of file (index or data)
*   deblock = 0 (see below)
*   dtree = ptr to 1st block with stuff to be deallocated at tree level.
*   dsap = ptr to 1st block at sapling level
*   dseed = byte (0-511) position to be zeroed from (inclusive).
*
* on exit:
*   stortype = modified result of storage type (if applicable)
*   firstbl & h = modified if storage type changed.
*   deblock = total number of blocks freed at all levels.
*   dtree, dsap, dseed unchanged.
*
* to trim a tree to a seed file, both dtree and dsap must be zero.
* to go from tree to sapling, dtree alone must be zero.
*
*
detree          lda          stortype                  ;which flavor of tree?
                cmp          #$20                      ;is it a 'seed' (<=$1f)
                bcc          seedeal                   ;branch if yes.
                cmp          #$30                      ;maybe a 'sapling'?
                bcc          sapdeal
                cmp          #$40                      ;well, at least be certain it's a
'tree'.
                bcc          tredeal                   ;branch if it is.
* ----------------- see rev note 22 ----------------------------
                lda          #alcerr                   ;block allocation error
* ------------------------------------------------------------
                jsr          sysdeath                  ;should never have been called.
*
seedeal         jmp          seedel0
sapdeal         jmp          sapdel0
*
*
tredeal         lda          #$80
                sta          topdest                   ;for tree top, start at end, work
backwards.
tredel1         jsr          drdfrst                   ;read specified first block into gbuf.
                bcs          bummerr                   ;return all errors.
                ldy          topdest                   ;get current pointer to top indexes.
                cpy          dtree                     ;have enough sapling indexes been
deallocated?
                beq          tredel7                   ;yes, now deallocate top guys!
                ldx          #7                        ;buffer up to 8 sapling index block
addrs.
tredel2         lda          gbuf,y                    ;fetch low block address
```

```
                sta     dealbufl,x              ; and save it.
                ora     gbuf+$100,y             ;is it a real block that is allocated?
                beq     tredel3                 ;branch if phantom block.
                lda     gbuf+$100,y             ;fetch hi block addr.
                sta     dealbufh,x              ; and save it.
                dex                             ;decrement and test for dealc buf
filled.
                bmi     tredel5                 ;branch if we've fetched 8 addresses.
*
                page
tredel3         dey                             ;look now for end of deallocation
limit.
                cpy     dtree                   ; is this the last position on tree
level?
                bne     tredel2                 ;branch if not.
                iny
                lda     #0                      ;fill rest of dealc buffer with nul
addresses.
tredel4         sta     dealbufl,x
                sta     dealbufh,x
                dex
                bpl     tredel4                 ;loop until filled.
*
tredel5         dey                             ;decrement to prepare for next time.
                sty     topdest                 ;save index.
                ldx     #7
tredel6         stx     dtmpx                   ;save index to dealc buf.
                lda     dealbufl,x
                sta     bloknml
                ora     dealbufh,x              ;are we finished?
                beq     tredel1                 ;branch if done with this level.
                lda     dealbufh,x              ;complete address with hi byte,
                sta     bloknmh
                jsr     rdgbuf                  ;read sapling level into gbuf.
                bcs     bummerr                 ;return any errors.
                jsr     dealblk                 ;go free all data indexes in this
block.
                bcs     bummerr
                skp     1
**************** see rev note #73 ******************
                jsr     wrtgbuf
                bcs     bummerr
                skp     1
                ldx     dtmpx                   ;restore index to dealc buff.
                dex                             ;are there more to free?
                bpl     tredel6                 ;branch if there are.
                bmi     tredel1                 ;branch always to do next bunch.
*
dtredone        equ     *
dsapdone        clc                             ;indicate no errors.
bummerr         rts                             ;return point for errors.
*
*
tredel7         ldy     dtree                   ;now deallocate all tree level
                iny                             ; blocks greater than specified block.
                jsr     dalblk1                 ;(tree top in gbuf)
                bcs     bummerr                 ;report any errors.
                jsr     wrtgbuf                 ;write updated top back to disk.
                bcs     bummerr
                ldy     dtree                   ;no figure out if tree can become
sapling
                beq     tredel8                 ;branch if it can!
                lda     gbuf,y                  ;otherwise, continue with partial
```

```
                sta         bloknml                 ; deallocation of last sapling index.
                ora         gbuf+$100,y             ;is there such a sapling index block?
                beq         dtredone                ;all done if not!
                lda         gbuf+$100,y             ;read in sapling level to be modified.
                sta         bloknmh
                jsr         rdgbuf                  ;read 'highest' sapling index into
gbuf.
                bcc         sapdel1
                rts
*
*
tredel8         jsr         shrink                  ;shrink tree to sapling.
                bcs         bummerr
*
*
sapdel0         jsr         drdfrst                 ;read specified only sapling level
index into gbuf.
                bcs         bummerr
sapdel1         ldy         dsap                    ;fetch pointer to last of desirable
indexes
                iny                                 ; bump to the first undesirable.
                beq         sapdel2                 ;branch if all are desirable.
                jsr         dalblk1                 ;deallocate all indexes above
appointed.
                bcs         bummerr
                jsr         wrtgbuf                 ;update disk with remaining indexes.
                bcs         bummerr
*
sapdel2         ldy         dsap                    ;now prepare to cleanup last data
block.
                beq         sapdel4                 ;branch if there is a posiblity of
making it a seed.
sapdel3         lda         gbuf,y                  ;fetch low order data block addr.
                sta         bloknml
                ora         gbuf+$100,y             ;is it a real block?
                beq         dsapdone                ;we're done if not.
                lda         gbuf+$100,y
                sta         bloknmh
                jsr         rdgbuf                  ;go read data block into gbuf.
                bcc         seedel1                 ;branch if good read
                rts                                 ;otherwise return error.
*
sapdel4         lda         dtree                   ;is both tree and sap levels zero?
                bne         sapdel3                 ;branch if not.
                jsr         shrink                  ;reduce this sap to a seed.
                bcs         bumerr1
*
* if no error, drop into seedel0
*
seedel0         jsr         drdfrst                 ;go read only data block.
                bcs         bumerr1                 ;report any errors.
seedel1         ldy         dseed+1                 ;check hi byte for no deletion
                beq         seedel2                 ;branch if all of second page is to be
deleted.
                dey                                 ;if dseed>$200 then were all done!
                bne         dseedone                ;branch if thats the case.
                ldy         dseed                   ;clear only bytes >= dseed.
seedel2         lda         #0
seedel3         sta         gbuf+$100,y             ;zero out unwanted data.
                iny
                bne         seedel3
                ldy         dseed+1                 ;was that all?
                bne         seedel5                 ;branch if it was.
```

```
                ldy         dseed
seedel4         sta         gbuf,y
                iny
                bne         seedel4
seedel5         jmp         wrtgbuf                 ;update data block to disk.
*
dseedone        clc                                 ;indicate no error.
bumerr1         rts                                 ;return error status.
*
drdfrst         lda         firstbl                 ;read specified first block into gbuf.
                ldx         firstbh
                jmp         rdblk                   ;go do it!
                skp         1
*************** see rev note #73 *****************
                skp         1
*********************************************
* beware that dealloc may bring in a new bitmap block and
* may destroy locations 46 and 47 which use to point
* to the current index block.
*********************************************
                skp         1
shrink          ldx         firstbh                 ;first deallocate top block.
                txa
                pha
                lda         firstbl
                pha                                 ;save block address of this index block
                jsr         dealloc                 ;go do it.
                pla
                sta         bloknml                 ;set master of sapling index block
address
                pla
                sta         bloknmh                 ;
                bcs         bumerr2                 ;report any errors.
                lda         gbuf                    ;get first block at lower level
                sta         firstbl
                lda         gbuf+$100
                sta         firstbh
                skp         1
                ldy         #$0
                jsr         swapme
                skp         1
                sec                                 ;now change file type, from
                lda         stortype                ; tree to sapling,
                sbc         #$10                    ; or from sapling to seed!
                sta         stortype
                skp         1
                jsr         wrtgbuf
                skp         1
*clc
bumerr2         rts                                 ;return error status.
*
*
dealblk         ldy         #0                      ;begin at the beginning.
dalblk1         lda         bloknml                 ;save disk address of
                pha                                 ; gbuf's data.
                lda         bloknmh
                pha
dalblk1a        sty         saptr                   ;save current index.
                lda         gbuf,y                  ;get address (low) of block to be
deallocated.
                cmp         gbuf+$100,y             ;test for nul block.
                bne         dalblk2                 ;branch if not nul.
                cmp         #0
```

```
                beq          dalblk3                    ;skip it if nul.
dalblk2         ldx          gbuf+$100,y                ;get the rest of the block address.
                jsr          dealloc                    ;free it up on volume bit map.
                bcs          dalblkerr                  ;return any error.
                ldy          saptr                      ;get index to sapling level index block
again.
                skp          1
                jsr          swapme
                skp          1
*
**************** see rev note #73 *****************
***************** see rev note #49 *****************
******************* see rev note #41 ***********************
*
* lda delflag ; are we being called from delete?
* bne dalblk3 ; yes, so don't zero index blocks!!
************************************************************
*lda #0
*sta gbuf,y
*sta gbuf+$100,y ;remove address form index block.
dalblk3         iny                                     ;point at next block address.
                bne          dalblk1a                   ;branch if more to deallocate (or
test).
                clc                                     ;indicate no error.
dalblkerr       tax                                     ;save error code, if any.
                pla                                     ;restore bloknml&h
                sta          bloknmh
                pla
                sta          bloknml
                txa                                     ;restore return code.
                rts
*
                skp          1
**************** see rev note #73 *****************
                skp          1
swapme          lda          delflag                    ;are we swapping or 0ing ?
                bne          swapme1                    ;skip if swapping
                tax                                     ;make x a 0
                skp          1
****************** see rev note 75 *******************
                beq          swapme2                    ;0 the index  (always taken)
                skp          1
swapme1         ldx          gbuf+$100,y                ;get index, hi
                lda          gbuf,y                     ;get index, lo
                skp          1
swapme2         sta          gbuf+$100,y                ;save index, hi
                txa
                sta          gbuf,y                     ;save index, lo
                rts                                     ;we done

; ###############################################################################################
; #   END OF FILE:  DETREE
; #   LINES      :  270
; #   CHARACTERS :  14617
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.DEVSRCH.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DEVSRCH
; ###############################################################################################

greet           lda         $c030
                sta         $c00c                   ;disable 80 columns (rev-e)
                sta         $c000                   ;disable '80store'
                jsr         $fe84
                jsr         $fb2f
                jsr         $fe93
                jsr         $fe89
                cld
* sei ; don't turn interrupts off!! *** see rev note #44 ***
                jsr         home                    ;clear screen
                ldx         #slin2-slin1-1          ;move greeting to screen.
grt1            lda         slin1,x                 ;"apple ii"
                sta         $4a8+20-4,x             ;$4a8 starts the line.
                dex
                bpl         grt1
                ldx         #slin3-slin2-1
grt2            lda         slin2,x                 ;"prodos (version)   (date)"
                sta         $5a8+20-grtlz,x         ;$5a8 starts the line
                dex
                bpl         grt2
                ldx         #slin4-slin3-1
grt3            lda         slin3,x                 ;(blank line)
                sta         $6a8+20-grtlzz,x        ;$6a8 starts the line
                dex
                bpl         grt3
                ldx         #slin5-slin4-1
grt4            lda         slin4,x
                sta         $750,x                  ;$750 starts the line
                dex
                bpl         grt4
                ldx         #grt6-slin5-1
grt5            lda         slin5,x
                sta         $7d0+20-grtlzzz,x       ;$7d0 starts the line
                dex
                bpl         grt5
                lda         $c030
                rts
*
slin1           asc         'APPLE II' ; 8 Chars Exactly'
                ifeq        os-prodos
slin2           asc         'PRODOS 8 V1.7        ' '
                fin
                ifeq        os-ednet
slin2           asc         'prodos ednet 1.0x01  '
                fin
* change 72 <------------------------------
                asc         '08-AUG-88' ;date (no leading spaces unless date <10)'
slin3           asc         '          '
slin4           asc         'COPYRIGHT APPLE COMPUTER, INC., 1983-88' ;39 chars '
slin5           asc         'ALL RIGHTS RESERVED.'
grt6            equ         *
grtlz           equ         slin3-slin2/2
grtlzz          equ         slin4-slin3/2
```

```
grtlzzz         equ          grt6-slin5/2
*
indrcn          equ          $10
devid           equ          $12
relc800         equ          $cfff
drv1adr         equ          $bf10
drv2adr         equ          $bf20
lstdev          equ          $bf30
* devcnt equ $bf31
* devlst equ $bf32
glbclk          equ          $bf06
* sltbyt equ $bf99
jump            equ          $4c
                page
************************************************
*
* this routine finds all disk devices plugged into the
*  system's slots that follow apple's disk id convention.
*  the routine sets up the address and device table in
*  prodos's system global page.  13-sector disk ii's are
*  not configured since they may not function properly in
*  the 16 sector mode...
*
* profiles, and other intelligent devices, are expected to
*  have roms containing the disk i/o drivers.
*
* this routine was revved 8/21/86 to make sure that correct number
*  of smartport devices are installed, and that disk //s are always
*  searched last by placing them at the end of the list.  disk // unit
*  numbers are stacked at the end of the device table, while regular
*  devices are stacked at the beginning.  after all slots have been
*  searched the two lists are combined. see note 60.
*
numdevices      ds           8,0                     ;eight bytes for smartport call
*
driveradr       dw           0
d2idx           dfb          0
diskins2        dfb          0                       ;msb set if drive in slot 2
*
************************************************
                skp          1
devsrch         ldx          #0
                stx          dst
                stx          dst+1
                stx          indrcn                  ;set up for search.
                dex
                stx          devcnt                  ;device count=none
                lda          #14                     ;start disk // area at the end of
devlist
                sta          d2idx
********************** see note #65 ***********************
*
* make a quick check of slot 2.  if there is a disk card there
*  clear the msb of diskins2.  this will limit the number of
*  devices in any slot 5 smartport card to 2.
*
                lda          #$c2
                sta          indrcn+1                ;check slot 2
                jsr          cmpid                   ;is there a disk card in slot 2??
                ror          diskins2                ;clear msb if so, set otherwise
*
************************************************************
*
```

```
                lda         #$c7                    ;search slots from high to low.
                sta         indrcn+1
                skp         1
findsks1        equ         *
*
***************** code here became a subroutine ****************
************************ see note #65 ************************
*
                jsr         cmpid
                bcs         nxtdsk                  ;carry set means no card this slot
*
*************************************************************
*
                lda         (indrcn),y              ;check last byte of $cn rom
                beq         diskii                  ;if =00 then 16 sector disk ii.
                cmp         #$ff                    ;if =ff then 13 sector disk ii.
                bcs         nxtdsk                  ;ignore if 13 sector boot rom.
                sta         driveradr               ;else, assume it's an intelligent disk
for now.
*
                ldy         #7                      ;check for a smartport device
                lda         (indrcn),y
                bne         nosport
                jmp         smartport
*
nosport         equ         *
                ldy         #$fe                    ;get attributes byte.
                lda         (indrcn),y
                and         #$03                    ; & verify it provides read, write, and
status calls
                cmp         #$03
                sec                                 ;assume it is a bozo brand disk...
                bne         nxtdsk
                skp         1
                jsr         setdevid
                clc
                php                                 ;remember that it's not a disk //
                lsr         a                       ;move # of units (0=1, 1=2) to carry.
                lda         indrcn+1                ;low entry addr already on stack, save
hi addr.
                bne         adevice                 ;branch always taken.
                skp         1
diskii          sta         devid                   ;disk ii's have nul attributes
                sec
                php                                 ;remember it's a disk //
                lda         d2blkio
                sta         driveradr
                lda         d2blkio+1
*
* the carry is already set telling installdev to install two devices
*  for disk //s.
*
adevice         equ         *
                sta         driveradr+1
*
                jsr         installdev              ;install one or two devices this slot
*
                plp                                 ;get back if it's disk //
                bcc         nodisk2
*
                dex                                 ;move the list pointer back
                dex         devcnt                  ;(installdev left x set)
                stx         devcnt
```

```
                dec         d2idx                   ;increase the disk two index
                dec         d2idx
                ldy         d2idx
*
                inx                                 ;increase x in case =$ff
                lda         devlst+1,x              ;
                sta         devlst,y
                lda         devlst,x
                sta         devlst+1,y
                dex                                 ;back to devcnt again
*
nodisk2         equ         *
*
*-------------------------------------------------------------
*
****************** see rev note #53 & #54 **********************
*
*ldy #$7 ; check smartport id rom byte at $cn07...
*lda (indrcn),y ; is it a $0?
*bne notsmartport ; branch if not.
*lda indrcn+1 ; now we are only special casing slot 5!
*cmp #$c5 ; is the smartport card in $c500 space?
*bne notsmartport ; branch if not.
*inx ; get to next unit number in devlst.
*lda #$2b ; deposit unit number $2b(slot 2, drive 1)
*sta devlst,x ; in devlst. (smartport unit 3)
*inx ; bump to next spot in devlst.
*lda #$ab ; deposit unit number $ab(slot 2, drive 2)
*sta devlst,x ; in devlst. (smartport unit 4)
*stx devcnt ; update device count.
*lda #$c5 ; now update hi byte
*sta drv1adr+5 ; of driver entry addresses...
*sta drv2adr+5 ;
*lda drv1adr+10 ; get low driver entry address...
*sta drv1adr+4 ; and update low bytes of
*sta drv2adr+4 ; driver entry addresses...
*notsmartport equ *
**********************************************
* -------------------- see rev note 33 ------------------------
* -------------------- see rev note 24 ----------------------
*lda indrcn+1 ;get slot #
*cmp #$c5 ;only slots 5 & 6 allow drives 3 & 4
*bcc nmdrvs ;  to be mapped into slot (#-4)
*cmp #$c7
*bcs nmdrvs
*lda devid
*and #$2 ;check for 3 or 4 drives
*beq nmdrvs ;branch if not
*lda devid
*lsr a ;move 4 drive bit into carry
*php ;  and save it.
*lda indrcn+1 ;get index to global device table for
*pha
*and #3 ;  this slot and map into lower slot.
*asl a
*tay
*pla
*and #7 ;use index again to..
*asl a
*asl a
*asl a ;  form device number.
*asl a
*and #$3f
```

```
*jsr stadrv
*plp ;get bit showing if 4 drives back
*ror a ;if 4 drives, then bit 7=1.
*bpl nodrv4 ;branch if a 3-drive device
*inx ;else assume that 4th drive is present.
*sta devlst,x ;  and add it to the device list.
*nodrv4 stx devcnt ;save updated device count.
*asl a ;shift # of drives back into carry.
*lda drv1adr+8,y ;get low address from parent slot
*pha
*lda indrcn+1 ;get high address of device driver.
*sta drv1adr+1,y ;save it in down-mapped slot
*bcc ndrv4a ;branch if 3 drives
*sta drv2adr+1,y ;save high address of drive 4
*pla
*pha
*sta drv2adr,y ;save low address for drive 4.
*ndrv4a pla ;now save low address for drive 3
*sta drv1adr,y
*nmdrvs equ *
* ------------------------------------------------------
*
nxtdsk2         equ          *
                clc
                skp          1
nxtdsk          jsr          sltrom                  ;test for clock & other devices
                skp          1
                dec          indrcn+1                ;set up for next lower slot.
                lda          indrcn+1                ;have all slots been checked?
                and          #7
                beq          *+5                     ;(to the tay)
                jmp          findsks1                ;branch if not.
*
* now copy the disk // list to the end of the regular list
*  start by making the device count include disk //s
*
                ldx          devcnt                  ;load up current devcnt-1
                lda          #14
                sec
                sbc          d2idx
                beq          d2snone                 ;if there were no disk //s, forget it
*
                clc
                adc          devcnt                  ;sum of disk //s and others
                sta          devcnt
*
                inx                                  ;move to open space in regular list
                ldy          #13                     ;first disk // entry
*
mlab            equ          *
                lda          devlst,y
                pha
                lda          devlst,x
                sta          devlst,y
                pla
                sta          devlst,x
*
                inx
                dey
                sty          d2idx                   ;use as a temp
                cpx          d2idx
                blt          mlab                    ;continue 'til indices cross...
*
```

```
d2snone         equ         *
                ldy         #0
                ldx         devcnt                  ;now change the device order
srtdsk1         lda         devlst,x                ; so that the boot device
                pha
                and         #$7f
                eor         lstdev                  ; will have highest priority
                asl         a
                bne         srtdsk2
                pla
                iny
srtdsk2         dex
                bpl         srtdsk1
                ldx         devcnt                  ;now reverse order of search, hi to lo.
                tya                                 ;was boot device found?
                beq         srtdsk3
                lda         lstdev                  ;make boot device first in search
order.
                sta         devlst,x
                dex
                bmi         dsrend                  ;branch if only one device.
                dey                                 ;is this a 2 drive device?
                bmi         srtdsk3                 ;branch if not.
                eor         #$80                    ;make boot device, drive 2 next.
                sta         devlst,x
                dex
                bmi         dsrend                  ;branch if only one device, 2 drives.
srtdsk3         pla
                sta         devlst,x
                dex
                bpl         srtdsk3
dsrend          jsr         fndtrd                  ;save accumuated machine id.
                beq         whosit
                sta         machid
                rts
                skp         1
whosit          jmp         whatsit
                skp         1
stadrv          equ         *
                ora         devid                   ;combine with attributes.
                ldx         devcnt
                inx                                 ;put device number into device list.
                sta         devlst,x
                asl         a                       ;now form drive 2 device number, if
any.
                rts
                skp         1
*
sltrom          bcc         isrom                   ;branch if disk drive.
                ldy         #6                      ;test this slot for clock card.
clkchk          lda         (indrcn),y
                cmp         clkid,y
                bne         notclk                  ;branch if not clock.
                dey
                dey
                bpl         clkchk
                lda         indrcn+1                ;transfer high slot addr minus $c1
(default)
                sbc         #$c1                    ; to relocate references to clock rom.
                sta         clock64
                lda         #jump                   ;also enable jump vector in globals.
                sta         glbclk
                lda         apple                   ;mark clock as present
```

```
                beq         dsrend
                ora         #1
                sta         apple
                bne         isrom                   ;branch always taken.
                skp         1
notclk          ldy         #5                      ;test for 80 col card.
                lda         (indrcn),y              ; byte at a time.
                cmp         #$38
                bne         notcons
                ldy         #7                      ;test values are same as pascal's
                lda         (indrcn),y
                cmp         #$18
                bne         notcons
                ldy         #$b
                lda         (indrcn),y
                cmp         #1
                bne         notcons
                iny
                lda         (indrcn),y
                and         #$f0                    ;mask off low nibble
                cmp         #$80                    ;generic for 80-col card
                bne         notcons
                lda         apple
                beq         dsrend
                ora         #2
                sta         apple                   ;mark config for 80 col. present.
                bne         isrom                   ;branch always taken.
                skp         1
notcons         ldy         #0                      ;test for any rom.
                ldx         #0
                lda         (indrcn),y
                cmp         #$ff                    ;test for apple iii non slot.
                beq         norom                   ;branch if invalid rom.
tstrom          cmp         (indrcn),y              ;look for floating bus.
                bne         norom
                inx                                 ;loop 256 times.
                bne         tstrom
isrom           lda         indrcn+1                ;mark a bit in slot byte
                and         #$7                     ; to indicate rom present.
                tax
                lda         sltbit,x
                ora         sltbyt
                sta         sltbyt
norom           rts
                skp         1
d2blkio         dw          rwts
dskid           equ         *
clkid           dfb         $08,$20,$28,$00,$58,$03,$70,$3c
sltbit          dfb         $00,$02,$04,$08,$10,$20,$40,$80
                skp         1
fndtrd          clc
                ldy         sltbit
ftrad1          lda         (look),y
                and         #$df
                adc         sltbit
                sta         sltbit
                rol         sltbit
                iny
                cpy         sltbit+3
                bne         ftrad1
                tya
                asl         a
                asl         a
```

```
                asl             a
                asl             a
                tay
                eor             sltbit
                adc             #$b
                bne             ftrad2
                lda             apple
                rts
ftrad2          lda             #0
                rts
*
installdev      equ             *                       ;made a sub 8/21/86
                php                                     ;and how many drives (carry).
                lda             indrcn+1                ;get index to global device table for
                and             #7                      ;this slot
                asl             a
                tay                                     ; ... into y reg.
                asl             a
                asl             a                       ;now form device number.
                asl             a
                jsr             stadrv
                plp
                ror             a                       ;if 2 drives, then bit 7=1.
                bpl             nodrv2                  ;branch if a 1-drive device (i.e.
profile)
                inx                                     ;else presume that second drive is
present.
                sta             devlst,x
nodrv2          stx             devcnt                  ;save updated device count.
                asl             a                       ;shift # of drives back into carry.
                lda             driveradr               ;get high address of device driver.
                sta             drv1adr,y
                bcc             ndrv2a                  ;branch if single drive
                sta             drv2adr,y
ndrv2a          equ             *
                lda             driveradr+1
                sta             drv1adr+1,y
                bcc             ndrv2b
                sta             drv2adr+1,y
ndrv2b          equ             *
                rts
*
*
* this piece of code (not a subroutine) is branched to if the slot
*  search code finds a smartport device.  it does a smartport status
*  call (code = 0) to determine the number of devices connected to
*  the "card".  it then installs from 0..4 units in the table.
*
smartport       equ             *
                jsr             setdevid                ;set up the devid byte from attributes
*
* only map more than two devices if card is in slot 5
*
                lda             indrcn+1
                sta             driveradr+1             ;didn't set this yet
*cmp #$c5
*bne donesp
*
* do the call to smartport to get the number of devices
*
                lda             driveradr
                skp             1
*****************************************
```

```
* patch 74
*****************************************
                skp          1
                sta          pscall+1                  ;modify operand
                skp          2
                clc
                adc          #3
                sta          spcall+1
                lda          driveradr+1               ;no page cross possible
                sta          spcall+2
                skp          1
*****************************************
* patch 74
*****************************************
                skp          1
                sta          pscall+2                  ;modify operand
                asl                                    ;convert $cn to $n0
                asl                                    ;
                asl                                    ;
                asl                                    ;
                sta          $43                       ;unit number
                lda          #$00                      ;clear a few bytes
                sta          $42                       ;force a prodos status call
                sta          $44                       ;dummy pointer
                sta          $46                       ;number of bytes to transfer
                sta          $47                       ;number of bytes to transfer
                lda          #$10                      ;dummy pointer should be <>0
                sta          $45                       ;dummy pointer
                skp          1
*****************************************
* patch 74
*****************************************
                skp          1
pscall          equ          *                         ;prodos status call
                jsr          $0                        ;filled in by above code
                skp          1
spcall          equ          *
                jsr          $0                        ;filled in by above code
                dfb          $0                        ;this is a status call
                dw           spcallparms
*
* don't add devices if there are none connected
*
                lda          numdevices
                beq          donesp
*
* do the first and second device if it exists
*
                cmp          #2                        ;c set if 2,3,4
                jsr          installdev
*
* do the third and fourth drives if they exist
*  they cannot exist for any card other than one in slot 5
*
                lda          indrcn+1
                cmp          #$c5
                bne          donesp
*
******************** see note #65 *********************
*
* if this is slot 5, and if there is a disk card in slot 2,
*  only install 2 devices this slot.  thank you.
*
```

```
                bit         diskins2                ;is there a disk in slot two?
                bpl         donesp
*
*******************************************************
*
                lda         numdevices
                cmp         #3                      ;c set if 3,4,...
                blt         donesp
                cmp         #4                      ;c set if 4,5,6,...
                lda         #$c2                    ;make it think it's slot 2
                sta         indrcn+1
                jsr         installdev
                lda         #$c5
                sta         indrcn+1
*
donesp          equ         *
                jmp         nxtdsk2                 ;we know it's a disk device
*
*
setdevid        equ         *
                ldy         #$fe
                lda         (indrcn),y
                lsr         a
                lsr         a
                lsr         a
                lsr         a
                sta         devid
                rts
*
*
************** moved from inline ****************
**************   see note #65    ****************
*
* input:   indrcn -  point to $cn00 of mystery card
* output:  carry    clear if disk card here, set ow
*          y        $ff
*
cmpid           equ         *
                lda         relc800                 ;release $c800 space from previous
slot.
                ldy         #$5
cmpid2          lda         (indrcn),y              ;compare id bytes.
                cmp         dskid,y                 ;$cn07=don't care.
                sec
                bne         cibye                   ;$cn05=03
                dey                                 ;$cn03=00
                dey                                 ;$cn01=20
                bpl         cmpid2                  ;loop until all 4 id bytes match.
                clc
cibye           equ         *
                rts
*
***********************************************
*
* smartport parameter area
*
spcallparms     equ         *
                dfb         3                       ;number of parameters
                dfb         0                       ;unit number (code for smartport stat)
                dw          numdevices
                dfb         0                       ;status code = 0 (code for general
status)
*
```

```
*---------------------------------------------------------------
*
*  the original code (previous to the 8/21/86 revision) see note 60
*
*
*
*greet lda $c030
*sta $c00c ;disable 80 columns (rev-e)
*sta $c000 ;disable '80store'
*jsr $fe84
*jsr $fb2f
*jsr $fe93
*jsr $fe89
*cld
** sei ; don't turn interrupts off!! *** see rev note #44 ***
*jsr home ;clear screen
*ldx #slin2-slin1-1 ;move greeting to screen.
*grt1 lda slin1,x ;"apple ii"
*sta $4a8+20-4,x ;$4a8 starts the line.
*dex
*bpl grt1
*ldx #slin3-slin2-1
*grt2 lda slin2,x ;"prodos (version)   (date)"
*sta $5a8+20-grtlz,x ;$5a8 starts the line
*dex
*bpl grt2
*ldx #slin4-slin3-1
*grt3 lda slin3,x ;(blank line)
*sta $6a8+20-grtlzz,x ;$6a8 starts the line
*dex
*bpl grt3
*ldx #grt5-slin4-1
*grt4 lda slin4,x
*sta $7d0,x ;$7d0 starts the line
*dex
*bpl grt4
*lda $c030
*rts
*
*slin1 asc 'apple ii' ;8 chars
*ifeq os-prodos
*slin2 asc 'prodos 8 v1.2a1      '
*fin
*ifeq os-ednet
*slin2 asc 'prodos ednet 1.0x01  '
*fin
*date
*slin3 asc '            '
*slin4 asc 'copyright apple computer, inc., 1983-86' ;39 chars
*grt5 equ *
*grtlz equ slin3-slin2/2
*grtlzz equ slin4-slin3/2
*
*indrcn equ $10
*devid equ $12
*relc800 equ $cfff
*drv1adr equ $bf10
*drv2adr equ $bf20
*lstdev equ $bf30
** devcnt equ $bf31
** devlst equ $bf32
*glbclk equ $bf06
** sltbyt equ $bf99
```

```
*jump equ $4c
*page
*rep 60
*
** this routine finds all disk devices plugged into the
**  system's slots that follow apple's disk id convention.
**  the routine sets up the address and device table in
**  prodos's system global page.  13-sector disk ii's are
**  not configured since they may not function properly in
**  the 16 sector mode...
**
** profiles, and other intelligent devices, are expected to
**  have roms containing the disk i/o drivers.
**
*rep 60
*skp 1
*devsrch ldx #0
*stx dst
*stx dst+1
*stx indrcn ;set up for search.
*dex
*stx devcnt ;device count=none
*lda #$c7 ;search slots from high to low.
*sta indrcn+1
*skp 1
*findsks1 lda relc800 ;release $c800 space from previous slot.
*ldy #$5
*cmpid lda (indrcn),y ;compare id bytes.
*cmp dskid,y ;$cn07=don't care.
*sec
*bne nxtdsk1 ;$cn05=03
*dey ;$cn03=00
*dey ;$cn01=20
*bpl cmpid ;loop until all 4 id bytes match.
*lda (indrcn),y ;check last byte of $cn rom
*beq diskii ;if =00 then 16 sector disk ii.
*cmp #$ff ;if =ff then 13 sector disk ii.
*bcs nxtdsk1 ;ignore if 13 sector boot rom.
*pha ;else, assume it's an intelligent disk for now.
*dey ;get attributes byte.
*lda (indrcn),y
*pha ; & verify it provides read, write, and status calls
*and #$03
*cmp #$03
*beq smart ;branch if it does.
*pla ;else assume it is bozo brand disk...
*pla
*sec
*nxtdsk1 jmp nxtdsk ;branch always taken.
*skp 1
*smart pla ;get attributes again.
*lsr a
*lsr a
*lsr a
*lsr a ;move upper 4 bits to lower 4 bits of the devnum.
*sta devid
*lsr a ;move # of units (0=1, 1=2) to carry.
*lda indrcn+1 ;low entry addr already on stack, save hi addr.
*bne adevice ;branch always taken.
*skp 1
*diskii sta devid ;disk ii's have nul attributes
*lda d2blkio ;save low and hi addr for
*pha ; disk ii block i/o routine.
```

```
*lda d2blkio+1
*sec ;indicate possibly 2 drives per slot.
*adevice pha ;save vector address for now.
*php ;and how many drives (carry).
*lda indrcn+1 ;get index to global device table for
*and #7 ;this slot
*asl a
*tay ; ... into y reg.
*asl a
*asl a ;now form device number.
*asl a
*jsr stadrv
*plp
*ror a ;if 2 drives, then bit 7=1.
*bpl nodrv2 ;branch if a 1-drive device (i.e. profile)
*inx ;else presume that second drive is present.
*sta devlst,x
*nodrv2 stx devcnt ;save updated device count.
*asl a ;shift # of drives back into carry.
*pla ;get high address of device driver.
*sta drv1adr+1,y
*bcc ndrv2a ;branch if single drive
*sta drv2adr+1,y
*pla
*pha
*sta drv2adr,y ;save low address for drive 2.
*ndrv2a pla ;now save low address for drive 1 driver.
*sta drv1adr,y
**
****************** see rev note #53 & #54 **********************
**
*ldy #$7 ; check smartport id rom byte at $cn07...
*lda (indrcn),y ; is it a $0?
*bne notsmartport ; branch if not.
*lda indrcn+1 ; now we are only special casing slot 5!
*cmp #$c5 ; is the smartport card in $c500 space?
*bne notsmartport ; branch if not.
*inx ; get to next unit number in devlst.
*lda #$2b ; deposit unit number $2b(slot 2, drive 1)
*sta devlst,x ; in devlst. (smartport unit 3)
*inx ; bump to next spot in devlst.
*lda #$ab ; deposit unit number $ab(slot 2, drive 2)
*sta devlst,x ; in devlst. (smartport unit 4)
*stx devcnt ; update device count.
*lda #$c5 ; now update hi byte
*sta drv1adr+5 ; of driver entry addresses...
*sta drv2adr+5 ;
*lda drv1adr+10 ; get low driver entry address...
*sta drv1adr+4 ; and update low bytes of
*sta drv2adr+4 ; driver entry addresses...
*notsmartport equ *
*********************************************
* -------------------- see rev note 33 ------------------------
* -------------------- see rev note 24 ----------------------
*lda indrcn+1 ;get slot #
*cmp #$c5 ;only slots 5 & 6 allow drives 3 & 4
*bcc nmdrvs ;  to be mapped into slot (#-4)
*cmp #$c7
*bcs nmdrvs
*lda devid
*and #$2 ;check for 3 or 4 drives
*beq nmdrvs ;branch if not
*lda devid
```

```
*lsr a ;move 4 drive bit into carry
*php ;  and save it.
*lda indrcn+1 ;get index to global device table for
*pha
*and #3 ;  this slot and map into lower slot.
*asl a
*tay
*pla
*and #7 ;use index again to..
*asl a
*asl a
*asl a ;  form device number.
*asl a
*and #$3f
*jsr stadrv
*plp ;get bit showing if 4 drives back
*ror a ;if 4 drives, then bit 7=1.
*bpl nodrv4 ;branch if a 3-drive device
*inx ;else assume that 4th drive is present.
*sta devlst,x ;  and add it to the device list.
*nodrv4 stx devcnt ;save updated device count.
*asl a ;shift # of drives back into carry.
*lda drv1adr+8,y ;get low address from parent slot
*pha
*lda indrcn+1 ;get high address of device driver.
*sta drv1adr+1,y ;save it in down-mapped slot
*bcc ndrv4a ;branch if 3 drives
*sta drv2adr+1,y ;save high address of drive 4
*pla
*pha
*sta drv2adr,y ;save low address for drive 4.
*ndrv4a pla ;now save low address for drive 3
*sta drv1adr,y
*nmdrvs equ *
* ------------------------------------------------------------
*clc
*skp 1
*nxtdsk jsr sltrom ;test for clock & other devices
*skp 1
*dec indrcn+1 ;set up for next lower slot.
*lda indrcn+1 ;have all slots been checked?
*and #7
*beq *+5 ;(to the tay)
*jmp findsks1 ;branch if not.
*skp 1
*tay ;make y=0
*ldx devcnt ;now change the device order
*srtdsk1 lda devlst,x ; so that the boot device
*pha
*and #$7f
*eor lstdev ; will have highest priority
*asl a
*bne srtdsk2
*pla
*iny
*srtdsk2 dex
*bpl srtdsk1
*ldx devcnt ;now reverse order of search, hi to lo.
*tya ;was boot device found?
*beq srtdsk3
*lda lstdev ;make boot device first in search order.
*sta devlst,x
*dex
```

```
*bmi dsrend ;branch if only one device.
*dey ;is this a 2 drive device?
*bmi srtdsk3 ;branch if not.
*eor #$80 ;make boot device, drive 2 next.
*sta devlst,x
*dex
*bmi dsrend ;branch if only one device, 2 drives.
*srtdsk3 pla
*sta devlst,x
*dex
*bpl srtdsk3
*dsrend jsr fndtrd ;save accumuated machine id.
*beq whosit
*sta machid
*rts
*skp 1
*whosit jmp whatsit
*skp 1
*stadrv equ *
*ora devid ;combine with attributes.
*ldx devcnt
*inx ;put device number into device list.
*sta devlst,x
*asl a ;now form drive 2 device number, if any.
*rts
*skp 1
*sltrom bcc isrom ;branch if disk drive.
*ldy #6 ;test this slot for clock card.
*clkchk lda (indrcn),y
*cmp clkid,y
*bne notclk ;branch if not clock.
*dey
*dey
*bpl clkchk
*lda indrcn+1 ;transfer high slot addr minus $c1 (default)
*sbc #$c1 ; to relocate references to clock rom.
*sta clock64
*lda #jump ;also enable jump vector in globals.
*sta glbclk
*lda apple ;mark clock as present
*beq dsrend
*ora #1
*sta apple
*bne isrom ;branch always taken.
*skp 1
*notclk ldy #5 ;test for 80 col card.
*lda (indrcn),y ; byte at a time.
*cmp #$38
*bne notcons
*ldy #7 ;test values are same as pascal's
*lda (indrcn),y
*cmp #$18
*bne notcons
*ldy #$b
*lda (indrcn),y
*cmp #1
*bne notcons
*iny
*lda (indrcn),y
*and #$f0 ;mask off low nibble
*cmp #$80 ;generic for 80-col card
*bne notcons
*lda apple
```

```
*beq dsrend
*ora #2
*sta apple ;mark config for 80 col. present.
*bne isrom ;branch always taken.
*skp 1
*notcons ldy #0 ;test for any rom.
*ldx #0
*lda (indrcn),y
*cmp #$ff ;test for apple iii non slot.
*beq norom ;branch if invalid rom.
*tstrom cmp (indrcn),y ;look for floating bus.
*bne norom
*inx ;loop 256 times.
*bne tstrom
*isrom lda indrcn+1 ;mark a bit in slot byte
*and #$7 ; to indicate rom present.
*tax
*lda sltbit,x
*ora sltbyt
*sta sltbyt
*norom rts
*skp 1
*d2blkio dw rwts
*dskid equ *
*clkid dfb $08,$20,$28,$00,$58,$03,$70,$3c
*sltbit dfb $00,$02,$04,$08,$10,$20,$40,$80
*skp 1
*fndtrd clc
*ldy sltbit
*ftrad1 lda (look),y
*and #$df
*adc sltbit
*sta sltbit
*rol sltbit
*iny
*cpy sltbit+3
*bne ftrad1
*tya
*asl a
*asl a
*asl a
*asl a
*tay
*eor sltbit
*adc #$b
*bne ftrad2
*lda apple
*rts
*ftrad2 lda #0
*rts
v

; ###############################################################################################
; #   END OF FILE:  DEVSRCH
; #   LINES      :  975
; #   CHARACTERS :  34131
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.DOATALK.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  DOATALK
; ###############################################################################################

* note: this file was added at time of rev note #en2.
*
                page
*
atcreate        equ         *
atdestroy       equ         *
atrename        equ         *
atsetinfo       equ         *
atgetinfo       equ         *
atonline        equ         *
atsetprefx      equ         *
atgetprefx      equ         *
atopen          equ         *
atnewline       equ         *
atread          equ         *
atwrite         equ         *
atclose         equ         *
atflush         equ         *
atsetmark       equ         *
atgetmark       equ         *
atseteof        equ         *
atgeteof        equ         *
atsetbuf        equ         *
atgetbuf        equ         *
*
                ldx         #0                      ;
display         lda         msg,x                   ;
                sta         $400,x
                beq         done
                inx                                 ;
                bne         display
done            rts
*
                msb         on
msg             asc         'this command is for /atalk!!!'
                db          0
                msb         off
                ds          $f800-*,0               ; fill remaining space with zeros.
here            equ         *

; ###############################################################################################
; #   END OF FILE:  DOATALK
; #   LINES      :  39
; #   CHARACTERS :  1240
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.EQUATES.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  EQUATES
; ###############################################################################################

                sbtl        'prodos machine language interface'
dispadr         equ         $1000                   ;dispatcher load address
rwts            equ         $d000                   ;disk driver entry
                ifeq        os-prodos
orig            equ         $d700
orig1           equ         $de00
                fin
                ifeq        os-ednet
orig            equ         $f800                   ;start of buffers ** see rev note #en1
**
orig1           equ         $d000                   ;mli code starting address * #en1 *
                fin
globals         equ         $bf00
inthandler      equ         $ff9b                   ; start of interrupt handler.
network         equ         $d400                   ;applebus driver (main lc - bank 2)
pathbuf         equ         orig
fcb             equ         orig+$100               ;file control block
vcb             equ         orig+$200               ;volume control block
bmbuf           equ         orig+$300               ;bit map buffer
gbuf            equ         orig+$500               ;general buffer
*
*
                ifeq        os-ednet
*
*  load addresses in build.ednet basic program
*
mli.0           equ         $2000                   ;mli loader/relocater
ram.1           equ         mli.0+$b00              ;$2b00 - installer for /ram
ram.0           equ         ram.1+$100              ;$2c00 - /ram driver in aux mem
ram.2           equ         ram.0+$200              ;$2e00 - /ram driver in main lc.
mli.2           equ         ram.2+$100              ;$2f00 - mli
mli.1           equ         mli.2+$2800             ;$5700 - global page
tclock.0        equ         mli.1+$100              ;$5800 - tclock driver
mli.3           equ         tclock.0+$100           ;$5900 - interrupts
xrw.0           equ         mli.3+$100              ;$5a00 - disk core routines
sel.0           equ         xrw.0+$700              ;$6100 - dispatcher
                fin
*
                ifeq        os-prodos
*
*  load addresses in build.prodos basic program
*
mli.0           equ         $2000                   ;mli loader/relocater
ram.1           equ         mli.0+$b00              ;$2b00 - installer for /ram
ram.0           equ         ram.1+$100              ;$2b00 - /ram driver in aux mem
ram.2           equ         ram.0+$200              ;$2d00 - /ram driver in main lc.
mli.2           equ         ram.2+$100              ;$2e00 - mli
mli.1           equ         mli.2+$2100             ;$4f00 - global page
tclock.0        equ         mli.1+$100              ;$5000 - tclock driver
mli.3           equ         tclock.0+$9b            ;$509b - interrupts
xrw.0           equ         tclock.0+$100           ;$5100 - disk core routines
sel.0           equ         xrw.0+$700              ;$5800 - dispatcher
*
```

```
********* see rev note #50 *********
*
cclock.0        equ          sel.0+$300               ; $5b00 - cortland clock driver
***********************************
* -------------------- see rev note 20 and 50 -----------------
*bus.0 equ sel.0+$300 ;$5b00 - applebus driver
* -------------------------------------------------------
                fin
*
tclk.in         equ          pathbuf+$42              ;clock driver execution address
*
* standard error codes
*
badscnum        equ          $01                      ;bad system call #
badscpcnt       equ          $04                      ;bad call parameter count
intblful        equ          $25                      ;interupt table full.
xioerr          equ          $27
xnodrv          equ          $28                      ;no device connected.
xnowrite        equ          $2b                      ;disk/media is hardware write protected
dswtched        equ          $2e                      ;wrong volume is mounted on active
device.
devoffline      equ          $2f                      ; device off line, or no disk in drive.
badpath         equ          $40                      ;invalid pathname syntax
fcbfull         equ          $42                      ;file control block full
badrefnum       equ          $43                      ;invalid refnum
vnferr          equ          $45                      ;volume not found
pathnotfnd      equ          $44                      ;pathname not found
fnferr          equ          $46                      ;file not found
duperr          equ          $47                      ;duplicate file name error
ovrerr          equ          $48                      ;not enough disk space for
preallocation
dirfull         equ          $49                      ;directory full error
cpterr          equ          $4a                      ;file incompatible sos version
typerr          equ          $4b                      ;not currently supported storage type
eoferr          equ          $4c                      ;position attempted beyond end of file
posnerr         equ          $4d                      ;illegal position (l.t. 0 or g.t.
$ffffff)
accserr         equ          $4e                      ;file access r/w request conflicts with
attributes.
filbusy         equ          $50                      ;either write was requested or write
access already allocated.
direrr          equ          $51                      ; directory entry count inconsistent
with actual entries
notsos          equ          $52                      ;not an xdos diskette
badlstcnt       equ          $53                      ;invalid value in list parameter
vcbfull         equ          $55                      ;too many devices...
badbuf          equ          $56
dupvol          equ          $57                      ;duplicate volume can't be logged in.
bitmapadr       equ          $5a                      ; bit map disk address impossible
*
* fatal errors
*
badirq          equ          $01                      ;unclaimed interrupt
vcberr          equ          $0a                      ;volume control block not usable
fcberr          equ          $0b
alcerr          equ          $0c                      ;allocation blocks invalid
                page
*
* constants
*
seedtyp         equ          1
saptyp          equ          2
tretyp          equ          3
```

```
dirtyp          equ          $d
hedtyp          equ          $e
rdcmd           equ          $1
wrtcmd          equ          $2
statcmd         equ          $00                    ;request status, no error=ready
pretime         equ          $20                    ;command needs current date/time stamp
preref          equ          $40                    ;command requires fcb address and
verification
prepath         equ          $80                    ;command has pathname to preprocess
*
* volume status constants (bits)
*
*
* file status constants
*
datalc          equ          $1                     ;data block not allocated.
idxalc          equ          $2                     ;index not allocated
topalc          equ          $4                     ;top index not allocated
stpmod          equ          $8                     ;storage type modified
usemod          equ          $10                    ;file usage modified
eofmod          equ          $20                    ;end of file modified
datmod          equ          $40                    ;data block modified
idxmod          equ          $80                    ;index block modified
fcbmod          equ          $80                    ; has fcb/directory been modified?
(flush)
*
* file attributes constants
*
readen          equ          $1                     ;read enabled
writen          equ          $2                     ;write enabled
renamen         equ          $40                    ;rename ok when on.
dstroyen        equ          $80                    ;destroy ok when on.
                page
* header index constants
*
hnlen           equ          $0                     ;header name length (offset into
header)
*hname equ $1 ; header name
hpenab          equ          $10                    ;password enable byte
hpass           equ          $11                    ;encoded password
hcrdt           equ          $18                    ;header creation date
* hcrtm equ $1a ;header creation time
hver            equ          $1c                    ;sos version that created directory
hcmp            equ          $1d                    ;backward compatible with sos version
hattr           equ          $1e                    ;header attributes- protect etc.
* hentln equ $1f ;length of each entry
* hment equ $20 ;maximum number of entries/block
hcent           equ          $21                    ;current number of files in directory
hrblk           equ          $23                    ;owner's directory address
hrent           equ          $25                    ;owner's directory entry number
hreln           equ          $26                    ;owner's directory entry length
vbmap           equ          hrblk
vtblk           equ          hrent                  ;(used for root directory only)
*
* volume control block index constants
*
vcbsize         equ          $20                    ;current vcb is 32 bytes per entry (ver
0)
vcbnml          equ          0                      ;volume name length byte
vcbnam          equ          1                      ;volume name
vcbdev          equ          $10                    ;volume's device
vcbstat         equ          $11                    ;volume status. (80=files open. 40=disk
switched.)
```

```
vcbtblk         equ          $12                       ;total blocks on this volume
vcbtfre         equ          $14                       ;number of unused blocks
vcbroot         equ          $16                       ;root directory (disk) address
*vcbmorg equ $18 ; map organization (not supported by v 0)
*vcbmbuf equ $19 ; bit map buf num
vcbdmap         equ          $1a                       ;first (disk) address of bitmap(s)
vcbcmap         equ          $1c                       ;relative address of bit map with space
(add to vcbdmap)
*vcbmnum equ $1d ; relative bit map currently in memory
vcbopnc         equ          $1e                       ;current number of open files.
*vcb addr $1f reserved
*
* file control block index constants
*
fcbrefn         equ          0                         ;file reference number (position
sensitive)
fcbdevn         equ          1                         ;device (number) on which file resides
*fcbhead equ 2 ; block address of file's directory header
*fcbdirb equ 4 ; block address of file's directory
fcbentn         equ          6                         ;entry number within directory block
fcbstyp         equ          7                         ;storage type - seed, sapling, tree,
etc.
fcbstat         equ          8                         ;status - index/data/eof/usage/type
modified.
fcbattr         equ          9                         ;attributes - read/write enable,
newline enable.
fcbnewl         equ          $a                        ;new line terminator (all 8 bits
significant).
fcbfbuf         equ          $b                        ;buffer number
fcbfrst         equ          $c                        ;first block of file
fcbidxb         equ          $e                        ;block address of index (0 if no index)
fcbdatb         equ          $10                       ;block address of data
fcbmark         equ          $12                       ;current file marker.
fcbeof          equ          $15                       ;logical end of file.
fcbuse          equ          $18                       ;actual number of blocks allocated to
this file.
*fcb addr $1a reserved
fcblevl         equ          $1b                       ; level at which this file was opened
fcbdirty        equ          $1c                       ; fcb marked as modified
fcbnlmsk        equ          $1f
                page
*
* zero page stuff
*
par             equ          $40
device          equ          par+2
dhpcmd          equ          device
unitnum         equ          device+1
dbufpl          equ          device+2
dbufph          equ          dbufpl+1
bloknml         equ          device+4
bloknmh         equ          bloknml+1
*
intcmd          equ          dhpcmd
*
ztemps          equ          par+8
tpath           equ          ztemps
drbufpl         equ          ztemps
drbufph         equ          drbufpl+1
tindx           equ          ztemps
datptr          equ          ztemps+2
posptr          equ          ztemps+4
usrbuf          equ          ztemps+6
```

```
*
* xdos parameters:
*
c.pcnt          equ         $0                      ; (count)
c.devnum        equ         $1                      ; (value)
c.refnum        equ         $1                      ; (value)
c.intnum        equ         $1                      ; (value)
c.path          equ         $1                      ;&2 (pointer)
c.isnewl        equ         $2                      ; (mask)
c.datbuf        equ         $2                      ;&3 (value)
c.bufadr        equ         $2                      ;&3 (address)
c.intadr        equ         $2                      ;&3 (address)
c.mark          equ         $2                      ;->4 (value)
c.eof           equ         $2                      ;->4 (value)
c.attr          equ         $3                      ; (flags)
c.newl          equ         $3                      ; (character)
c.bufptr        equ         $3                      ;&4 (pointer)
c.nwpath        equ         $3                      ;&4 (pointer)
c.filid         equ         $4                      ; (value)
c.reqcnt        equ         $4                      ;&5 (value)
c.blknum        equ         $4                      ;&5 (address)
c.outref        equ         $5
c.auxid         equ         $5                      ;&6 (value)
c.trnscnt       equ         $6                      ;&7 (value)
c.fkind         equ         $7                      ; (value)
c.date          equ         $8                      ;&9 (value)
c.outblk        equ         $8                      ;&9 (count)
c.time          equ         $a                      ;&b (value)
c.moddate       equ         $a                      ;&b (value)
c.modtime       equ         $c                      ;&d (value)
c.credate       equ         $e                      ;&f (value)
c.cretime       equ         $10                     ;&11 (value)
*
altram          equ         $c083
irqvect         equ         $fffe                   ;location of irq vector
romirq          equ         $fa41                   ;monitor rom irq entry
m.break         equ         $fa59                   ;monitor break vector
m.reset         equ         $ff59                   ;monitor reset entry
m.and           equ         m.reset                 ;& vector
m.ctl.y         equ         m.reset                 ;control y vector
m.nmi           equ         m.reset                 ;nmi vector
*
rtbl            equ         $10
src             equ         $12
dst             equ         $14
cnt             equ         $16
cde             equ         $18
ecde            equ         $1a
look            equ         $0a                     ;must go at $000a!!
apple           equ         $0c
unit            equ         $43
home            equ         $fc58
gopro           equ         $bf00
lodintrp        equ         $800
pbuf            equ         $280
reset           equ         $3f2
name            equ         pbuf
prefix          equ         $c6
onlyne          equ         $c5                     ;mli online call #
iopen           equ         $c8
igeof           equ         $d1
iread           equ         $ca
iclos           equ         $cc
```

```
getfileinfo     equ          $c4                       ; prodos get file info call number.
idbrd           equ          $80
*
* soft switches
*
ramin           equ          $c08b
ramin2          equ          $c083
romin           equ          $c082                     ;swap rom in, write protect ram
romin1          equ          $c081                     ;swap rom in w/o w-prot ram
slot3rom        equ          $c00b                     ;write location to switch in slot 3 rom
read3rom        equ          $c017                     ; bit 7 =1 if slot space 3 enabled
int3rom         equ          $c00a                     ;write location to switch in internal 3
rom

; ###############################################################################################
; #   END OF FILE:  EQUATES
; #   LINES      :  298
; #   CHARACTERS :  16329
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.FNDFIL.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  FNDFIL
; ###############################################################################################

               page
*
*
findfile       jsr         lookfile                ;see if file exists
               bcs         nofind                  ;branch if an error was encountered
moventry       ldy         h.entln                 ;move entire entry info to a safe area
movent1        lda         (drbufpl),y
               sta         dfil+d.stor,y
               dey
               bpl         movent1
               lda         #0                      ;to indicate all is well
nofind         rts                                 ;return condition codes.
               page
*
lookfile       jsr         preproot                ;find volume and set up other boring
stuff
               bcs         fnderr                  ;pass back any error encountered
               bne         lookfil0                ;branch if more than root.
               lda         #<gbuf                  ; otherwise, report a badpath error
               sta         drbufph                 ;(but first create a phantom entry for
open)
               lda         #4
               sta         drbufpl
               ldy         #d.auxid                ;first move in id, and date stuff.
phantm1        lda         (drbufpl),y
               sta         dfil,y
               dey
               cpy         #d.credt-1
               bne         phantm1
phantm2        lda         rootstuf-d.filid,y
               sta         dfil,y
               dey
               cpy         #d.filid-1
               bne         phantm2
               lda         #dirtyp*$10             ;fake directory file
               sta         dfil+d.stor
               lda         #badpath                ;(carry is set)
               rts
*
*
lookfil0       lda         #0                      ;reset free entry indicator
               sta         nofree
               sec                                 ;indicate that the directory to be
searched has header in this block
lookfil1       lda         #0                      ;reset entry counter
               sta         totent
               jsr         looknam                 ;look for name pointed to by pnptr
               bcc         namfound                ;branch if name was found.
               lda         entcntl                 ;have we looked at all of the
               sbc         totent                  ; entries in this directory?
               bcc         dcrenth                 ;maybe, check hi count.
               bne         lookfil2                ;no, read next directory block
```

```
                cmp         entcnth                 ;has the last entry been looked at
(acc=0)
                beq         errfnf                  ;yes, give 'file not found' error.
                bne         lookfil2                ;branch always.
dcrenth         dec         entcnth                 ;should be at least 1
                bpl         lookfil2                ;(this should be branch always...)
errdir          lda         #direrr                 ;report directory messed up.
fnderr          sec                                 ;indicate error has been encountered.
                rts
*
                page
lookfil2        sta         entcntl                 ;keep running count
                lda         #<gbuf                  ;reset indirect pointer
                sta         drbufph
                lda         gbuf+2                  ;get link to next directory block
                bne         nxtdir0                 ;(if there is one)
                cmp         gbuf+3                  ;are both zero, i.e. no link?
                beq         errdir                  ;if so, then not all entries were
accounted for.
nxtdir0         equ         *                       ;acc has value for block # (low)
                ldx         gbuf+3
                jsr         rdblk                   ;go read the next linked directory in.
                bcc         lookfil1                ;branch if no error.
                rts                                 ;return error (in accumulator).
*
*
errfnf          lda         nofree                  ;was any free entry found?
                bne         fnf0
                lda         gbuf+2                  ;test link
                bne         telfree
                cmp         gbuf+3                  ;if both are zero, then give up
                beq         fnf0                    ; simply report 'not found'
telfree         sta         d.entblk
                lda         gbuf+3
                sta         d.entblk+1              ;assume first entry of next block
                lda         #1                      ; is free for use.
                sta         d.entnum
                sta         nofree                  ; mark d.entnum as valid (for create)
fnf0            jsr         nxtpnam1                ;test for 'file not found' versus 'path
not found'
errpath1        sec                                 ;if non-zero then 'path not found'.
                beq         fnf1
                lda         #pathnotfnd             ;report no such path.
                rts
fnf1            lda         #fnferr                 ;report file not found.
                rts
                page
*
namfound        jsr         nxtpname                ;adjust index to next name in path.
                beq         filfound                ;branch if that was last name.
                ldy         #d.stor                 ;be sure this is a directory entry
                lda         (drbufpl),y             ;high nibble will tell
                and         #$f0
                cmp         #dirtyp*16              ;is it a sub-directory?
                bne         errpath1                ;report the user's mistake
                ldy         #d.frst                 ;get address of first sub-directory
block
                lda         (drbufpl),y
                sta         bloknml                 ;(no checking is done here for a valid
                iny                                 ; block number... )
                sta         d.head                  ;save as file's header block too.
                lda         (drbufpl),y
                sta         bloknmh
```

```
                sta     d.head+1
                jsr     rdgbuf                  ;read sub-directory into gbuf
                bcs     fnderr1                 ;return immediately any error
encountered.
                lda     gbuf+hcent+4            ;get the number of files
                sta     entcntl                 ; contained in this directory
                lda     gbuf+hcent+5
                sta     entcnth
                lda     gbuf+hcmp+4             ;test backward compatability
                bne     errcomp                 ;if not, don't proceed
                lda     gbuf+hpenab+4           ;make sure password disabled
                ldx     #0
                sec
                rol     a
tstpas0         bcc     tstpas1
                inx
tstpas1         asl     a
                bne     tstpas0
                cpx     #5                      ;is password disabled?
                beq     movhead
*
errcomp         lda     #cpterr                 ;tell them this directory is not
compatable
fnderr1         sec
                rts
*
movhead         jsr     movhed0                 ;move info about this directory
                jmp     lookfil0                ;do next local pathname
*
movhed0         ldx     #$a                     ;move info about this directory
movhed1         lda     gbuf+hcrdt+4,x
                sta     h.credt,x
                dex
                bpl     movhed1
                lda     gbuf+4                  ;if this is root, then nothing to do.
                and     #$f0
                eor     #$f0                    ;test header type.
                beq     mvhed3                  ;branch if root.
                ldx     #3                      ;otherwise, save owner info about this
header.
mvhed2          lda     gbuf+hrblk+4,x
                sta     own.blok,x
                dex
                bpl     mvhed2
mvhed3          rts
*
                page
*
*
filfound        equ     *
entadr          lda     h.maxent                ;figure out which is entry number this
is.
                sec
                sbc     cntent                  ;max entries - count entries + 1 =
entry number
                adc     #0                      ;(carry is/was set)
                sta     d.entnum
                lda     bloknml
                sta     d.entblk
                lda     bloknmh                 ;and indicate block number of this
directory.
                sta     d.entblk+1
                clc
```

```
                rts
*
looknam         lda         h.maxent                ;reset count of files per block
                sta         cntent
                lda         #<gbuf
                sta         drbufph
                lda         #4
loknam1         sta         drbufpl                 ;reset indirect pointer to gbuf
                bcs         loknam2                 ;branch if this block contains a header
                ldy         #d.stor
                lda         (drbufpl),y             ;get length of name in directory
                bne         isname                  ;branch if there is a name.
                lda         nofree                  ;test to see if a free entry has been
declared.
                bne         loknam2                 ;yes bump to next entry
                jsr         entadr                  ;set address for current entry
                inc         nofree                  ;indicate a free spot has been found
                bne         loknam2                 ;branch always.
*
isname          and         #$f                     ;strip type (this is checked by
'filfound')
                inc         totent                  ;(bump count of valid files found)
                sta         namcnt                  ;save name length as counter
                ldx         pnptr                   ;get index to current path.
                cmp         pathbuf,x               ;are both names of the same length?
                bne         loknam2                 ;no, bump to next entry
*
cmpname         inx                                 ;(first) next letter index
                iny
                lda         (drbufpl),y             ;compare names letter by letter
                cmp         pathbuf,x
                bne         loknam2
                dec         namcnt                  ;have all letters been compared?
                bne         cmpname                 ;no, continue..
                clc                                 ;by golly, we got us a match!
noname          rts
*
loknam2         dec         cntent                  ;have we checked all possible entries
in this block?
                sec
                beq         noname                  ;yes, give up.
                lda         h.entln                 ;add entry length to current pointer
                clc
                adc         drbufpl
                bcc         loknam1                 ;branch if we're still in the first
page.
                inc         drbufph                 ;look on second page
                clc                                 ;carry should always be clear before
looking at next.
                bcc         loknam1                 ;branch always...
                page

; ###############################################################################################
; #   END OF FILE:  FNDFIL
; #   LINES      :  211
; #   CHARACTERS :  11829
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.GLOBALS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  GLOBALS
; ###############################################################################################

                sbtl         'global pages - 64k'
                org          globals
*
entry           jmp          mlient1                  ;mli call entry point
*
************ see rev note #36 ************************
*
*jspare jmp sys.rts ;jump vector to cold start/selector program, etc.
jspare          jmp          jspare                   ; will be changed to point to
dispatcher
*                               caller by the loader.
*****************************************************
datetime        dfb          $60                      ; changed to $4c (jmp) if clock
present.
                dw           tclk.in                  ; clock routine entry address.
syserr          jmp          syserr1                  ;error reporting hook.
sysdeath        jmp          sysdeath1                ;system failure hook.
serr            dfb          $00                      ;error code, 0=no error.
                skp          1
*
* device driver vectors.
*
devadr01        dw           gnodev                   ;slot zero reserved
                dw           gnodev                   ;slot 1, drive 1
                dw           gnodev                   ;slot 2, drive 1
                dw           gnodev                   ;slot 3, drive 1
                dw           gnodev                   ;slot 4, drive 1
                dw           gnodev                   ;slot 5, drive 1
                dw           gnodev                   ;slot 6, drive 1
                dw           gnodev                   ;slot 7, drive 1
                dw           gnodev                   ;slot zero reserved
                dw           gnodev                   ;slot 1, drive 2
                dw           gnodev                   ;slot 2, drive 2
                dw           gnodev                   ;slot 3, drive 2
                dw           gnodev                   ;slot 4, drive 2
                dw           gnodev                   ;slot 5, drive 2
                dw           gnodev                   ;slot 6, drive 2
                dw           gnodev                   ;slot 7, drive 2
                skp          1
*
* configured device list by device number
* access order is last in list first.
*
devnum          dfb          $00                      ;most recent accessed device.
devcnt          dfb          $ff                      ;number of on-line devices (minus 1).
devlst          dfb          $0,0,0,0                 ;up to 14 units may be active.
                dfb          0,0,0,0,0
                dfb          0,0,0,0,0
                skp          1
                ifeq         os-prodos
                asc          "(C)APPLE'83" ; AppleTALK writes over this area! DO NOT MOVE!"
                fin
mlient1         php
```

```
                sei
                jmp         mlicont
aftirq          sta         ramin
                jmp         fix45                   ;restore $45 after interrupt in lang
card
*
****** see rev note #en3 ******
*
* note: this code was moved down here so a branch could reach gorpm.
*
                ifeq        os-ednet
rpm.entry       equ         *
                php                                 ; needs 2 since bridge eats one of 'em.
                php                                 ; done since we return via rti.
                pha                                 ; save outgoing character on stack.
                bit         $ff58                   ; set v flag for rpm.
                bvs         rpmcont                 ; branch always...
gorpm           jmp         rpm.stub                ; call rpm.
                fin
*******************************
*
                ds          $bf56-*,0               ; got to align this stuff!
*
old45           dfb         0
afbank          dfb         0
                skp         1
*
* memory map of the lower 48k.  each bit represents one page
* (256 bytes) of memory.  protected areas are marked with a
* 1, unprotected with a 0.  prodos dis-allows reading or
* buffer allocation in protected areas.
*
memtabl         dfb         $c0,$00,$00,$00,$00,$00,$00,$00
                dfb         $00,$00,$00,$00,$00,$00,$00,$00
                dfb         $00,$00,$00,$00,$00,$00,$00,$01
                page
*
* the addresses contained in this table are buffer addresses
* for currently open files.  these are informational only,
* and should not be changed by the user except through the
* mli call setbuf.
*
gl.buff         dw          $0000                   ;file number 1
                dw          $0000                   ;file number 2
                dw          $0000                   ;file number 3
                dw          $0000                   ;file number 4
                dw          $0000                   ;file number 5
                dw          $0000                   ;file number 6
                dw          $0000                   ;file number 7
                dw          $0000                   ;file number 8
                skp         1
*
* interrupt vectors are stored here.  again, this area is
* informational only, and should be changed only by calls
* to the mli to allocate_interrupt.  values of the a, x, y,
* stack, and status registers at the time of the most recent
* interrupt are also stored here.  in addition, the address
* interrupted is also preserved.  these may be used for
* performance studies and debugging, but should not be changed
* by the user.
*
intrupt1        dw          $0000                   ;interupt routine 1
intrupt2        dw          $0000                   ;interupt routine 2
```

```
intrupt3        dw          $0000                   ;interupt routine 3
intrupt4        dw          $0000                   ;interupt routine 4
intareg         dfb         $00                     ;a-register
intxreg         dfb         $00                     ;x-register
intyreg         dfb         $00                     ;y-register
intsreg         dfb         $00                     ;stack register
intpreg         dfb         $00                     ;status register
intbankid       dfb         $01                     ;rom, ram1, or ram2 ($d000 in lc)
intaddr         dw          $0000                   ;program counter return addr.
                skp         1
*
* the user may change the following options prior
* to calls to the mli.
*
datelo          dw          $0000                   ;bits 15-9=yr, 8-5=mo, 4-0=day
timelo          dw          $0000                   ;bits 12-8=hr, 5-0=min; low-hi format.
level           dfb         $00                     ;file level: used in open, flush,
close.
bubit           dfb         $00                     ;backup bit disable, setfileinfo only.
spare1          dfb         $00                     ; Used to save A reg
newpfxptr       dfb         $00                     ; used as AppleTalk alternate prefix
ptr
                page
*
* the following are informational only.  machid identifies
* the system attributes:
* (bit 3 off) bits 7,6-  00=ii  01=ii+   10=iie   11=/// emulation
* (bit 3 on)  bits 7,6-  00=na  01=na    10=//c   11=na
*             bits 5,4-  00=na  01=48k   10=64k   11=128k
*             bit  3    modifier for machid bits 7,6.
*             bit  2    reserved for future definition.
*             bit  1=1-  80 column card
*             bit  0=1-  recognizable clock card
*
* sltbyt indicates which slots are determined to have roms.
* pfixptr indicates an active prefix if it is non-zero.
* mliactv indicates an mli call in progress if it is non-zero.
* cmdadr is the address of the last mli call's parameter list.
* savx and savy are the values of x and y when the mli
*  was last called.
*
machid          dfb         $00                     ;machine identification.
sltbyt          dfb         $00                     ;'1' bits indicate rom in slot(bit#)
pfixptr         dfb         $00                     ;if = 0, no prefix active..
mliactv         dfb         $00                     ;if <> 0, mli call in progress
cmdadr          dw          $0000                   ;return address of last call to mli.
savex           dfb         $00                     ;x-reg on entry to mli
savey           dfb         $00                     ;y-reg on entry to mli
                skp         1
*
* the following space is reserved for language card bank
* switching routines.  all routines and addresses are
* subject to change at any time without notice and will,
* in fact, vary with system configuration.
* the routines presented here are for 64k systems only.
*
                skp         1
exit            eor         $e000                   ;test for rom enable.
                beq         exit1                   ;branch if ram enabled.
                sta         romin                   ;else enable rom and return.
                bne         exit2                   ;branch always
                skp         1
exit1           lda         bnkbyt2                 ;for alternate ram (mod by mlient1)
```

```
                eor         $d000                   ; enable.
                beq         exit2                   ;branch if not alternate ram.
                lda         altram                  ;else enable alt $d000
exit2           pla                                 ;restore return code.
                rti                                 ;re-enable interupts and return
                skp         1
mlicont         sec
                ror         mliactv                 ;indicate to interupt routines mli
active.
*
***** see rev note #en3 *******
*
                ifeq        os-ednet
                clv                                 ; overflow flag must be cleared for
standard mli call.
                fin
******************************
rpmcont         lda         $e000                   ;preserve language card / rom
                sta         bnkbyt1                 ; orientation for proper
                lda         $d000                   ; restoration when mli exits...
                sta         bnkbyt2
                lda         ramin                   ;now force ram card on
                lda         ramin                   ; with ram write allowed.
*
*********** see rev note #en3 ************
*
                ifeq        os-ednet
                bvs         gorpm                   ; if v set go do rpm.
                fin
****************************************
                jmp         entrymli
                page
*
* interupt exit/entry routines
*
                skp         1
irqxit          lda         intbankid               ;determine state of ram card
irqxit0         beq         irqxit2                 ; if any.  branch if enabled.
                bmi         irqxit1                 ;branch if alternate $d000 enabled.
                lsr         a                       ;determine if no ram card present.
                bcc         romxit                  ;branch if rom only system.
                lda         romin1                  ;else enable rom first.
                bcs         romxit                  ;branch always taken...
irqxit1         lda         altram                  ;enable alternate $d000.
irqxit2         lda         #1                      ;preset bankid for rom.
                sta         intbankid               ;(reset if ram card interupt)
romxit          lda         intareg                 ;restore accumulator...
                rti                                 ; and exit!
                skp         1
irqent          bit         ramin                   ;this entry only used when rom
                bit         ramin                   ; was enabled at time of interupt.
                jmp         irqrecev                ; a-reg is stored at $45 in zpage.
bnkbyt1         dfb         $00
bnkbyt2         dfb         $00
                skp         1
*
*************** see rev note #36 **********************
*
*sys.rts bit ramin ;make certain language card is switched in
* jmp sys.end ;or anywhere else we need to go
*
*****************************************************
*
```

```
*
*
                ds          $bffc-*,0               ; pad
ibakver         dfb         $00                     ; reserved
iversion        dfb         $00                     ;version # of currently running
interpreter
                skp         2
kbakver         dfb         $00                     ;undefined: reserved for future use
kversion        dfb         $07                     ; Represents release 1.7
;

; ###############################################################################################
; #   END OF FILE:  GLOBALS
; #   LINES      :  244
; #   CHARACTERS :  12535
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.INTERRUPT.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  INTERRUPT
; ###############################################################################################

inthandler      equ         *                       ; interrupt handler
                ifne        opers-prodos
                cld
                fin
                lda         #0
                sta         statparm+1
                ifne        opers-pascal
                ldy         #<statparm
                ldx         #>statparm
                else
                ldy         adrstatparm+1
                ldx         adrstatparm
                fin
                jsr         applebus                ; is interrupt from us ?
                bcs         srcsckt9                ; not ours, just return
                lda         status
                pha
                ldx         #zpsize-1               ; save zero page
                lda         cmndlist,x
                sta         savezp,x
                dex
                bpl         *-6
                ifne        opers-prodos
                stx         intrflag                ; indicates we are in interrupt
                fin
                lda         statparm+1              ; what kind of interrupt ?
                and         #$0f                    ; get packet bits
                sta         pcktmap                 ; number of packets
nxtpckt         lsr         pcktmap                 ; have we read them all ?
                bcs         rdpckt                  ; no
                bne         nxtpckt                 ; try again if some bit are still set
chktimer        bit         statparm+1
                bvc         intrpdone
                jsr         atptimeout
                jsr         nbptimeout
                dec         bridgetime
                bne         chkusrtime
                dec         bridgetime+1
                bne         chkusrtime
                lda         #0
                sta         a.bridge
chkusrtime      equ         *
                lda         usertimer+1             ; does user want a timer ?
                beq         *+5                     ; no
usertimer       equ         *+1
                jsr         $0000
intrpdone       ldx         #zpsize-1               ; restore zero page
                lda         savezp,x
                sta         cmndlist,x
                dex
                bpl         *-6
                ifne        opers-prodos
                inx
```

```
                stx         intrflag                ; indicates we are outside interrupt
                fin
                pla
                sta         status                  ; restore status
                clc
                rts
;
rdpckt          equ         *                       ; read the lap part
                ifne        opers-pascal
                ldy         #<rlapparm
                ldx         #>rlapparm
                else
                ldy         adrrlapparm+1
                ldx         adrrlapparm
                fin
                jsr         readheader
                bne         nxtpckt                 ; packete too short, garbage
                lda         laptype                 ; what type ?
                tax
                dex
                beq         short
                dex
                beq         long
                ldx         prtclhandler+1          ; do we have a protocol handler ?
                sec
                beq         aftprthandler           ; no, since carry is set, will be
discarded
                ldx         lapsrce                 ; pass source node in x reg, lap type
in acc
prtclhandler    equ         *+1
                jsr         $0000
aftprthandler   jmp         chkreject               ; discard packet if carry flag is set
;
short           lda         #5
                sta         ddphsize
                ifne        opers-pascal
                ldy         #<rddpparm
                ldx         #>rddpparm
                else
                ldy         adrrddpparm+1
                ldx         adrrddpparm
                fin
                jsr         readheader              ; read the ddp header
                bne         nxtpckt
                do          savespace
                ldy         #move1-movetable
                jsr         movedata
                beq         readddp                 ; since movedata set z flag, bra
                else
                lda         ddphead+sd.dsckt
                sta         ddphead+ld.dsckt
                lda         ddphead+sd.ssckt
                sta         ddphead+ld.ssckt
                lda         ddphead+sd.protocol
                sta         ddphead+ld.protocol
                lda         lapdest
                sta         ddphead+ld.dnode
                lda         lapsrce
                sta         ddphead+ld.snode
                lda         #0                      ; also clear z flag to indicates ok
                sta         ddphead+ld.snet
                sta         ddphead+ld.snet+1       ; short ddp has the same net number
                beq         readddp
```

```
                fin
long            lda         #13
                sta         ddphsize
                ifne        opers-pascal
                ldy         #<rddpparm
                ldx         #>rddpparm
                else
                ldy         adrrddpparm+1
                ldx         adrrddpparm
                fin
                jsr         readheader
                do          savespace
                bne         nxtpckt
                else
                bne         reject.1                ; if error try next packet
                fin
                ldx         #1                      ; filter out wrong net
filternet       lda         ddphead+ld.dnet,x
                cmp         thisnet,x
                bne         reject
                dex
                bpl         filternet
                lda         ddphead+ld.chksum
                ora         ddphead+ld.chksum+1
                sta         chksmf                  ; if checksum is not zero, do checksum
from now on
                jsr         headersum               ; calcuate header checksum, note
headersum must return z set
readddp         lda         ddphead+ld.dsckt        ; packet for which socket
                beq         reject                  ; type 0 not allowed
                jsr         srchscktable            ; search the socket table
                bne         reject                  ; not in table, throw it away
                txa
                asl         a
                tax
                lda         listentable+1,x
                beq         reject                  ; table nil, throw it away now, but add
something later
                sta         execadr+1
                lda         listentable,x
                sta         execadr
                ifne        opers-pascal
                ldy         #<lapdata
                ldx         #>lapdata
                else
                ldy         adrlapdata+1
                ldx         adrlapdata
                fin
                jsr         doexecute
chkreject       bcc         *+5                     ; no need to discard if user took care
of it
reject          jsr         discard
reject.1        jmp         nxtpckt
;
;
;
readheader      jsr         applebus                ; read the header
                sec
                beq         discrd.9                ; ok, return with z and c set
                cmp         #2                      ; buffer overflow
                beq         discrd.9                ; ok, return with z and c set
;bcc discrd.9 ; if =1 not ok, return with z flag clear
;
```

```
carrydiscard    equ         *                       ; discard if carry set
                bcc         discrd.9                ; the rule is only if carry set
discard         equ         *                       ; discard packet
                ifne        opers-pascal
                ldy         #<rrstparm
                ldx         #>rrstparm
                else
                ldy         adrrrstparm+1
                ldx         adrrrstparm
                fin
                jsr         applebus
                lda         #1                      ; carry was clear, now clear z flag
discrd.9        rts
;
pcktmap         dfb         0
;
statparm        dfb         4,0
;
rlapparm        dfb         5
                dw          lapdata
                dw          3
;
rddpparm        dfb         5
                dw          ddphead
ddphsize        dw          5
rrstparm        dfb         2
                dw          0
                dfb         $ff
rnetparm        dfb         5
                dw          thisnet,4
                ifeq        opers-pascal
adrrlapparm     dw          rlapparm
adrstatparm     dw          statparm
adrlapdata      dw          lapdata
adrrddpparm     dw          rddpparm
adrrrstparm     dw          rrstparm
adrrnetparm     dw          rnetparm
                fin
savezp          ds          zpsize,0                ; save $40-$47 here
;
                do          savespace
movedata        equ         *
                ldx         movetable,y
                beq         discrd.9                ; 0 is end marker, done
                lda         dataarea,x              ; source
                iny
                ldx         movetable,y
                sta         dataarea,x              ; destination
                iny
                bne         movedata                ; this should be a bra
;
movetable       equ         *
move1           equ         *                       ; move short header to long header
                dfb         ddphead+sd.dsckt-dataarea,ddphead+ld.dsckt-dataarea
                dfb         ddphead+sd.ssckt-dataarea,ddphead+ld.ssckt-dataarea
                dfb         ddphead+sd.protocol-dataarea,ddphead+ld.protocol-dataarea
                dfb         lapdest-dataarea,ddphead+ld.dnode-dataarea
                dfb         lapsrce-dataarea,ddphead+ld.snode-dataarea
                dfb         const0-dataarea,ddphead+ld.snet-dataarea
                dfb         const0-dataarea,ddphead+ld.snet+1-dataarea
                dfb         0                       ; end marker
move2           equ         *                       ; make long ddp headr
                dfb         const2-dataarea,laptype-dataarea
```

```
                dfb         ddphead+sd.ssckt-dataarea,ddphead+ld.ssckt-dataarea
                dfb         ddphead+sd.dsckt-dataarea,ddphead+ld.dsckt-dataarea
                dfb         const16-dataarea,wddpsize-dataarea
                dfb         const13-dataarea,ddphead+ld.length+1-dataarea
                dfb         lapdest-dataarea,ddphead+ld.dnode-dataarea
                dfb         ournode-dataarea,ddphead+ld.snode-dataarea
                dfb         thisnet-dataarea,ddphead+ld.snet-dataarea
                dfb         thisnet+1-dataarea,ddphead+ld.snet+1-dataarea
                dfb         const0-dataarea,ddphead+ld.length-dataarea
                dfb         const0-dataarea,ddphead+ld.chksum-dataarea
                dfb         const0-dataarea,ddphead+ld.chksum+1-dataarea
;dfb a.bridge-dataarea,lapdest-dataarea
                dfb         0
move3           equ         *                       ; make short ddp header
                dfb         const1-dataarea,laptype-dataarea
                dfb         ddphead+ld.protocol-dataarea,ddphead+sd.protocol-dataarea
                dfb         const0-dataarea,ddphead+sd.length-dataarea
                dfb         const5-dataarea,ddphead+sd.length+1-dataarea
                dfb         const8-dataarea,wddpsize-dataarea
                dfb         const0-dataarea,chksmf-dataarea
                dfb         0
move4           equ         *                       ; make nbp header
                dfb         const2-dataarea,ddphead+ld.protocol-dataarea
                dfb         const2-dataarea,ddphead+sd.ssckt-dataarea
                dfb         const2-dataarea,ddphead+sd.dsckt-dataarea
                dfb         const2-dataarea,nbpbuffer+3-dataarea
                dfb         thisnet-dataarea,nbpbuffer-dataarea
                dfb         thisnet+1-dataarea,nbpbuffer+1-dataarea
                dfb         ournode-dataarea,nbpbuffer+2-dataarea
                dfb         const0-dataarea,nbpbuffer+4-dataarea
                dfb         const0-dataarea,chksmf-dataarea
                dfb         0
                fin
;
;
rtmplisten      equ         *                       ; listener for rtmp
                ldx         ddphead+ld.protocol     ; rtmp is type 1
                dex
                bne         rtmp.8                  ; not rtmp, discard
                ifne        opers-pascal
                ldy         #<rnetparm              ; read the network number
                ldx         #>rnetparm
                else
                ldy         adrrnetparm+1           ; read the network number
                ldx         adrrnetparm
                fin
                jsr         readheader
                bne         rtmp.9                  ; if error, already discarded, carry
clear is right
                lda         #361-256
                sta         bridgetime
                lda         #2
                sta         bridgetime+1
rtmp.8          sec                                 ; disregard everything else
rtmp.9          rts

; ###############################################################################################
; #   END OF FILE:  INTERRUPT
; #   LINES      :  290
; #   CHARACTERS :  13429
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.MEMMGR.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  MEMMGR
; ###############################################################################################

               page
alcbuffr       ldy          #c.bufptr+1              ;index to user specified buffer
alcbufr1       lda          (par),y                  ;this buffer must be on a page
boundary.
               tax                                   ;save in x for validation
               cmp          #$8
               bcc          obadbuf                  ;cannot be lower than video!
               cmp          #$bc                     ;nor greater than $bb00
               bcs          obadbuf                  ; since it would wipe out globals...
               sta          datptr+1
               dey
               lda          (par),y                  ;low addr should be zero!
               sta          datptr
               bne          obadbuf                  ;branch if it isn't
               inx                                   ;add 4 pages for 1k buffer
               inx
               inx
               inx
alcbufr2       dex                                   ;test for conflicts
               jsr          cmembit                  ;test for free buffer space
               and          memtabl,y                ;report memory conflict.
               bne          obadbuf                  ; if any...
               cpx          datptr+1                 ;test all four pages.
               bne          alcbufr2
               inx                                   ;add 4 pages again for allocation.
               inx
               inx
               inx
alcbufr3       dex                                   ;set proper bits to 1
               jsr          cmembit
               ora          memtabl,y                ; to mark it's allocation.
               sta          memtabl,y
               cpx          datptr+1                 ;set all four pages.
               bne          alcbufr3
               ldy          fcbptr                   ;now calculate buffer number.
               lda          fcb+fcbrefn,y
               asl          a                        ;buffer number=(entnum)*2
               sta          fcb+fcbfbuf,y            ;save it in fcb.
               tax                                   ;use entnum*2 as index to global buffer
addr tables.
               lda          datptr+1                 ;get addr already validated as good.
               sta          gl.buff-1,x              ;store hi addr (entnums start at 1, not
zero)
               clc
               rts                                   ;all done allocating buffers.
*
obadbuf        lda          #badbuf                  ;tell user buffer is in use or not
legal otherwise.
               sec                                   ;indicate error.
               rts
*
getbufadr      equ          *
               tax                                   ;index into global buffer table.
```

```
                lda     gl.buff-2,x             ;low buffer addr
                sta     bufaddrl
                lda     gl.buff-1,x             ;and high addr.
                sta     bufaddrh
                rts
*
relbuffr        jsr     getbufadr               ;preserve buf adr in 'bufaddr'
                ora     #$00                    ;returns high buffer addr in acc.
                beq     relbuf1                 ;branch if unallocated buffer space.
                lda     #0                      ;take address out of buffer list.
                sta     gl.buff-1,x             ;(x was set up by getbufadr)
                sta     gl.buff-2,x
freebuf         ldx     bufaddrh                ;get hi addr of buffer again.
                inx                             ;add 4 pages to account for 1k space
                inx
                inx
                inx
frebuf1         dex                             ;drop to next lower page.
                jsr     cmembit                 ;get bit and posn to memtabl of this
page
                eor     #$ff                    ;invert mask.
                and     memtabl,y               ;mark addr as free space now.
                sta     memtabl,y
                cpx     bufaddrh                ;all pages freed yet?
                bne     frebuf1                 ;branch if not.
relbuf1         clc                             ;indicate no error.
                rts
*
* calculate memory allocation bit position
*  entry: x=hi addr of buffer, low addr assumed zero.
*
*  exit: acc=allocation bit mask, x=unchanged, y=pointer to memtabl byte.
*
cmembit         txa                             ;move page address to acc.
                and     #7                      ;which page in any 2k set?
                tay                             ;use as index to determine
                lda     whichbit,y              ; bit position representation.
                pha                             ;save bit position mask for now.
                txa                             ;get page address again.
                lsr     a
                lsr     a                       ;now determine 2k set.
                lsr     a
                tay                             ;return it in y.
                pla                             ;restore bit mask.
                rts                             ;return bit position in a&y, pointer to
memtabl in x.
*
valdbuf         lda     usrbuf+1                ;get high addr of user's buffer
                cmp     #2                      ;must be greater than page 2
                bcc     obadbuf                 ;report bad buffer.
                skp     1
                ldx     cbytes+1
                lda     cbytes                  ;get cbytes-1 value.
                sbc     #1                      ;(carry is set)
                bcs     vldbuf0
                dex
vldbuf0         clc
                adc     usrbuf                  ;calculate end of request addr.
                txa                             ;do hi addr.
                adc     usrbuf+1                ;all we care about is final addr.
                tax                             ;must be less than $bf (globals)
                cpx     #$bf
                bcs     obadbuf
```

```
                inx                                     ;loop thru all  affected pages.
vldbuf1         dex                                     ;check next lower page.
                jsr         cmembit
                and         memtabl,y                   ;if zero then no conflict
                bne         obadbuf                     ;branch if conflict...
                cpx         usrbuf+1                    ;was that the last (lowest) page?
                bne         vldbuf1                     ;branch if not.
                clc                                     ;indicate all pages ok.
                rts                                     ;all done here
*
                page
getbuf          ldy         #c.bufadr                   ;give user address of file buffer
                lda         bufaddrl                    ; referenced by refnum.
                sta         (par),y
                iny
                lda         bufaddrh
                sta         (par),y                     ;no errors possible if this routine is
called.
                clc
                rts
*
setbuf          ldy         #c.bufadr+1
                jsr         alcbufr1                    ;allocate new buffer address over old
one
                bcs         sbuferr                     ;report any conflicts immediately
                lda         bufaddrh
                sta         usrbuf+1
                lda         bufaddrl
                sta         usrbuf
                jsr         freebuf                     ;now free address space of old buffer.
                ldy         #0
                ldx         #3
setbuf1         lda         (usrbuf),y                  ;move all four pages of the buffer to
new location.
                sta         (datptr),y
                iny
                bne         setbuf1
                inc         datptr+1
                inc         usrbuf+1
                dex
                bpl         setbuf1
                clc                                     ;indicate all is well...
sbuferr         rts                                     ;all done.
                page
************************************************
*
*  this is the routine that moves the 3 pages of dispatcher 1
*  from $d100 of the alt 4k bank to its execution address ($1000).
*  since it is in the mli and must swap the $d000-$dfff banks,
*  it must be resident at all times above $e000.
*
************************************************
                skp         2
* prodos16 patches into this area.
* if this code moves from $fcd5 then
* prodos16 must change.
                skp         2
calldisp        equ         *
                lda         ramin2
                lda         ramin2                      ;bring in the other $d000 space
                ldy         #3
svzpg           lda         0,y                         ;save zero page locations 0-3
                pha                                     ;  since we will use them as pointers
```

```
                dey
                bpl         svzpg
                lda         #<dispadr               ;destination address of user-code
                sta         3
                lda         #$d1                    ;dispatcher is stored at $d100-$d3ff
                sta         1
                lda         #0
                sta         0
                sta         2
                tay                                 ;y will be 0 now
                ldx         #3                      ;3 pages of code to move
mvpge           dey                                 ;nifty routine to move a page of code
                lda         (0),y                   ;move all 255 bytes on the page
                sta         (2),y
                tya
                bne         mvpge
                inc         1                       ;move pointers to next page
                inc         3
                dex                                 ;move all pages needed
                bne         mvpge
************************* see rev note #34 *********************
*pla ;restore the zero-page locations
****************************************************************
* ldy #$fc ;use wraparound from $ffff to $0
* restzpg pla  ; to restore z-pg locations used.
* sta $ff04,y
* iny
* bne restzpg
* fall into this code with 'x'=$00
                skp         1
@1              equ         *
                pla                                 ;recall zero page bytes
                sta         $00,x                   ;
                inx                                 ;next byte please
                cpx         #$04                    ;move 4 only
                bcc         @1                      ;loop til done
                nop                                 ;keep code same length
                skp         2
* prodos16 bra's to here with code patched
* over code above.  if this code moves
* then prodos16 must change.
                skp         2
                lda         ramin
                lda         ramin                   ;swap mli's $d000 space back in.
                lda         #>dispadr
                sta         reset                   ;set up the reset vector
                lda         #<dispadr               ;  to dispatch entry
                sta         reset+1
                eor         #$a5                    ;set up power up byte
                sta         reset+2
                jmp         dispadr
                page
* -------------------- see rev note 15 -------------------------
*  this is the (goadr) for a ram based driver located in the
*    aux language card.  it must do housekeeping and then jsr
*    to auxgo.  upon return, status must be restored before
*    leaving with an rts.
*
*  since it is in the mli and must swap the $d000-$dfff banks,
*    it must be resident at all times above $e000.
*
****************************************************************
**                                                            **
```

```
**  restriction: in order to absolutely tell a disk ][ with a      **
**   device id of 0 from a ram based driver with a device id of 0, **
**   this driver must not, repeat not, begin on a page boundary!   **
**                                                                  **
**********************************************************************
*
ramdrvr         equ         *
                php                                 ;save the processor status
                pla                                 ;get status into acc
                pha
                clv                                 ;clear the overflow flag
                and         #$4                     ;check for interrupts disabled (flag
on)
                beq         *+5                     ;branch to sei if not
                bit         ramdrtn                 ;set overflow flag on if int flag was
on
                sei                                 ;disable interrupts since we are going
to aux lc
                lda         ramin2
                lda         ramin2                  ;swap in other $d000 space
                sec                                 ;  and set carry as a process flag for
                ldx         #$5                     ;  lc.  get ready to move 5 param
bytes,
                jsr         auxgo                   ;  and then go!
*
*  ok, we're back (i hope) with error code in acc and error stat
*  in carry.
*
                sta         serr                    ;save error code in serr
                lda         ramin
                lda         ramin                   ;restore first $d000 bank
                plp                                 ;restore processor status
                clc
                lda         serr                    ;in case call is direct & needs err
code
                beq         ramdrtn
                sec                                 ;set carry on an error code not 0
ramdrtn         rts                                 ;  and return
*
                ifeq        os-prodos
*
********** see rev notes #20,#42 *****
*
*** dobus is the setup and caller for the atp driver.
*** since the driver resides in bank 2, this calling code must
*** reside at or above $e000.
*
dobus           equ         *
                sta         ramin2                  ; this should be all that is needed for
b2 r/w.
                jsr         network                 ; call the atp driver.
                sta         ramin                   ; swap back bank 1 preserving status.
                rts                                 ; errors will be returned by driver.
**************************************
                fin
                ifeq        os-ednet
*
************ see rev note #en3 *****************
*
bridge          equ         *
*
*
* on entry the stack is assumed to look like the following:
```

```
*
*  processor status
*  a register
*  x register
*  return address hi
*  return address low
*
****
*
* input : a = offset into atalk entry address table.
* output : a = error code, carry set if error.  x,y not changed.
*
call.aux.dest   equ         $200                    ; where call.aux routine will go.
*
*
                sei                                 ; no interrupts please...
                asl         a                       ; a = a*2 for displacement into table.
                pha                                 ; save for later...
*
                inc         call.level              ; now update current entry level.
                bne         skip.save               ; only save routine on first entry.
*
                ldx         #call.aux.len           ; save off pg 2 area we are using...
saveoff         lda         call.aux.dest-1,x       ;
                sta         save.pg2-1,x            ; save in local storage.
                dex                                 ;
                bne         saveoff                 ;
*
                ldx         #call.aux.len           ; now put the call.aux routine
move.bridge     lda         call.aux-1,x            ; at $200.
                sta         call.aux.dest-1,x       ;
                dex                                 ;
                bne         move.bridge             ;
*
skip.save       equ         *
*
                pla                                 ; now get x back..
                tax                                 ;
*
                lda         bridge.flag             ; save off locally used data
                pha                                 ; on the stack to maintain re-entrancy.
                lda         save.main.sp            ;
                pha                                 ;
*
                lda         atalktbl,x              ; get low byte of address.
                sta         entry.addrl             ; move it to its proper place in
                lda         atalktbl+1,x            ; the call.aux routine.
                sta         entry.addrh             ; move high byte in.
*
                tsx                                 ; now move x,a, and p values from stack
                lda         $105,x                  ; to local storage so the call.aux
                sta         savx                    ; routine can use them.
                lda         $106,x                  ; now get a...
                sta         sava                    ;
                lda         $107,x                  ; now get p...
                sta         savp                    ;
*
                jsr         call.aux.dest           ; now go do aux lc atalk routine!
*
*
                sta         x.sava                  ; save a upon return from aux.
                dec         call.level              ; now update call level back one.
                bpl         skip.restore            ; restore only on first call.
```

```
*
                ldx         #call.aux.len             ;
restore         lda         save.pg2-1,x              ; restore page 2 area used...
                sta         call.aux.dest-1,x         ;
                dex                                   ;
                bne         restore                   ; branch if not done.
*
skip.restore    pla                                   ; now restore local stuff from stack.
                sta         save.main.sp              ;
                pla                                   ;
                sta         bridge.flag               ;
                pla                                   ;
                sta         return                    ;
                pla                                   ;
                sta         return+1                  ;
                pla                                   ;
                tax                                   ;
                pla                                   ; discard original user's a.
                pla                                   ; discard original status.
                lda         return+1                  ; now prepare to return...
                pha                                   ;
                lda         return                    ;
                pha                                   ;
                lda         x.sava                    ; now recover a.
                rts                                   ; and that's all.
*
*
call.aux        equ         *                         ; this routine is relocatable and will
run at $200.
*
* input: savx,y are assumed to have an address of a parameter list.
*        sava has an atalk command number.
* output: a = error code, carry set will indicate an error.
*
                sta         $c009                     ; switch in aux lc, zp, stack.
                bvs         called.int                ; if called from int, save stack
locally.
                lda         #00                       ; indicate normal
                sta         bridge.flag               ; mode of entry
                tsx                                   ; put sp in x.
                stx         $100                      ; save main sp in $100 in alt stack.
                bvc         common.stuff              ;always taken
called.int      equ         *                         ;called from interrupt routine
                lda         #1                        ; indicate came through interrupt
                sta         bridge.flag               ; routine
                tsx                                   ; get stack pointer
                stx         save.main.sp              ; save it for later
common.stuff    equ         *                         ;rest is common to both
no.stack.setup  ldx         $101                      ; get aux sp
                txs                                   ; and set sp with it.
*
                lda         #<return.addr-1           ; push return addr on
                pha                                   ; stack so control
                lda         #>return.addr-1           ; will return after aux atalk
                pha                                   ; routine rts's.
*
*
x.entry.addrh   equ         *+1
                lda         #0                        ; ** self modified!
                pha                                   ; save aux entry address on stack.
x.entry.addrl   equ         *+1
                lda         #0                        ; ** self modified!
                pha                                   ;
```

```
*
x.savp          equ          *+1
                lda          #0                         ; ** self modified **
                pha                                     ; now restore interrupt status.
*
x.savx          equ          *+1
                ldx          #0                         ; * warning: self modified!
x.sava          equ          *+1
                lda          #0                         ; *warning: self modified!
entry.addr      equ          *+1
                rti                                     ; rti to aux while also enabling ints..
*
return.addr     equ          call.aux.dest+*-call.aux   ;
                sei                                     ; disable interrupts from here on
out...
                tsx                                     ; save aux sp
                stx          $101                       ; at $101 in aux stack.
                ldx          bridge.flag                ; check to see how we got here
                bne          from.int                   ; from an int
                ldx          $100                       ; get main sp back
                txs                                     ; into sp.
                sta          $c008                      ; switch in main lc
                rts                                     ; and return
from.int        equ          *
                ldx          save.main.sp               ; recover stack pointer
                txs                                     ; and set it up
                sta          $c008                      ; switch in main lc, zp, stack.
                rts                                     ; and return.
*
call.aux.len    equ          *-call.aux                 ; length of call.aux routine.
*
save.main.sp    equ          call.aux.dest+*-call.aux
                db           0                          ; this temp must remain in main memory!
bridge.flag     equ          call.aux.dest+*-call.aux
                db           0                          ; ditto! (used while alc in)
*
*
savx            equ          call.aux.dest+x.savx-call.aux
sava            equ          call.aux.dest+x.sava-call.aux
savp            equ          call.aux.dest+x.savp-call.aux
entry.addrh     equ          call.aux.dest+x.entry.addrh-call.aux
entry.addrl     equ          call.aux.dest+x.entry.addrl-call.aux
*
save.pg2        ds           call.aux.len,0             ; save pg 2 used here.
call.level      db           $ff                        ; used to signify what call level we're
at.
return          dw           $0000                      ;
*
***********   rpm stub   **********
*
rpm.stub        equ          *
                txa                                     ; save x on the stack.
                pha                                     ;
*
                tsx                                     ; ** must inc return address so the
                inc          $105,x                     ; rti will return correctly.
                bne          skiphi                     ;
                inc          $106,x                     ;
skiphi          equ          *
*
                clv                                     ; clear overflow flag to signify not
from int.
                lda          #2                         ; offset for rpm.
```

```
               jsr          bridge                    ; and go do it!
               jmp          exitrpm                   ; go restore lc state and return.
************************************************
               fin

; ###########################################################################################
; #   END OF FILE:  MEMMGR
; #   LINES      :  478
; #   CHARACTERS :  25276
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.NBPDRIVER.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  NBPDRIVER
; ###############################################################################################

;
nbplkup         equ          8
nbpcfrm         equ          9
;
nbppm.entity    equ          4
nbppm.timeout   equ          6
nbppm.retrycnt  equ          7
nbppm.timleft   equ          8
nbppm.tryleft   equ          9
nbppm.rsbadr    equ          10
nbppm.rsbsiz    equ          12
nbppm.maxrsp    equ          14
nbppm.rspnse    equ          15
nbppm.dnet      equ          10
nbppm.dnode     equ          12
nbppm.dsckt     equ          13
nbppm.sktret    equ          14
;
whichnbp        ldy          #0
                lda          (cmndlist),y              ; what type of command
                and          #1                        ; if nbplkup, set z flag
nbp.rts         rts
;
lookupname      equ          *                         ; nbp name lookup
                ldy          #nbppm.rspnse             ; no reply received yet
                bne          nbpsetup                  ; bra, call nbp set up and return
;
confirmname     equ          *                         ; nbp name confirm
                ldy          #nbppm.sktret             ; no socket received
; jmp nbpsetup ; call nbp set up and return
;
; common routine between lookupname and confirmname
;
tempadr         equ          cmndlist+2
;
nbpsetup        lda          #0
                sta          (cmndlist),y
                php
                sei
                lda          nbpptrsave+1              ; are we still doing another nbp
operation ?
                beq          nbpstp.1                  ; no, then we can do it
                jmp          toomanyproc
nbpstp.1        inc          outnbpid
                ldy          #nbppm.entity
nbpstp.1a       lda          (cmndlist),y
                sta          tempadr-nbppm.entity,y
                sta          entityadr-nbppm.entity,y
                iny
                cpy          #nbppm.entity+2
                bne          nbpstp.1a
                ldx          #3
                lda          #0                        ; calculate lenght of entity name
```

```
nbpstp.2        sec
                tay
                adc         (tempadr),y
                dex
                bne         nbpstp.2
                sta         entitylen
                ldy         #nbppm.retrycnt
                lda         (cmndlist),y            ; retry count
                ldy         #nbppm.tryleft
                sta         (cmndlist),y
                lda         cmndlist
                sta         nbpptrsave
                lda         cmndlist+1
                sta         nbpptrsave+1
                jsr         nbpquery                ; make a query to the network
                jmp         waitdone                ; wait until it is completed
;
; nbp timeout, maybe time to resend packet
; zp used are $40-41 plus those used by dowrtddp
;
nbptimeout      equ         *
                lda         nbpptrsave+1
                beq         nbp.rts                 ; pointer nil, just return
                sta         cmndlist+1
                lda         nbpptrsave
                sta         cmndlist
                ldy         #nbppm.timleft
                do          savespace
                jsr         decindy                 ; decrement time left
                bne         nbp.rts                 ; still some left
                ldy         #nbppm.tryleft          ; else decrement retry if not zero
                lda         (cmndlist),y
                beq         nbptim.6                ; no more retry left
                jsr         decindy
                else
                lda         (cmndlist),y            ; decrement time left
                sec
                sbc         #1
                sta         (cmndlist),y
                bne         nbp.rts                 ; still some left
                ldy         #nbppm.tryleft
                lda         (cmndlist),y
                bne         *+5
                jmp         nbptim.6
                sec
                sbc         #1
                sta         (cmndlist),y
                fin
nbpquery        ldy         #nbppm.timeout
                lda         (cmndlist),y
                ldy         #nbppm.timleft
                sta         (cmndlist),y
                lda         #$21                    ; assume it is lkup
                sta         outnbp
                jsr         whichnbp
                bne         nbptim.3                ; confirm
                sta         ddphead+ld.dnet         ; it is just local, zero dest net
                sta         ddphead+ld.dnet+1
                ldx         a.bridge                ; do we have a bridge ?
                bne         nbptim.2a               ; yes
                dex                                 ; else broadcast
                bne         nbptim.2b               ; bra
nbptim.2a       lda         #$11                    ; go to bridge, command is brrq
```

```
                sta         outnbp
nbptim.2b       stx         lapdest                 ; destination is bridge or broadcast
                bne         nbptim.4                ; either x not 0 or acc=$11, therefore
bra
nbptim.3        ldy         #nbppm.dnet
nbptim.3a       lda         (cmndlist),y
                sta         ddphead+ld.dnet-nbppm.dnet,y ; copy dest net
                iny
                cpy         #nbppm.dnet+2
                bne         nbptim.3a
                lda         (cmndlist),y
                sta         lapdest                 ; destination node
                do          savespace
nbptim.4        ldy         #move4-movetable
                jsr         movedata
                else
nbptim.4        lda         #2
                sta         ddphead+ld.protocol     ; nbp is type 2
                sta         ddphead+sd.ssckt        ; from socket 2
                sta         ddphead+sd.dsckt        ; to socket 2
                sta         nbpbuffer+3             ; entity socket ; may not be needed
                lda         thisnet
                sta         nbpbuffer
                lda         thisnet+1
                sta         nbpbuffer+1
                lda         ournode
                sta         nbpbuffer+2
                lda         #0
                sta         nbpbuffer+4
                sta         chksmf                  ; do not do checksum
                fin
                ifne        opers-pascal
                ldx         #>nbpwds
                ldy         #<nbpwds
                else
                ldx         adrnbpwds
                ldy         adrnbpwds+1
                fin
                jsr         dowrtddp                ; do write ddp
                beq         nbprts                  ; ok, done
                tax                                 ; save error
                ldy         #nbppm.tryleft
                lda         (cmndlist),y            ; any more tryies
                bne         nbprts                  ; yes, then not a error
                txa                                 ; get back error
                bne         nbptim.8                ; bra
nbptim.6        jsr         whichnbp
                beq         nbptim.8                ; retry expires, terminate it with
error 0
nbptim.7        lda         #notconfirmed           ; if to confirm, then it was not
confirmed
nbptim.8        jmp         nbperror
;
; socket listener for nbp
;
tblptr          equ         cmndlist+2
nbpbflimit      equ         cmndlist+4
rdmvvar         equ         cmndlist+6
nbprcvct        equ         cmndlist+7
                ifeq        opers-prodos
tuplect         equ         $fa                     ; in prodos interrupt, this is saved so
we can use it
                fin
```

```
;
nbplisten       lda         #2                      ; nbp has protocol 2
                cmp         ddphead+ld.protocol
                bne         nbpls.01                ; if not, discard it
                jsr         readnbpfx               ; since acc = 2, read 2 byte for nbp
                bne         nbpls.01                ; not available, do not proceed
                lda         nbpbuffer
                tax
                and         #$0f
                beq         nbpls.01                ; no tuple, discard it
                sta         tuplect
                txa
                and         #$f0
                cmp         #$30                    ; lkup-reply ?
                beq         nbpls.00                ; yes, got a reply
                cmp         #$20                    ; lkup ?
                bne         nbpls.01                ; no, discard it
                lda         nbphook+1               ; do we have a lookup hook?
                beq         nbpls.01                ; no, discard it
                lda         nbpbuffer+1
                ldx         tuplect
nbphook         equ         *+1
                jmp         $0000                   ; nbp hook
;
nbpls.00        lda         nbpptrsave+1            ; are we processing any nbp
                beq         nbpls.01                ; no, ignore it
                sta         cmndlist+1
                lda         outnbpid
                cmp         nbpbuffer+1             ; is this the one we asked
                beq         nbpls.02                ; yes, process it
nbpls.01        sec                                 ; discard and then return
nbprts          rts
nbpls.02        lda         nbpptrsave
                sta         cmndlist
nbpls.03        dec         tuplect
                bmi         nbpls.01                ; all taken care of
                lda         #5
                jsr         readnbpfx               ; read a entity addr
                bne         nbpls.01                ; not available, do not proceed
                jsr         whichnbp
                beq         nbpls.04
                lda         nbpbuffer+3             ; get actual socket
                ldy         #nbppm.sktret
                sta         (cmndlist),y
                dey
                cmp         (cmndlist),y            ; did it agree with what we ask for?
                do          savespace
                beq         nbpdone                 ; confirmed
                else
                bne         *+5                     ; different socket
                jmp         nbpdone                 ; confirmed
                fin
                lda         #diffsocket             ; confirmed with wrong socket
                bne         nbperror                ; bra
nbpls.04        clc
                ldy         #nbppm.rsbadr
                lda         (cmndlist),y            ; get response buffer and its boundary
                sta         tblptr
                ldy         #nbppm.rsbsiz
                adc         (cmndlist),y
                sta         nbpbflimit
                dey
                lda         (cmndlist),y
```

```
                sta         tblptr+1
                ldy         #nbppm.rsbsiz+1
                adc         (cmndlist),y
                sta         nbpbflimit+1
                ldy         #nbppm.rspnse           ; get response received, this is
current table size
                lda         (cmndlist),y
                tax
                inx
                stx         nbprcvct
nbpls.05        dec         nbprcvct
                beq         nbpls.10                ; not in table
                ldy         #4                      ; check if item is already in table
nbpls.06        lda         (tblptr),y
                cmp         nbpbuffer,y
                bne         nbpls.07                ; not the same entry
                dey
                bpl         nbpls.06
                jsr         skipfield               ; duplicate, ignore it
                bne         nbprts                  ; ignore bad packet
                jsr         skipfield
                bne         nbprts                  ; ignore bad packet
                jsr         skipfield
                bne         nbprts                  ; ignore bad packet
nbpls.06c       jmp         nbpls.03
nbpls.07        ldy         #5                      ; points to entity name
                ldx         #3                      ; repeat for all three fields
nbpls.08        sec                                 ; add 1 extra
                tya
                adc         (tblptr),y              ; add length count
                tay
                dex
                bne         nbpls.08
                jsr         inctblptr               ; skip over entry in table
                beq         nbpls.05                ; since inctblptr set z flag, bra ->
check more from table
; we come to this point when nbprcvct is 0, indicate search is over
; we shall use the same location to indicate that whether buffer is full or not
; so we don't need to clear the flag and save 1 instruction
nbpls.10        lda         #5
                sta         rdmvvar
                jsr         movenbp                 ; move entity addr
                jsr         rdmvfield
                bcc         rdmv.rts                ; bad packet ignored
                jsr         rdmvfield
                bcc         rdmv.rts                ; bad packet ignored
                jsr         rdmvfield
                bcc         rdmv.rts                ; bad packet ignored
                lda         #bufferfull             ; assume buffer full
                ldy         nbprcvct                ; was buffer full ?
                bne         nbperror                ; then do not increase count
                ldy         #nbppm.rspnse
                do          savespace
                jsr         incindy
                else
                clc
                lda         (cmndlist),y            ; increase item count
                adc         #1
                sta         (cmndlist),y
                fin
                dey                                 ; y = 10
                cmp         (cmndlist),y            ; compare number of response
                bcc         nbpls.06c               ; still ok
```

```
nbpdone         lda          #0
nbperror        ldy          #0
                sty          nbpptrsave+1              ; no more nbp operation
                jmp          retasynsec                ; set asyn result, do io completion,
then sec carry
;
rdmvfield       equ          *                         ; read from field and move into table
                lda          #1
                jsr          readmove
                bne          rdmv.rts                  ; operation failed
                lda          nbpbuffer                 ; this is number of bytes
readmove        sta          rdmvvar
                jsr          readnbp
                bne          rdmv.rts
;jsr movenbp ; if read ok then move it
;rts
;
movenbp         lda          nbprcvct                  ; is table full already
                bne          rdmvsec                   ; yes, do not move
                sec                                    ; check if table can accept rdmvvar
more bytes
                lda          nbpbflimit
                sbc          tblptr
                cmp          rdmvvar
                bcs          rdmv.1
                lda          nbpbflimit+1
                sbc          tblptr+1
                bcs          rdmv.1                    ; enough room
                sta          nbprcvct                  ; since acc is non-zero, nbprcvct now
indicate error
                rts                                    ; returns with carry clear
rdmv.1          ldy          rdmvvar                   ; enough room, then move it
                dey
rdmv.2          lda          nbpbuffer,y               ; and adjust table pointer
                sta          (tblptr),y
                dey
                bpl          rdmv.2
                lda          rdmvvar
inctblptr       clc
                adc          tblptr
                sta          tblptr
                bcc          rdmv.7
                inc          tblptr+1
rdmv.7          lda          #0                        ; set z flag
rdmvsec         sec
rdmv.rts        rts
;
skipfield       lda          #1
                jsr          readnbpfx                 ; read field byte
                bne          rdmv.rts
                lda          nbpbuffer
; jsr readnbp ; read the field
; rts
; fall through to readnbp
;
readnbp         cmp          #33
                bcs          rdnp.9                    ; never read more than size of buffer
readnbpfx       sta          rnbpcnt                   ; number of bytes to read
                ifne         opers-pascal
                ldy          #<rnbpparm
                ldx          #>rnbpparm
                else
                ldy          adrrnbpparm+1
```

```
                ldx         adrrnbpparm
                fin
                jmp         readheader              ; read the nbp header
rdnp.9          jmp         discard
;
                ifeq        opers-pascal
adrrnbpparm     dw          rnbpparm
                fin
                ifne        opers-prodos
tuplect         dfb         0
                fin
;
                ifeq        opers-pascal
adrnbpwds       dw          nbpwds
                fin
;
rnbpparm        dfb         5
                dw          nbpbuffer
rnbpcnt         dw          0
;
nbpwds          ds          4,0                     ; 4 bytes to be filled in
                dw          7,outnbp
entitylen       dw          0
entityadr       dw          0
                dw          $ffff

; ###########################################################################################
; #   END OF FILE:  NBPDRIVER
; #   LINES      :  383
; #   CHARACTERS :  17583
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.NEWFNDVOL.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  NEWFNDVOL
; ###############################################################################################

*
preproot        jsr         findvol                 ;search vcb's and devices for specified
volume.
                bcs         novolume                ;branch if not found.
                lda         #0                      ;zero out directory temps.
                ldy         #$42                    ;
clrdsp          sta         own.blok,y              ;and owner info.
                dey
                bpl         clrdsp
                lda         devnum                  ;set up device number for this
directory.
                sta         d.dev
                jsr         movhed0                 ;set up other header info from
directory in gbuf.
                ldy         #1                      ;and clean up misc.
                ldx         vcbptr
                inx
rootmisc        lda         vcb+vcbtblk,x           ;misc info includes
                sta         h.tblk,y                ;total # of blocks,
                lda         vcb+vcbdmap,x           ; the address of the first bitmap
                sta         h.bmap,y
                lda         bloknml,y               ;and directory's disk address,
                sta         d.head,y
                lda         h.fcnt,y                ;and lastly, setting up a counter for
the number of
                sta         entcntl,y               ; files in this directory.
                dex                                 ;move low order bytes too.
                dey
                bpl         rootmisc
*
nxtpname        jsr         nxtpnam1                ;get new pnptr in y and next namlen in
acc.
                sty         pnptr                   ;save new pathname pointer.
                rts                                 ;(status reg according to accumulator)
*
nxtpnam1        ldy         pnptr                   ;bump pathname pointer to next name
                lda         pathbuf,y               ; in the path...
                sec
                adc         pnptr                   ;if this addition results in zero,
                tay                                 ; then prefixed directory has been
moved to
                bne         nxpnmret                ; another device. branch if not.
                lda         devnum                  ;revise devnum for prefixed directory
                sta         p.dev
nxpnmret        lda         pathbuf,y               ;test for end of name.
                clc                                 ;indicate no errors.
novolume        rts
*
*
findvol         lda         #0
                ldy         pfixptr                 ;use prefix volume name to look up vcb.
                bit         prfxflg                 ;is this a prefixed path?
                bpl         fndvol2                 ;branch in it is.
```

```
*
fndvol1         tay                             ;set ptr to volume name.
fndvol2         sty         vnptr               ;save pointer.
                sta         devnum              ;zero out device number until vcb
located.
fndvol3         pha                             ;acc now used as vcb lookup index
                tax                             ;move pointer to x for index.
                lda         vcb,x               ;get vcb volume name length
                bne         matchvol            ;branch if claimed vcb to be tested.
nxtvcb          ldy         vnptr               ;restore ptr to reqested volume name.
                pla                             ;now adjust vcb index to next vcb
entry.
                clc
                adc         #$20
                bcc         fndvol3             ;branch if more vcb's to check.
                bcs         lookvol             ;otherwise go look for unlogged
volumes.
*
matchvol        sta         namcnt              ;save length of vol name to be compared
mtchvol1        cmp         pathbuf,y           ;is it the same as requested vol name?
                bne         nxtvcb              ;branch if not.
                inx
                iny                             ;bump to next character
                lda         vcb,x
                dec         namcnt              ;was that the last character?
                bpl         mtchvol1            ;branch if not.
*
                pla                             ;restore pointer to vcb that matches.
                tax
                sta         vcbptr              ;save it for future reference.
                lda         vcb+vcbdev,x        ;get its device number
                sta         devnum              ;save it.
                lda         #$00                ; assume prefix is not used and
                sta         bloknmh             ; that root directory is to be used.
                lda         #$02
                sta         bloknml
                lda         vnptr               ;= 0 if no prefix.
pfxdir          tay                             ;if prefix, then find ptr to prefixed
dir name.
                sta         pnptr               ;save path ptr.
                beq         fndvol4             ;branch if no prefix.
                sec                             ;bump to next dir in prefix path
                adc         pathbuf,y
                bcc         pfxdir              ;branch if there is another dir in
prefix.
                lda         p.blok              ;volume verification will occur at
                sta         bloknml             ; sub directory level.
                lda         p.blok+1
                sta         bloknmh
*
******verify volume name******
*
fndvol4         jsr         rdgbuf              ;read in directory (or prefix
directory)
                bcs         wrgvol              ;if error then look on other devices
                jsr         cmppnam             ;compare directory name with pathname
                bcc         wrg7                ;if they match, don't look elswhere.
*
wrgvol          ldx         vcbptr              ;find out if current (matched) vcb is
active.
                lda         vcb+vcbstat,x       ; i.e. does it have open files?
                bmi         lvolerr             ;report not found if active.
*
```

```
lookvol         lda         vnptr                   ;make path pointer same as volume ptr.
                sta         pnptr
                jsr         mvdevnums               ;copy all device numbers to be
examined.
                lda         devnum                  ;log current device first, before
searching others
                bne         wrg3
wrg1            ldx         devcnt                  ;scan look list for devices we need
wrg2            lda         loklst,x                ; to search for the requested volume.
                bne         wrg4                    ;branch if we've a device to look at.
                dex
                bpl         wrg2                    ;look at next guy.
lvolerr         lda         #vnferr                 ;report that no mounted volume
                sec                                 ; matches the requested.
wrg7            rts
*
wrg3            ldx         devcnt                  ;now remove the device from the list
wrg4            cmp         loklst,x                ; of prospective devices (so we don't
look twice)
                beq         wrg5                    ;branch if match.
                dex                                 ;look until found.
                bpl         wrg4                    ;branch always taken! (usually!) * * *
* -------------------- see rev note 16 -----------------------------
                bmi         lvolerr                 ;never unless device was manually
removed from devlst (/ram)
* ---------------------------------------------------------------
*
wrg5            sta         devnum                  ;preserve dedice we're about to
investigate.
                lda         #0
                sta         loklst,x                ;mark this one as tested.
                jsr         fnddvcb                 ;find vcb that claims this device, if
any.
                bcs         fnvolerr                ;branch if vcb full.
                ldx         vcbptr                  ;did 'fnddvcb' find it or did it return
free vcb?
                lda         vcb,x
                beq         wrg6                    ;branch if free vcb.
                lda         vcb+vcbstat,x           ;is this volume active?
                bmi         wrg1                    ;if so, no need to re-log.
wrg6            lda         #2                      ;go read root directory into gbuf.
                ldx         #0
                jsr         rdblk
                bcs         wrg1                    ;ignore if unable to read.
                jsr         logvcb                  ;go log in this volume's proper name.
                bcs         wrg1                    ;look at next if non xdos disk was
mounted.
                jsr         cmppnam                 ;is this the volume we're looking for?
                bcs         wrg1                    ;branch if not.
fnvolerr        rts                                 ;return to caller.
*
mvdevnums       ldx         devcnt                  ;copy all device numbers to be
examined.
mvdvnum1        lda         devlst,x
                and         #$f0                    ;strip device type info.
                sta         loklst,x                ;copy them to a temporary workspace.
                dex
                bpl         mvdvnum1
                ldx         devcnt
                rts
*
fnddvcb         lda         #0                      ;look for vcb with this device number.
                ldy         #$ff
```

```
fddvcb          tax                             ;new index to next vcb.
                lda         vcb+vcbdev,x        ;check all devnums.
                cmp         devnum              ;is this the vcb were looking for?
                bne         fnxtvcb             ;branch if not.
                stx         vcbptr
                clc                             ;indicate found.
                rts
fnxtvcb         lda         vcb,x               ;is this a free vcb?
                bne         fnxvcb1             ;branch if not.
                iny
                stx         vcbptr
fnxvcb1         txa                             ;now...
                clc                             ;bump index to next vcb.
                adc         #$20
                bne         fddvcb
                tya                             ;were any free vcb's available?
                bmi         vfulerr             ;branch if not.
                clc                             ;indicate no errors
vfulerr         lda         #vcbfull
                rts
*
cmppnam         ldx         #0                  ;index to directory name.
                ldy         pnptr               ;index to pathname
                lda         gbuf+4              ;get directory name length (and type)
                cmp         #$e0                ;also make sure it's a directory
                bcc         vnocmp              ;branch if not a directory.
                and         #$f                 ;isolate name length
                sta         namcnt              ;save as counter.
                bne         cmpvnm1             ;branch if valid length
vnocmp          sec                             ;indicate not what were looking for.
                rts
cmpvnm0         lda         gbuf+4,x            ;get next char.
cmpvnm1         cmp         pathbuf,y
                bne         vnocmp              ;branch if not the same.
                inx                             ;chek nxt char.
                iny
                dec         namcnt
                bpl         cmpvnm0             ;branch if more to compare.
                clc                             ;otherwise we got a match!!!
                rts
*
logvcb          ldx         vcbptr              ;is this a previously logged in volume
                lda         vcb,x               ;(acc=0?)
                beq         logvcb1             ;no, go ahead and prepare vcb.
                jsr         cmpvcb              ;does vcb match volume read?
                bcc         vcblogd             ;yes, don't disturb it.
*
logvcb1         lda         #0                  ;zero out vcb entry
                ldy         #vcbsize-1
zervcb          sta         vcb,x
                inx
                dey
                bpl         zervcb
                jsr         tstsos              ;make sure it's an xdos diskette.
                bcs         vcblogd             ;if not, return carry set.
                jsr         tstdupvol           ;find out if a duplicate with open
files already exists
                bcs         notlog0
                lda         gbuf+4              ;move volume name to vcb
                and         #$f                 ;strip root marker
                tay
                pha
                ora         vcbptr
```

```
                tax
movolnm         lda         gbuf+4,y
                sta         vcb,x
                dex
                dey
                bne         movolnm
                pla                                 ;get length again
                sta         vcb,x                   ;save that too.
                lda         devnum
                sta         vcb+vcbdev,x            ;save device number also.
                lda         gbuf+vtblk+4            ;and totol number of blocks on this
unit,
                sta         vcb+vcbtblk,x
                lda         gbuf+vtblk+5
                sta         vcb+vcbtblk+1,x
                lda         bloknml                 ;and address of root directory
                sta         vcb+vcbroot,x
                lda         bloknmh
                sta         vcb+vcbroot+1,x
                lda         gbuf+vbmap+4            ;and lastly, the address
                sta         vcb+vcbdmap,x           ; of the first bitmap
                lda         gbuf+vbmap+5
                sta         vcb+vcbdmap+1,x
notlog0         clc                                 ;indicate that it was logged if
posible.
vcblogd         rts
*
cmpvcb          lda         gbuf+4                  ;compare volume name in vcb
                and         #$f                     ; with name in directory
                cmp         vcb,x                   ;are they same length
* -------------------- see rev note 23 --------------------
                stx         xvcbptr
* ---------------------------------------------------------
                bne         notsam1
                tay
                ora         xvcbptr
                tax
vcbcmp1         lda         gbuf+4,y
                cmp         vcb,x
notsam1         sec                                 ;anticipate different names
                bne         notsame
                dex
                dey
                bne         vcbcmp1
                clc                                 ;indicate match.
notsame         equ         *
* -------------------- see rev note 23 --------------------
                ldx         xvcbptr                 ;get back offset to start of vcb
* ---------------------------------------------------------
                rts
*
tstdupvol       lda         #0                      ;look for other logged in volumes with
same name.
tsdupv1         tax
                jsr         cmpvcb
                bcs         tsdupv2                 ;branch if no match.
                lda         vcb+vcbstat,x           ;test for any open files
                bmi         founddup                ;tell the sucker he can't look at this
volume!
                lda         #0                      ;take duplicate off line if no open
files.
                sta         vcb,x
                sta         vcb+vcbdev,x
```

```
                beq         nodupvol                ;return that all is ok to log in new.
*
tsdupv2         txa                                 ;index to next vcb
                clc
                and         #$e0                    ;strip odd stuff.
                adc         #vcbsize                ;bump to next entry.
                bcc         tsdupv1                 ;branch if more to look at.
*
nodupvol        clc
                rts
*
founddup        sta         duplflag                ;a duplicate has been detected.
                stx         vcbentry                ;save pointer to conflicting vcb.
                sec                                 ;indicate error
                rts
                page
*
tsfrblk         ldx         vcbptr                  ;find out if enough free blocks
                lda         vcb+vcbtfre+1,x         ;available to accomdate the request.
                ora         vcb+vcbtfre,x           ;but first find out if we got a proper
count for this volume.
                bne         cmpfreb                 ;branch if count is non-zero
tkfrecnt        jsr         cntbms                  ;get # of bitmaps.
                sta         bmcnt                   ;save it.
                lda         #0                      ;start count at zero.
                sta         scrtch
                sta         scrtch+1
                lda         #$ff                    ;mark 'first free' temp as unknown
                sta         nofree
                jsr         upbmap                  ;(nothing happens if it don't hafta.)
                bcs         tfberr                  ;branch if we got trouble,
                ldx         vcbptr                  ;get address of first bit map.
                lda         vcb+vcbdmap,x
                sta         bloknml
                lda         vcb+vcbdmap+1,x
                sta         bloknmh
bmaprd          jsr         rdgbuf                  ;use g(eneral)buff(er) for temporary
                bcs         tfberr                  ; space to count free blocks (bits)
                jsr         count                   ;go count em
                dec         bmcnt                   ;was that the last bit map?
                bmi         chgvcb                  ;if so, go change fcb to avoid doing
this again!
                inc         bloknml                 ;note: the organization of the bit maps
                bne         bmaprd                  ; are contiguous for sos version 0
                inc         bloknmh                 ; if some other organization is
implemented, this code
                jmp         bmaprd                  ; must be changed!
                page
*
chgvcb          ldx         vcbptr                  ;mark which block had first free space
                lda         nofree
                bmi         dskfull                 ;branch if no free space was found.
                sta         vcb+vcbcmap,x           ;update the free count.
                lda         scrtch+1                ;get high count byte
                sta         vcb+vcbtfre+1,x         ;update volume control block.
                lda         scrtch
                sta         vcb+vcbtfre,x           ;and low byte too...
cmpfreb         lda         vcb+vcbtfre,x           ;compare total available
                sec
                sbc         reql                    ; free blocks on this volume.
                lda         vcb+vcbtfre+1,x
                sbc         reqh
                bcc         dskfull
```

```
                clc
                rts
dskfull         lda         #ovrerr
                sec
tfberr          rts
                page
*
count           ldy         #0                      ;begin at the beginning.
frcont          lda         gbuf,y                  ;get bit pattern
                beq         frcnt1                  ;don't bother counting nothin'
                jsr         cntfree
frcnt1          lda         gbuf+$100,y             ;do both pages with same loop
                beq         frcnt2
                jsr         cntfree
frcnt2          iny
                bne         frcont                  ;loop till all 512 bytes counted
                bit         nofree                  ;has first block with free space been
found yet?
                bpl         frcnt3                  ;branch if it has.
                lda         scrtch                  ;test to see if any blocks were counted
                ora         scrtch+1
                beq         frcnt3                  ;branch if none counted.
                jsr         cntbms                  ;get total # of maps.
                sec                                 ;subtract countdown from total bit maps
                sbc         bmcnt
                sta         nofree
frcnt3          rts
*
cntfree         asl         a                       ;count the number of bits in this byte.
                bcc         cfree1
                inc         scrtch
                bne         cfree1
                inc         scrtch+1
cfree1          ora         #0
                bne         cntfree                 ;loop until all bits counted.
                rts
*
cntbms          ldx         vcbptr
                ldy         vcb+vcbtblk+1,x         ;return the # of bit maps
                lda         vcb+vcbtblk,x           ; posible with the total count
                bne         cntbms1                 ; found in the vcb...
                dey                                 ; adjust for bitmap block boundary
cntbms1         tya
                lsr         a                       ;divide by 16. the result is the number
                lsr         a                       ; of bit maps.
                lsr         a
                lsr         a
                rts
*

; ###############################################################################################
; #   END OF FILE:  NEWFNDVOL
; #   LINES      :  385
; #   CHARACTERS :  22320
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.OPENDRIVER.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  OPENDRIVER
; ###############################################################################################

machid          equ         $bf98                   ; for //c initialization
acia            equ         $c09a
aciaintr        equ         $5f9
;
closedriver     dey                                 ; y now = 0
                sty         openflag                ; no longer open
                sty         initparm+1              ; then sign off from the card
callinit        equ         *
                ifne        opers-pascal
                ldx         #>initparm
                ldy         #<initparm              ; cannot be 0
                else
                ldx         adrinitparm
                ldy         adrinitparm+1
                fin
                bne         applebus                ; this is bra, call card & return
;
;
;
slotadr         equ         cmndlist+2              ; zp usage for opendriver
slotcount       equ         cmndlist+4
opendriver      equ         *
                ldx         openflag
                bmi         openerr1
                stx         slotadr                 ; openflag is zero, therefore x is 0
                inx                                 ; x = 1, i.e. assume slot is given
                and         #7                      ; slot number is in acc
                bne         opndrv.1                ; we have a slot, search there
                lda         #7                      ; start with slot 7
                tax                                 ; and search 7 slots
opndrv.1        equ         *                       ; find a slot with a appletalk card
                ora         #$c0                    ; generate slot number
                sta         slotadr+1
                stx         slotcount               ; number of slots to search
opndrv.1a       ldx         #7                      ; look at 4 id bytes
opndrv.1b       ldy         cmptable,x
opndrv.1c       lda         (slotadr),y
                dex
                cmp         cmptable,x
                beq         opndrv.1d
                dec         slotadr+1               ; wrong id byte, not appletalk card
                dec         slotcount
                bne         opndrv.1a
                lda         #nointcard              ; card not found
                bne         openerr1+2              ; error, bra
openerr1        lda         #alreadyopen
                sta         status
                rts
opndrv.1d       dex
                bpl         opndrv.1b               ; check more bytes
                lda         slotadr+1               ; right id byte, this is slot number
opndrv.2        sta         cardslot
                ldy         #$ff
```

```
                lda         (slotadr),y             ; get version number of card
                sta         cardversion
                ldy         #3
                lda         (cmndlist),y            ; get node number
                sta         initparm+1
                jsr         callinit
                bne         opendone                ; return if error
                lda         #0
                sta         prtclhandler+1
                sta         usertimer+1
                sta         nbphook+1
                sta         a.bridge
;initialize socket table, atprqtable
                ldx         #listentable-ddptable-2
                sta         ddptable+1,x
                dex
                bne         *-4
                do          twoc
                ifeq        opers-prodos
                lda         machid
                and         #$c8
                cmp         #$88
                bne         nottwoc
                lda         acia
                ora         #$0f
                sta         acia
                lda         #$c0
                sta         aciaintr
nottwoc         equ         *
                fin
                fin
                do          twoc
                ifeq        opers-kernel
                lda         $fbb3
                cmp         #$06
                bne         nottwoc
                lda         $fbc0
                cmp         #$40
                bcs         nottwoc
                lda         acia
                ora         #$0f
                sta         acia
                lda         #$c0
                sta         aciaintr
nottwoc         equ         *
                fin
                fin
                ifne        opers-pascal
                ldx         #>startparm
                ldy         #<startparm
                else
                ldx         adrstartparm
                ldy         adrstartparm+1
                fin
                jsr         applebus                ; start timer and bus interrupt
                lda         initparm+1              ; get node number
                sta         ournode                 ; also save for our own use
                ldy         #1                      ; copy the data to be returned to
caller
opendrv.4       lda         cardslot-1,y
                sta         (cmndlist),y
                iny
                ifeq        opers-prodos
```

```
                cpy         #7                      ; prodos copy up to mlibypass
                else
                cpy         #5                      ; others copy up to interrupt handler
                fin
                bne         opendrv.4
                lda         #$80
                sta         openflag
opendone        rts
;
cmptable        dfb         $9b,$0c,$01,$0b,$18,$07,$38,$05
;
                ifeq        opers-pascal
adrstartparm    dw          startparm
adrinitparm     dw          initparm
                fin
;
;
;
hookadr         equ         cmndlist+2
;
sethook         equ         *                       ; set various hook into driver
                php
                sei                                 ; we must disable interrupt first
                asl         a                       ; offset into table, high bit in carry
                tax
                bcs         sethk.5                 ; getinfo
                lda         hooktable,x
                sta         hookadr
                lda         hooktable+1,x
                sta         hookadr+1
                cpx         #hooksize               ; is it in range
                bcs         sethk.9                 ; no
sethk.3         iny                                 ; now y is 2
                lda         (cmndlist),y
                sta         (hookadr),y
                cpy         #3
                bne         sethk.3
                plp
                rts
sethk.5         cpx         #4                      ; is it in range
                bcs         sethk.9
sethk.7         iny                                 ; y is now 2
                lda         version,x
                sta         (cmndlist),y
                inx
                cpy         #3
                bne         sethk.7
                plp
                rts
sethk.9         lda         #illegalvalue
                bne         opnskt.8                ; return error and pop stack
;
hooktable       equ         *                       ; all address must be two less bacause
y is 2
                dw          prtclhandler-2
                dw          usertimer-2
                dw          nbphook-2
hooksize        equ         *-hooktable
;
;
;
openddpsckt     equ         *                       ; open a ddp socket
                pha
```

```
                ldy         #3
                lda         (cmndlist),y            ; get listener address
                tax
                iny
                lda         (cmndlist),y
                tay
                pla
getnscktno      jsr         opensckt                ; open the socket
                ldy         #2
                sta         (cmndlist),y            ; return socket id
                rts
;
; internal subroutine to open a socket
;
opensckt        equ         *
                php
                sei
                stx         sktlst.lo
                sty         sktlst.hi
                tay
                bmi         opnskt.6                ; illegal socket number if > $7f
                beq         opnskt.1                ; dynamic, no search
                jsr         srchscktable
                beq         opnskt.7                ; duplicate
                lda         #0
opnskt.1        jsr         srchscktable            ; find open entry
                bne         opnskt.7                ; table full
                tya                                 ; get back socket
                bne         opnskt.2                ; well known, put it in
                txa                                 ; dynamic, generate one
                adc         #$7f                    ; remember that carry was set
opnskt.2        sta         ddptable,x
                tay
                txa
                asl         a
                tax                                 ; x = x * 2
sktlst.lo       equ         *+1
                lda         #0
                sta         listentable,x
sktlst.hi       equ         *+1
                lda         #0
                sta         listentable+1,x
                plp
                tya                                 ; get back socket #
                rts
opnskt.6        lda         #badsocket
                bne         opnskt.8
opnskt.7        lda         #socketopen
opnskt.8        sta         status
                plp
                lda         #0                      ; if error, return socket number 0
                rts
;
chkvalidsckt    equ         *                       ; check for valid socket #, if not exit
with error
                cmp         #3                      ; do not use rtmp and nbp
                bcc         chksckt.8
chkopensckt     jsr         srchscktable            ; look for it in socket table
                beq         clseddp.9               ;  found
chksckt.8       lda         #scktnotopen
                sta         status
                rts
;
```

```
clsatp.8        plp
                rts
                fin
;
;
atprqptr        equ         cmndlist+2
;
;
closeddpsckt    equ         *
                jsr         chkvalidsckt
                bne         clseddp.9
                lda         #0
                sta         ddptable,x
                txa
                asl         a
                tax
                lda         #0                      ; zero out parameter links header.
                sta         atprptable+1,x
                do          safeclose
                lda         atprptable+1,x
                beq         clseddp.9
                sta         atprqptr+1
                lda         atprptable,x
                sta         atprqptr
                lda         #0
                sta         atprptable+1,x
clsatp.1        ldy         #1
                lda         #reqabort               ; misc error
                sta         (atprqptr),y
                jsr         nextinqueue
                bne         clsatp.1
                else
                lda         #0
                sta         atprptable+1,x
                fin
clseddp.9       rts
;
srchscktable    equ         *
                ldx         #ddptblsiz-1
srcsckt1        cmp         ddptable,x
                beq         srcsckt9                ; found, return with carry set
                dex
                bpl         srcsckt1                ; ddptblsiz must be less than 128
srcsckt9        rts
;
;
startparm       dfb         4
                dfb         $80                     ; enable both timer and packet
interrupt
;
initparm        dfb         1
                dfb         0
;
ddptblsiz       equ         8
ddptable        dfb         1,2
                ds          ddptblsiz-2,0           ; rest of the table
processtable    equ         *
nbpptrsave      ds          2,0
rspcbsize       equ         4
rspcbbegin      equ         *-processtable
rspcbtable      ds          rspcbsize*2,0
rspcbend        equ         *-processtable
atprqtblsiz     equ         4
```

```
atprqbegin      equ          *-processtable
atprqtable      ds           atprqtblsiz*2,0
atprqend        equ          *-processtable
atprptable      equ          *-4
                ds           ddptblsiz*2-4,0
listentable     dw           rtmplisten,nbplisten      ; built-in listener table
                ds           ddptblsiz*2-4,0           ; rest of listener table

; ###############################################################################################
; #   END OF FILE:  OPENDRIVER
; #   LINES      :  308
; #   CHARACTERS :  12455
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.PLDR.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  PLDR
; ###############################################################################################

*
                skp          1
rtbl            equ          $10
src             equ          $12
dst             equ          $14
cnt             equ          $16
cde             equ          $18
ecde            equ          $1a
machid          equ          $bf98
                org          $2000
                incl         /mli.2/proldr
                incl         /mli.2/devsrch
                incl         /mli.2/reloc
                ds           $27ff-*,0
doramdsk        nop
zzzend          equ          *

; ###############################################################################################
; #   END OF FILE:  PLDR
; #   LINES      :  16
; #   CHARACTERS :  504
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.POSN.OPEN.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  POSN.OPEN
; ###############################################################################################

               page
getmark        ldx          fcbptr                  ;get index to open file control block.
               ldy          #c.mark                 ;and index to user's mark parameter.
gmark1         lda          fcb+fcbmark,x           ;transfer current position
               sta          (par),y                 ; to user's parameter list.
               inx
               iny
               cpy          #c.mark+3               ;have all three bytes been transferred.
               bne          gmark1                  ;branch if not...
               clc                                  ;no errors.
               rts
*
*
errmeof        lda          #posnerr                ;report invalid position.
               sec
               rts
*
*
setmark        ldy          #c.mark+2               ;get index to user's desired position
               ldx          fcbptr                  ;and file's control block index
               inx                                  ;(bump by 2 for index to hi eof)
               inx
               sec                                  ;indicate comparisons are necessary
smark1         lda          (par),y                 ;move it to 'tpos'
               sta          tposll-c.mark,y
               bcc          smark2                  ;branch if we already know mark<eof.
               cmp          fcb+fcbeof,x            ;carry or z flag must be clear to
qualify.
               bcc          smark2                  ;branch if mark qualifies for sure.
               bne          errmeof                 ;branch if mark>eof
               dex
*
smark2         dey                                  ;prepare to move/compare next lower
byte of mark.
               tya                                  ;test for all bytes moved/tested.
               eor          #c.mark-1               ;to preserve carry status.
               bne          smark1                  ;branch if more.
*
*
rdposn         equ          *
               ldy          fcbptr                  ;first test to see if new position is
               lda          fcb+fcbmark+1,y         ; within the same (current) data block.
               and          #$fe
               sta          scrtch
               lda          tposlh                  ;get middle byte of new position
               sec
               sbc          scrtch
               sta          scrtch
               bcc          typmark                 ;branch if possibly l.t. current
position
               cmp          #2                      ;must be within 512 bytes of beginning
of current
               bcs          typmark
```

```
                lda         tposhi                  ;now make sure were talkin about
                cmp         fcb+fcbmark+2,y         ; the same 64k chunk!
                bne         typmark                 ;branch if we aren't.
                jmp         svmark                  ;if we is, adjust fcb, posptr and
return.
*
typmark         equ         *                       ;now find out which type
                lda         fcb+fcbstyp,y           ; of file we're positioning on.
                beq         ferrtyp                 ;there is no such type as zero, branch
never!
                cmp         #4                      ;is it a tree class file?
                bcc         trepos                  ;yes, go position
                jmp         dirmark                 ; no, test for directory type.
*
*
ferrtyp         ldy         #fcbptr                 ;clear illegally typed fcb entry
                sta         fcb+fcbrefn,y
                lda         #badrefnum              ;tell em there is no such file
                sec
                rts
*
                page
trepos          equ         *                       ;use storage type as number of index
levels
                lda         fcb+fcbstyp,y           ; (since 1=seed, 2=sapling, and 3=tree)
                sta         levels
                lda         fcb+fcbstat,y           ;must not forget previous data.
                and         #datmod                 ;therefore, see if previous data was
modified
                beq         posnew1                 ; then disk must be updated.
                jsr         wfcbdat                 ;go write current data block.
                bcs         poserr                  ;return any error encountered.
*
posnew1         ldy         fcbptr                  ;test to see if current
                lda         fcb+fcbmark+2,y         ; index block is going to be usable...
                and         #$fe                    ; or in other words-
                sta         scrtch                  ; is new position within 128k of the
beginning
                lda         tposhi                  ; of current sapling level chunk.
                sec
                sbc         scrtch
                bcc         posnew2                 ;branch if a new index block is also
needed
                cmp         #2                      ;new position is > than begining of
old. is it within 128k?
                bcs         posnew2                 ;branch if not.
                ldx         levels                  ;is the file we're dealing with a seed?
                dex
                bne         datlevel                ;no, use current indexes.
tstiny          lda         tposlh                  ;is new position under 512?
                lsr         a
                ora         tposhi
                bne         noidxdat                ;no, mark both data and index block as
un-allocated.
                lda         fcb+fcbfrst,y           ;first block is only block and it's
data!
                sta         bloknml
                lda         fcb+fcbfrst+1,y         ;(high block address)
                jmp         rnewpos                 ;go read in block and set appropriate
statuses.
*
                page
posnew2         equ         *                       ;gota check to see if previous
```

```
                lda     fcb+fcbstat,y           ; index block was modified.
                and     #idxmod
                beq     posnidx                 ;read in over it if current is up to
date.
                jsr     wfcbidx                 ;go update index on disk (block addr in
fcb)
                bcs     poserr
posnidx         ldx     levels                  ;before reading in top index, check to
be sure
                cpx     #3                      ; that there is a top index...
                beq     posindex                ;branch if file is full blown tree.
                lda     tposhi                  ;is new position within range of a
                lsr     a                       ; sapling file (l.t. 128k)?
                php                             ;anticipate no good.
                lda     #topalc+idxalc+datalc   ;(to indicate no level is allocated for
new position.)
                plp                             ;z flag tells all...
                bne     nodata                  ;go mark 'em all dummy.
                jsr     clrstats                ;go clear status bits 0,1,2 (index/data
alloc status).
                dex                             ;(unaffected since loaded above) check
for seed
                beq     tstiny                  ;if seed, check for position l.t.
512...
                jsr     rfcbfst                 ;go get only index block
                bcs     poserr                  ;branch if error
                ldy     fcbptr                  ;save newly loaded index block's
address
                lda     bloknml
                sta     fcb+fcbidxb,y
                lda     bloknmh
                sta     fcb+fcbidxb+1,y
                bcc     datlevel                ;branch always...
*
poserr          equ     *                       ;carry always set when branched to
                rts
*
posindex        jsr     clrstats                ;clear all allocation requirements for
previous position
                jsr     rfcbfst                 ;get highest level index block.
                bcs     poserr
                lda     tposhi                  ;then test for a sap level index block
                lsr     a
                tay
                lda     (tindx),y
                inc     tindx+1
                cmp     (tindx),y               ;(both hi and lo will be zero if no
index exists)
                bne     saplevel
                cmp     #0                      ;are both bytes zero?
                bne     saplevel
                dec     tindx+1                 ;don't leave wrong pointers laying
around!
noidxdat        lda     #idxalc+datalc          ;show neither index or data block
allocated.
                jmp     nodata
*
                page
saplevel        sta     bloknml                 ;read in next lower index block
                lda     (tindx),y               ;(hi address)
                sta     bloknmh
                dec     tindx+1
                jsr     rfcbidx                 ;read in sapling level
```

```
                bcs             poserr
datlevel        lda             tposhi                  ;now get block address of data block
                lsr             a
                lda             tposlh                  ;( if there is one )
                ror             a
                tay
                lda             (tindx),y               ;data block address low
                inc             tindx+1
                cmp             (tindx),y
                bne             posnew3
                cmp             #0
                bne             posnew3
                lda             #datalc                 ;show data block as never been
allocated
                dec             tindx+1
*
nodata          ldy             fcbptr                  ;set status to show whats missin'
                ora             fcb+fcbstat,y
                sta             fcb+fcbstat,y
                lsr             a                       ;throw away bit that says data block
un-allocated
                lsr             a                       ; cuz we know that. carry now indicates
if index block
                jsr             zipdata                 ; also is invalid and needs to be
zeroed (carry undisturbed)
                bcc             svmark                  ;branch if index block doesn't need
zippin.
*
***************** see rev note #40 ***************************
*
                jsr             zeroindex               ; go zero index block in user's i/o
buffer
                jmp             svmark
*
zeroindex       lda             #0
                tay
zipidx          sta             (tindx),y               ; zero out the index half of the user's
                iny                                     ; i/o buffer.
                bne             zipidx
                inc             tindx+1
zpidx1          sta             (tindx),y
                iny
                bne             zpidx1
                dec             tindx+1                 ;restore proper address
                rts                                     ; that's all.
************************************************************
*
zipdata         lda             #0                      ; also is invalid and needs to be
zeroed.
                tay
zipdat0         sta             (datptr),y              ;zero out data area
                iny
                bne             zipdat0
                inc             datptr+1
zpdat1          sta             (datptr),y
                iny
                bne             zpdat1
                dec             datptr+1
                rts
*
                page
*
posnew3         sta             bloknml                 ;get data block of new position
```

```
                lda     (tindx),y               ;(hi address)
                dec     tindx+1
rnewpos         sta     bloknmh
                jsr     rfcbdat
                bcs     pritz                   ;return any error
                jsr     clrstats                ;show whole chain is allocated
svmark          ldy     fcbptr                  ;update position in file control block
                iny
                iny
                ldx     #2
svmrk1          lda     fcb+fcbmark,y           ; remember oldmark in case
                sta     oldmark,x               ; calling routine fails later
                lda     tposll,x
                sta     fcb+fcbmark,y
                dey
                dex                             ;move 3 byte position marker
                bpl     svmrk1
*
                clc                             ;last, but not least, set up
                lda     datptr                  ; indirect address to buffer page
pointed
                sta     posptr                  ; to by the current position marker.
                lda     tposlh
                and     #1
                adc     datptr+1
                sta     posptr+1
pritz           rts                             ;carry set indicates error!
*
*
clrstats        ldy     fcbptr                  ;clear allocation states for data block
                lda     fcb+fcbstat,y           ; and both levels of indexes.
                and     #$ff-topalc-idxalc-datalc
                sta     fcb+fcbstat,y           ;this says that either they exist
currently
                rts                             ; or that they're unnecessary for
current position.
*
                page
*
dirmark         cmp     #dirtyp                 ;is it a directory?
                beq     dirpos                  ;yes...
                lda     #cpterr                 ;no, there is a compatablity problem-
                jsr     syserr                  ;the damn thing should of never been
opened!
*
dirpos          lda     scrtch                  ;recover results of previous
subtraction.
                lsr     a                       ;use difference as counter as to how
many
                sta     cntent                  ; blocks must be read to get to new
position.
                lda     fcb+fcbmark+1,y         ;test for position direction.
                cmp     tposlh                  ;carry indicates direction...
                bcc     dirfwrd                 ; if set, position forward.
dirvrse         ldy     #0                      ;otherwise, read directory file in
reverse order.
                jsr     dirpos1                 ;read previous block.
                bcs     drposerr                ;branch if anything goes wrong.
                inc     cntent                  ;count up to 128
                bpl     dirvrse                 ;loop if there is more blocks to pass
over.
                bmi     svmark                  ;branch always.
*
```

```
dirfwrd         ldy         #2                      ;position is forward from current
position.
                jsr         dirpos1                 ;read next directory block.
                bcs         drposerr
                dec         cntent
                bne         dirfwrd                 ;loop if position not found in this
block.
                beq         svmark                  ;branch always.
*
dirpos1         lda         (datptr),y              ;get link address of previous or
                sta         bloknml                 ; next directory block.
                iny                                 ;but first be sure there is a link.
                cmp         (datptr),y
                bne         dirpos2                 ;branch if certain link exists
                cmp         #0                      ;are bothe link bytes 0?
                bne         dirpos2                 ;nope, just happen to be the same
value.
                lda         #eoferr                 ;something is wrong with this directory
file!
drposerr        sec                                 ;indicate error
                rts
*
dirpos2         lda         (datptr),y              ;(high order block address)
                sta         bloknmh
*
* drop into rfcbdat (read file's data block)
*
                page
*
* note: for directory positioning, no optimization has been done since
*   since directory files will almost always be less than 6 blocks.
*   if more speed is required or directory type files are to be used
*   for other purposes requiring more blocks, then the recommended
*   method is to call rfcbdat for the first block and go directly to
*   device (via jmp (iounitl)) handler for subsequent accesses.
* also note that no checking is done for read/write enable since a
*   directory file can only be opened for read access.
*
rfcbdat         lda         #rdcmd                  ;set read command.
                sta         dhpcmd
                ldx         #datptr                 ;use x to point at address of data
buffer
                jsr         fileio1                 ;go do file input.
                bcs         rdaterr                 ;return any error
                ldy         fcbptr
                lda         bloknml
                sta         fcb+fcbdatb,y           ;save block number just read in fcb.
                lda         bloknmh
                sta         fcb+fcbdatb+1,y
rdaterr         rts                                 ;carry set indicates error
*
rfcbidx         lda         #rdcmd                  ;prepare to read in index block.
                sta         dhpcmd
                ldx         #tindx                  ;point at address of current index
buffer
                jsr         fileio1                 ;go read index block.
                bcs         rdfcberr                ; report error
                ldy         fcbptr
                lda         bloknml
                sta         fcb+fcbidxb,y           ;save block address of this index in
fcb.
                lda         bloknmh
                sta         fcb+fcbidxb+1,y
```

```
                clc
rdfcberr        rts
*
wfcbfst1        lda         #wrtcmd                 ;set write mode for device
                dfb         $2c                     ;skip next 2 bytes for pseudo-branch
always to rwfst
*
rfcbfst         lda         #rdcmd                  ;set read mode for device
rwfst           pha                                 ;save command
                lda         #fcbfrst
                ora         fcbptr                  ;add offset to fcbptr
                tay
                pla
                ldx         #tindx                  ;read block into index portion of file
buffer
*
* drop into dofileio
*
dofileio        sta         dhpcmd                  ;save command.
                lda         fcb,y                   ;get disk block address from fcb.
                sta         bloknml                 ;block zero not legal.
                cmp         fcb+1,y
                bne         fileio
                cmp         #0                      ;are both bytes zero?
                bne         fileio                  ;no, continue with request.
                lda         #alcerr                 ;otherwise report allocation error.
                jsr         sysdeath                ;never returns...
*
                page
fileio          lda         fcb+1,y                 ;get high address of disk block
                sta         bloknmh
fileio1         php                                 ;no interupts from here on out.
                sei
                lda         0,x                     ;get memory address of buffer from
                sta         dbufpl                  ; zero page pointed to by
                lda         1,x                     ; the x-register.
                sta         dbufph                  ; and pass address to device handler
                ldy         fcbptr
                lda         fcb+fcbdevn,y           ;of course having the device number
                sta         devnum                  ; would make the whole operation more
meaningful...
                lda         #$ff                    ;also,
                sta         ioaccess                ; set to indicate reg call made to dev
handler.
                lda         devnum                  ;transfer the device number for
dispatcher to convert to unit number.
                sta         unitnum
                lda         #0
                sta         serr                    ; clear global error value
                jsr         dmgr                    ; call the driver
                bcs         fioerr                  ;branch if error
                plp                                 ;restore interupts.
                clc
                rts
fioerr          plp                                 ;restore interupts
                sec
                rts
*
wfcbfst         equ         *                       ; first update the bitmap
                jsr         upbmap
                jmp         wfcbfst1                ;and go write file's first block!
*
*
```

```
wfcbdat         ldx          #datptr
                lda          #fcbdatb                ;point at memory address with x and
disk address with y.
                ora          fcbptr                  ;add offset to fcbptr
                tay                                  ;and put it in y.
                lda          #wrtcmd                 ;write data block.
                jsr          dofileio
                bcs          filioerr                ; report any errors
                lda          #$ff-datmod             ;mark data status as current.
                jmp          fcbupdat
*
wfcbidx         equ          *
                jsr          upbmap                  ; go update bitmap
                ldx          #tindx                  ; point at address of index buffer
                lda          #fcbidxb                ; and block address of that index
block.
                ora          fcbptr
                tay
                lda          #wrtcmd
                jsr          dofileio                ;go write out index block.
                bcs          filioerr                ; report any errors
                lda          #$ff-idxmod             ;mark index status as current.
fcbupdat        ldy          fcbptr                  ;change status byte to
                and          fcb+fcbstat,y           ; reflect successful disk file update.
                sta          fcb+fcbstat,y           ;(carry is unaffected)
filioerr        rts
*
*
                page
open            jsr          findfile                ;first of all look up the file...
                bcc          open0
                cmp          #badpath                ;is an attempt to open a root
directory?
                bne          erropn                  ;no, pass back error
*
open0           jsr          tstopen                 ;find out if any other files are
writing
                bcc          open1                   ; to this same file. (branch if not)
errbusy         lda          #filbusy                ;report shared access not allowed.
erropn          sec
                rts                                  ;return error.
*
*****************  see rev note #47  ***************
*errcmpat lda #cpterr ;report file is incompatable!
wrgstype        lda          #typerr                 ; report file is of wrong storage type!
************************************************
                sec
                rts
*
open1           ldy          fcbptr                  ;get address of first free fcb found
                lda          fcbflg                  ;this byte indicates that an free fcb
found if non zero.
                bne          asgnfcb                 ; is available for use.
                lda          #fcbfull                ;report fcb full error.
                sec
                rts
*
asgnfcb         ldx          #$1f                    ;assign fcb, but first
                lda          #0                      ; clean out any old rubbish left
around...
clrfcb          sta          fcb,y
                iny
                dex
```

```
                bpl         clrfcb
                lda         #fcbentn                ;now begin claim by moving in file info
                tax                                 ; use x as source index
                ora         fcbptr                  ; and y as destination (fcb)
                tay
fcbownr         lda         d.dev-1,x               ; move ownership information.
                sta         fcb,y                   ;note: this code depends upon the
defined
                dey                                 ; order of both the fcb and directory
entry
                dex
                bne         fcbownr                 ; buffer (d.). beware of changes!!!
*************
                lda         dfil+d.stor             ;get storage type.
                lsr         a                       ;strip off file name length.
                lsr         a
                lsr         a                       ;(by dividing by 16)
                lsr         a
                tax                                 ;save in x for later type comparison
                sta         fcb+fcbstyp,y           ; and in fcb for future access.
                lda         dfil+d.attr             ;get files attributes and
                and         #readen+writen          ; use it as a default access request.
                cpx         #dirtyp                 ;if directory, don't allow write enable
                bne         svattr1
                and         #readen
svattr1         sta         fcb+fcbattr,y
                and         #writen                 ;check for write enabled requested.
                beq         open3                   ;branch if read only open.
                lda         totent                  ;otherwise, be sure no one else is
reading same
                bne         errbusy                 ; file (set up by tstopen).
*
open3           cpx         #tretyp+1               ;is it a tree type file?
                bcc         open4                   ;test for further compatablity. it must
                cpx         #dirtyp                 ; be either a tree or a directory.
******************  see rev note #47  *************
* bne errcmpat ;report incompatable.
                bne         wrgstype                ; report file is of wrong storage type.
************************************************
open4           ldx         #6                      ;move address of first block of file,
open5           sta         bloknmh                 ; end of file, and current usage count.
                lda         fcbptr
                ora         ofcbtbl,x               ;this is done via a translation
                tay                                 ; table between directory info and fcb.
                lda         dfil+d.frst,x
                sta         fcb,y
                dex                                 ;has all info been moved?
                bpl         open5
                sta         bloknml                 ;last loop stored hi addr of frst
block.
                ldy         fcbptr
                lda         cntent                  ;this was set up by
'tstopen'.............
                sta         fcb+fcbrefn,y           ;claim fcb for this file.
*
                jsr         alcbuffr                ;go allocate buffer in memtables.
                bcs         erropen2                ;give up if any errors occurr.
                jsr         fndfcbuf                ;returns address of bufs in data &
index pointers.
                lda         level                   ;mark level at which
                sta         fcb+fcblevl,y           ; file was opened
                lda         fcb+fcbstyp,y           ;file must be positioned to beginning.
                cmp         #tretyp+1               ;is it a tree file?
```

```
                bcs         opndir                  ;no, assume it's a directory.
                lda         #$ff                    ;fool the position routine into giving
                sta         fcb+fcbmark+2,y         ; a valid position with preloaded data,
etc.
                ldy         #2                      ;set desired position to zero.
                lda         #0
opnpos1         sta         tposll,y
                dey
                bpl         opnpos1
                jsr         rdposn                  ;let tree position routine do the rest.
                bcc         opendone                ;branch if successful.
*
                page
erropen2        pha                                 ;save error code.
                ldy         fcbptr                  ;return buffer to free space
                lda         fcb+fcbfbuf,y
                beq         erropen3                ;branch if no bufnum.
                jsr         relbuffr                ;doesn't matter if it was never
allocated.
                ldy         fcbptr                  ;since error was encountered before
file
erropen3        lda         #0                      ; was successfully opened, then
                sta         fcb+fcbrefn,y           ; its necessary to release fcb.
                pla
                sec
                rts
*
opndir          jsr         rfcbdat                 ;read in first block of directory file.
                bcs         erropen2                ;return any error after freeing buffer
& fcb
opendone        ldx         vcbptr                  ;index to volume control block
                inc         vcb+vcbopnc,x           ;add 1 to the number of files currently
open.
                lda         vcb+vcbstat,x           ;and indicate that this volume has
                ora         #$80                    ; at least 1 file active.
                sta         vcb+vcbstat,x
                ldy         fcbptr                  ;index to file control block
                lda         fcb+fcbrefn,y           ;return reference number to user.
                ldy         #c.outref
                sta         (par),y
                clc                                 ;indicate successful open!
                rts                                 ;all done...
                page
tstopen         lda         #0
                sta         cntent                  ;this temp returns the refnum of a free
fcb.
                sta         totent                  ;this is used as a flag to indicate
file is already open.
                sta         fcbflg                  ;this is a flag to indicate a free fcb
is available.
*
tstopn1         tay                                 ;index to next fcb
                ldx         fcbflg                  ;test for free fcb found.
                bne         tstopn2                 ;branch if already found.
                inc         cntent
tstopn2         lda         fcb+fcbrefn,y           ;is this fcb in use?
                bne         chkactv                 ;branch if it is.
                txa                                 ;if not, should we claim it?
                bne         tsnxfcb                 ;branch if free fcb already found.
                sty         fcbptr                  ;save index to free fcb.
                lda         #$ff                    ;set fcb flag to indicate free fcb
found.
                sta         fcbflg
```

```
                bne         tsnxfcb                 ;branch always to test next fcb.
*
chkactv         tya                                 ;add offset to index to ownership info.
                ora         #fcbentn
                tay                                 ;put it back in y.
                ldx         #fcbentn                ;index to directory entry owner info.
whowns          lda         fcb,y
                cmp         d.dev-1,x               ;all bytes must match to say that its
                bne         tsnxfcb                 ; the same file again.
                dey                                 ;index to next lower bytes.
                dex
                bne         whowns                  ;loop to check all owner info.
                inc         totent                  ;file is already open,
                lda         fcb+fcbattr,y           ;now see if its already opened for
write
                and         #writen                 ; if so, report file busy (with carry
set).
                beq         tsnxfcb                 ;branch if this file is read access
only.
                sec
                rts
*
tsnxfcb         tya                                 ;calc position of next fcb.
                and         #$e0                    ;first strip any posible index offsets.
                clc
                adc         #$20                    ;bump to next fcb.
                bne         tstopn1                 ;branch if more to compare.
                clc                                 ;report no conflicts.
                rts
*

; ###############################################################################################
; #   END OF FILE:  POSN.OPEN
; #   LINES      :  571
; #   CHARACTERS :  33547
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.PRODIR.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  PRODIR
; ###############################################################################################

                skp          1
* this is the volume directory and allocation
* "bit map" for /prodos on a disk ii.
                skp          1
                dfb          $00,$00,$03,$00,$f6,$50,$52,$4f
                dfb          $44,$4f,$53,$00,$00,$00,$00,$00
                dfb          $00,$00,$00,$00,$75,$00,$00,$00
                dfb          $00,$00,$00,$00,$00,$00,$00,$00
                dfb          $00,$00,$c3,$27,$0d,$00,$00,$06
                dfb          $00,$18,$01,$00,$00,$00,$00,$00
                ds           $1d0,0
                dfb          2,0,4
                ds           $1fd,0
                dfb          3,0,5
                ds           $1fd,0
                dfb          6
                ds           $1ff,0
                dfb          1
                ds           $22,$ff
                ds           $1dd,0
                dfb          $00

; ###############################################################################################
; #   END OF FILE:  PRODIR
; #   LINES      :  21
; #   CHARACTERS :  873
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.PROLDR.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  PROLDR
; ###############################################################################################

                sbtl         'prodos kernel loader'
*
********* see rev note #45/57 **********
*
*proldr equ *
* ldx #$ff ; initialize main stack pointer
* txs  ; to $ff.
***********************************
*
****************** see rev note #57 *********
*
                jmp          prostart                  ; normal boot entry point....
                inc          setuprts                  ; setup and rts entry point....
prostart        equ          *
*****************************************
                lda          unit                      ;get boot device number.
                sta          bunit                     ;save it for later 'prefix'
                jsr          greet                     ;put up greeting message.
                ldx          itpldr                    ;move interpreter loader to $800
                ldy          itpldr+1
                jsr          reloc
                bcs          m48k                      ;branch if error
*
proldr1         equ          *
                ldy          #0
                lda          #$ff                      ;make sure there is
                sta          $bfff                     ; at least 48k.
                eor          $bfff
                sec
*********** branch widened- see rev note #62 ***********
                bne          nogood0                   ;branch if not.
                sta          $bfff                     ;try again. once may have been luck!
                lda          $bfff
*********** branch widened- see rev note #62 ***********
                bne          nogood0                   ;hitch to hitch
                lda          romin
                jsr          whchrom                   ;get preliminary system configuration
                bcs          m48k                      ;branch if apple /// emulation
                lda          apple                     ;test for 48k configuration
                and          #$20                      ;by testing for 64k plus.
                bne          m64k                      ;branch if >48k
m48k            jmp          nogood                    ;must have at least 64k
m64k            ldx          ver64k                    ;now move/relocate whatever we got.
                ldy          ver64k+1
                jsr          reloc
                lda          kversion                  ;get current revision number
                sta          xdosver                   ;  and save it for directory use
*********** label added    see rev note #62 ***********
nogood0         equ          *                         ;hitch to nogood jump
                bcs          nogood1
                lda          romin
                ldx          $fbb3                     ;look for //e family
                cpx          #6
```

```
                bne         id1
                lda         #$e0                    ;phylum check on high 2 bits
                bit         $fbc0                   ;another approved location
                php                                 ;save the results from the bit
                lda         apple
                and         #$37                    ;mask off bits 7,6 and 3
                plp                                 ;get results back
                bvc         set3                    ;//c or //x
                bmi         set7                    ;branch if //e
set3            php                                 ;save the results from the bit again
                ora         #$8                     ;set bit 3 on
                plp
                bpl         mach2                   ;branch if //c
                ora         #$40
                bpl         savemach                ;always...
mach2           inc         cflag-licode+lodintrp   ; make it easy to see if we're on //c
later
                bvs         savemach
set7            ora         #$80
savemach        sta         apple
*
************ see rev note #50 **********
*
                lda         romin                   ; enable rom for cortland id routine.
                sec                                 ; carry will determine if cortland or
not.
                jsr         $fe1f                   ; rts in all a //'s prior to cortland
                bcs         itsaiie                 ; branch if really a //e.
                inc         cortland                ; set loader's cortland flag!
*
************ see rem note #62 **********
*
* if setuprts is zero, zero out os.boot for appletalk.
*  (setuprts reflects whether we're prodos 8 regular or running with
*  the p16 shell.
*
                lda         setuprts
                bne         itsp8
**** the os.boot value is hardwired to $e100bd ****
****  and the dfb defines a 65816 sta absolute long ****
                dfb         $8f,$bd,$00,$e1         ;zero is already in acc
itsp8           equ         *
*
***************************************
*
itsaiie         equ         *
***************************************
id1             equ         *
                lda         bunit                   ;place boot devnum in globals.
                sta         bbunit
                sta         lstdev
                jsr         devsrch                 ;finish setting up globals.
                lda         bbunit
                sta         lstdev
                jsr         lc1in
                ldx         tclkstuff               ;and set up clock.
                ldy         tclkstuff+1
                jsr         reloc
nogood1         bcs         nogood2                 ;give up any time we got problems.
*
*  dispatcher 1 must go in bank 2 of language card
*    in a 64k or larger system.
*
```

```
                lda         machid
                and         #$20
                cmp         #$20
                bne         chkram                  ;not 64k or greater
                lda         #>calldisp
                sta         jspare+1                ;put dispatcher relocater address
                lda         #<calldisp              ;  into jspare vector.
                sta         jspare+2
                lda         ramin2
                lda         ramin2                  ;switch in bank 2
                ldx         dspstuff                ; relocate dispatcher and rwts
                ldy         dspstuff+1              ; drivers to main lc, b2.
                jsr         reloc
******************* see rev note #38 *********************
* -------------------- see rev note 20 ------------------------
* ldx busstuff ;set up bus driver
* ldy busstuff+1
* jsr reloc
* -------------------------------------------------------------
************************************************************
                lda         #$ee                    ;nonsense byte to distinguish bank 2
                sta         $d000
                jsr         lc1in                   ;switch bank 1 back in
                bcc         chkram
nogood2         jmp         nogood
*
* test for 128k so ram disk can be installed
*
chkram          lda         machid
                and         #$30
                eor         #$30
                ifeq        os-prodos
                bne         noramdsk                ;branch if less than 128k system
                fin
                ifeq        os-ednet
                beq         ramdsk                  ; branch if its a 128k machine...
                sta         romin                   ; otherwise system death since
                jsr         home                    ; ednet requires 128k.
                ldy         #no.aux.msg.len-1
showit          lda         no.aux.msg,y            ;
                sta         qscrn,y                 ;
                dey                                 ;
                bpl         showit                  ;
die.here        jmp         die.here                ; and hang...
*
                msb         on
no.aux.msg      equ         *
                asc         '*** ednet requires 128k ram memory  ***'
no.aux.msg.len  equ         *-no.aux.msg
                msb         off
*
*
ramdsk          equ         *
                fin
*
*
************ see rev note #45 *************
*
                ldx         #$ff                    ; x used to init aux sp to $ff.
                php                                 ; save interrupt status
                pla                                 ; in a reg.
                sei                                 ; no interrupts for safety sake.
                sta         $c009                   ; swap in aux lc & zp & stack.
```

```
                stx         $101                    ; init aux sp to $ff.
                sta         $c008                   ; back to main lc, zp, and stack.
                pha                                 ; restore
                plp                                 ; interrupt status.
****************************************
                sta         int3rom                 ;make sure internal slot 3 rom is in
                jsr         ram.1                   ;go install ramdisk
*
*
*  now check for interrupt vector.  if vector <$d000 then we
*    have new roms and should re-point vector in language card to
*    rom vector and set a flag byte.  in vector is >$d000, reset
*    flag byte and do nothing.
*
noramdsk        equ         *
                lda         romin1                  ;switch in rom
                ldy         irqvect
                ldx         irqvect+1               ;get hi byte of irq vector
*
*--------------------- see rev note #29 ------------------------
*
* the jsr lc1in was moved here from after the bcs chk4card so the
* sta irqflag is written to the proper bank.
*
                jsr         lc1in                   ;put the lc,b1 for r/w back now!
*
                cpx         #$d0                    ;is it >$d000 (old roms)
                lda         #0                      ;anticipate not
                bcs         chk4card                ;  but branch if they are old roms
                sta         $c009                   ;swap aux lc, zpg and stack
                lda         #$ff                    ;set aux stack pointer at $ff
                sta         $101                    ;  while we're here.
                stx         irqvect+1
                sty         irqvect                 ;save rom vector in aux lang. card
                sta         $c008                   ;swap in main lc, zpg and stack
                stx         irqvect+1
                sty         irqvect                 ;save rom vector in main lang. card
                lda         #1                      ;set irqflag to show new roms
chk4card        sta         irqflag
*
********** see rev note #61 ***************
********** see rev note #50 ***************
*
                lda         #0
                sta         cortflag                ;assume we're not on a cortland
                lda         cortland                ; are we running on a cortland?
                beq         nocort                  ; if not branch, and muck w/slot 3!
                inc         cortflag                ;make it a one if we're on cortland
                jmp         docard
nocort          equ         *
*
****************************************
*
* ------------------- see rev note 19 ------------------------------
*  check for a rom in slot 3.  switch in internal $c300 firmware
*    if no rom seen.
*
                sta         int3rom                 ;start with internal firmware switched
in
                lda         sltbyt                  ;get slots rom pattern
                and         #8                      ;mask off all but slot 3
                bne         isromin3                ; branch if there is rom in slot three.
                jmp         noslot3rom              ; continue with boot....
```

```
isromin3        equ         *
*
* we've seen a rom in slot 3.  is it an external, identifiable
*   80-col card with interrupt routines?  if so, enable it.  if not,
*   switch in the internal $c300 firmware.
*
                sta         slot3rom                ;switch in slot 3 rom
                lda         $c305                   ;1st generic terminal card id byte
                cmp         #$38
                bne         hitswtch                ;branch if not a terminal card.
                lda         $c307                   ;2nd generic terminal card id byte
                cmp         #$18
                bne         hitswtch                ;branch if not a terminal card.
                lda         $c30b                   ;3rd generic terminal card id byte
                cmp         #$01
                bne         hitswtch                ;branch if not a terminal card.
                lda         $c30c                   ;is it an apple 80-col card compatible?
                and         #$f0                    ;mask off lo nibble
                cmp         #$80                    ; and check for $8n
                bne         hitswtch                ;branch if not an 80-col card.
                lda         machid                  ;get the machine id
                and         #$c8
                cmp         #$c0                    ;is is a //+?
                beq         docard                  ;branch if it is.
                lda         $c3fa                   ;check for interrupt handler routine
                cmp         #$2c                    ; in the magic $c3fa spot.
                beq         docard                  ;branch if interrupt handler is there!
hitswtch        sta         int3rom                 ;internal $c300 rom
*
* verify that the card in the aux slot is actually there.
*
                sta         $c001                   ;80-store
                sta         $c055                   ;page2
                lda         #$ee
                sta         $400
                asl         a
                asl         $400
                cmp         $400
                bne         maybe                   ;branch if not there
                lsr         a
                lsr         $400
                cmp         $400
maybe           sta         $c054                   ;main memory
                sta         $c000                   ;80-store off
                beq         docard                  ;branch if card is there
                lda         machid                  ;get machine id byte
                and         #$fd                    ;mask off 80-col bit
                bne         docard1                 ;always..
*
* ok, the card's good.  leave it enabled and update the machid.
*
docard          lda         machid                  ;get machine id byte
                ora         #2                      ;turn bit 2 on to show 80-col card
docard1         sta         machid
* ---------------------------------------------------------------
*
******** see rev note #50 **********
*
noslot3rom      equ         *
                lda         cortland                ; are we running on a cortland?
                beq         notcortland             ; branch if not.
                lda         #$4c                    ; enable clock routine by
```

```
                sta         $bf06                   ; putting a jmp in front of clock
vector.
                ldx         cclkstuff               ; now set up for relocating
                ldy         cclkstuff+1             ; the cortland clock driver.
                jsr         reloc                   ; and relocate it.
******* see rev note #58 ********
                lda         machid                  ; denote clock present in machid byte!
                ora         #1                      ; bit 0 set to 1.
                sta         machid                  ;
notcortland     equ         *
*********************************
*
******************** see rev note #57 **********
*
                lda         setuprts                ; get value of setup entry point
flag...
                beq         norts                   ; branch if normal boot...
                lda         romin                   ; make sure the rom is in for
consistancy...
                rts                                 ; return to the caller at the setup
entry point. ($2003)
setuprts        db          0                       ; 0->normal boot; <>0->return.
norts           equ         *
***********************************************
*
* now set prefix to boot device.
*
                jsr         gopro                   ; first 'online'(was labled
bootpfx,#en3)
                dfb         onlyne
                dw          olparm
                bne         nogood                  ;branch if problems.
                lda         pbuf+1                  ;get volume name length.
                and         #$f                     ;strip devnum.
                beq         nogood                  ;branch if error.
                clc                                 ;add 1 for leading '/'
                adc         #1
                sta         pbuf                    ;save prefix length.
                lda         #$2f                    ;place leading '/' in pbuf.
                sta         pbuf+1
                jsr         gopro                   ;set prefix.
                dfb         prefix
                dw          prparm
                bne         nogood                  ;branch if problems.
                tax                                 ;acc = 0 after successful mli call
                stx         dst
                ldy         #2                      ;read directory into buffer starting
                lda         #$c                     ;  at $c00
rdbdir          sta         dst+1
                sta         dbbufr+1                ;(using a pointer in zero page also)
                sty         dbblok
                stx         dbblok+1
                jsr         gopro
                dfb         idbrd                   ;block read
                dw          dbprms
                bne         nogood
                ldy         #3                      ;get next block number from link
                lda         (dst),y
                tax
                dey
                ora         (dst),y                 ;if both bytes are the same i.e. 0, 0
                beq         edbrd                   ; then no more blocks of directory
                lda         (dst),y
```

```
                tay
                lda         dst+1
                clc
                adc         #2                      ;add $200 to buffer pointer until
buffer
                cmp         #$14                    ; points past $13ff.
                bcc         rdbdir                  ;if ok, read next block
edbrd           equ         *
                jmp         lodintrp                ;all is well, load interpreter!!!
                skp         2
qscrn           equ         $5a9
                skp         2
nogood          sta         romin                   ;make sure rom is there.
                jsr         home                    ;clear video.
                ldy         #meslen                 ;print message centered on screen.
prmess          lda         errmess,y
                sta         qscrn,y
                dey
                bpl         prmess
hang            jmp         hang
*
                msb         on
meslen          equ         30
errmess         asc         "RELOCATION/  CONFIGURATION ERROR" "
*
olparm          dfb         $02
bunit           dfb         $60
                dw          pbuf+1
                skp         1
prparm          dfb         $01
                dw          pbuf
                skp         1
dbprms          dfb         $03
bbunit          dfb         $00
dbbufr          dw          $00
dbblok          dw          $00
*
*
                page
czero           equ         $00
cmove           equ         $01
cradd           equ         $03
crloc           equ         $04
cdone           equ         $ff
at.load.addr    equ         $4000                   ; atalk file will load here. (*#en3*)
                skp         1
itpldr          dw          tablitp                 ;table for relocation of interp loader.
ver64k          dw          tabl64
tclkstuff       dw          rlclk64
dspstuff        dw          dsp64
*
********* see rev note #50 ********
*
cclkstuff       dw          cortclock               ; address of cortland clock move table.
cortland        dfb         0                       ; cortland existance flag initially
zero!
***********************************
                ifeq        os-ednet
atrelptr        dw          at.rel.tbl              ; address of atalk driver relocater
table.
                fin
*
*  code move tables are explained in file reloc.
```

```
*
tablitp         dfb          cmove                     ; move interpreter loader code &
tables.
                dw           lodintrp                  ;code is address independent.
                dw           pclen
                dw           licode
*
pg3tbl          dfb          cmove
* ------------------- see rev note 15 -------------------------
auxgo           equ          $3d6                      ;entry point to aux lc driver call
routine
*
                dw           auxgo                     ;was $3f0
                dw           $002a                     ; and $0010
* ------------------------------------------------------------
                dw           pg3stuf
                skp          1
                dfb          cmove
                dw           look
                dw           2
                dw           dst
                skp          1
                dfb          cmove                     ;move 128k test to zero page
                dw           tst128
                dw           end128
                dw           strt128
                skp          1
                dfb          cdone
                page
dsp64           dfb          cmove
                dw           $d100                     ;(lang card bank 2)
                dw           $0300                     ;3 pages
                dw           sel.0
*
****** see rev note #50 *******
*
* ifeq os-prodos
*bus db cmove ; move the atp drivers to mlc, b2.
* dw $d400 ; starting at $d400.
* dw $0c00 ; 3k of it.
* dw bus.0 ; load time address.
* fin
*
* -------------------- see rev note #en1 ----------------------
                ifeq         os-ednet
                dfb          cmove
                dw           $d400                     ; relocate rwts routines to $d400,b2
                dw           $0700                     ; 7 pages of it.
                dw           xrw.0                     ; where it is at load time.
                fin
* ------------------------------------------------------------
                dfb          cdone
*
************** see rev note #en3 ******************
*
                ifeq         os-ednet
at.rel.tbl      db           cmove                     ; move atalk driver table.
                dw           $d000                     ; relocate to $d000, alc,b1(+b2 if
needed)
                dw           $0000                     ; adjusted at run time...
                dw           at.load.addr              ; location of code at load time.
                db           cdone                     ; end of relocation table.
                fin
```

```
*************************************************
               page
*************************************************
* the following table is for moving the 64k version of
* the mli to its execution address.
*************************************************
*
******************** see rev note #38 *********************
*
tabl64         dfb          cmove                    ; relocate the interrupt/break/reset
               dw           inthandler               ; handler and associated vectors.
               dw           $0065                    ; number of bytes to relocate.
               dw           mli.3                    ; source address of code to relocate.
*************************************************************
               skp          2
               dfb          cmove                    ;move preset 64k version of globals
               dw           globals
               dw           $0100
               dw           mli.1
               skp          2
               dfb          czero                    ;clear buffers/workspace
               dw           orig
               dw           $700
               skp          2
               dfb          cmove                    ;move 64k version of mli to language
card.
               dw           orig1                    ; see #45..put rwts in lc b2 to make
mli.2
               ifeq         os-prodos
               dw           $2100                    ; mli length.
               fin
               ifeq         os-ednet
               dw           $2800                    ; mli length + space from moved rwts.
               fin
               dw           mli.2                    ;from address it was loaded at.
               skp          2
*
               ifeq         os-prodos
               dfb          cmove                    ;move 64k version of
               dw           rwts                     ; disk ii routines
               dw           $0700
               dw           xrw.0
               fin
               skp          1
               dfb          cdone                    ;clock moved later.
               skp          2
rlclk64        dfb          cmove                    ;lastly move/relocate thunderclock
               dw           tclk.in                  ; whether needed or not.
               dw           $007d
               dw           tclock.0
               skp          1
               dfb          crloc                    ;adjust slot addresses.
               dw           tclk.in
               dw           $0069
               dw           tclk.in
               dfb          0
clock64        equ          *+2
               dfb          $c1,$c1,$00              ;last changed by devsrch to correct
slot#.
               skp          1
               dfb          cdone                    ;end of relocations.
*
********** see rev note #50 *********
```

```
*
cortclock       dfb         cmove                   ; cortland clock relocating table.
                dw          tclk.in                 ; destination address.
                dw          $7d                     ; length of 125 bytes.
                dw          cclock.0                ; source load address of driver.
                dfb         cdone                   ;
**********************************
                page
errbuf          equ         $56
gointerp        equ         $2000
iscrn           equ         $7a8
idxl            equ         $10
entlen          equ         $c23
*
*
***************** see rev note #56 *************
*
* let's load and jsr to the appletalk configuaration file "atinit"
* if it is found.  if it is not found, just continue with the loading
* and running of the ".system" file.
*
licode          equ         *
                jsr         gopro                   ; make a get file info call to make
                db          getfileinfo             ; atinit file is there and is
                dw          gfi.list                ; of the proper file type.
                bcc         gfi.ok                  ; branch if call successful...
                cmp         #$46                    ; was error file not found?
                beq         loadint                 ; branch if so and continue with interp
load...
                jmp         atloaderr               ; otherwise fatal i/o error in loading
atinit file
gfi.ok          equ         *
                lda         gfi.type                ; now see if atinit file is of proper
type...
                cmp         #$e2                    ; is it the correct file type?
                bne         atloaderr               ; error if wrong file type!
                jsr         gopro                   ;open atinit file.
                dfb         iopen
                dw          atopen                  ; parameter list...
                bne         atloaderr               ; branch if error...
                jsr         gopro                   ;get file's length
                dfb         igeof
                dw          efparm
                bne         atloaderr               ; branch on error...
                lda         eof+2                   ;make sure file will fit.
                bne         atloaderr               ; branch if too big...
                lda         eof+1
                cmp         #$98                    ;max size is 38k
                bcs         atloaderr               ; branch if too big...
                sta         rdlen+1
                lda         eof
                sta         rdlen                   ;read entire file.
                jsr         gopro
                dfb         iread
                dw          rdparm
                bne         atloaderr               ; branch if too big...
                jsr         gopro
                dfb         iclos
                dw          clparm
                bne         atloaderr               ; branch on error....
*
                lda         romin                   ; put rom on line for atinit....
```

```
                jsr         $2000                   ; call the atinit routine to set up
appletalk stuff.
loadint         equ         *
                jmp         goloadint               ; go execute the .system file!
*
atloaderr       equ         *                       ; fatal error if trouble loading atinit
file..
                ldx         aterr                   ; load length of error msg.
aterrlp         lda         aterr,x
                sta         iscrn,x
                dex
                bne         aterrlp
aterrhang       beq         aterrhang               ; hang. couldn't load atinit file....
aterr           str         'UNABLE TO LOAD ATINIT FILE'
gfi.list        equ         *-licode+lodintrp
                db          $a                      ; parameter count.
                dw          atinitname              ; pointer to file name.
                db          0                       ; access
gfi.type        equ         *-licode+lodintrp
                db          0                       ; file type.
                ds          13,0                    ; space for rest of parameters...
*
atopen          equ         *-licode+lodintrp
                db          3                       ; parameter count.
                dw          atinitname              ; pointer to "atinit" file name.
                dw          $1400                   ; address of i/o buffer.
                db          1                       ; reference number hard coded since no
other files.
*
atinitname      equ         *-licode+lodintrp
                str         "atinit" ; name of appletalk config file."
*
goloadint       equ         *-licode+lodintrp
*********************************************
                lda         #$0c                    ;search directory already
                sta         idxl+1                  ; in memory between $c00 & $13ff.
                lda         #4                      ;start 1 entry past header.
                bne         addeln                  ;always...
*
nxentr          lda         idxl                    ;calc next entry posn.
addeln          clc
                adc         entlen                  ;bump to next entry address.
                sta         idxl
                bcs         pgecros                 ;branch if page cross.
                adc         entlen                  ;test for end of block.
                bcc         nocros                  ;branch if definitely not page cross.
                lda         idxl+1
                lsr         a                       ;end of block?
                bcc         nocros                  ;branch if not.
                cmp         #9                      ;end of directory?
                beq         noint1                  ;branch if no interpreter file.
                lda         #4                      ;reset index to first entry in next
block.
                sta         idxl
pgecros         inc         idxl+1                  ;bump to next page.
nocros          ldy         #$10                    ;first off, check file type.
                lda         #$ff                    ;must be prodos sys file.
                eor         (idxl),y
                bne         nxentr                  ;branch if not.
                tay                                 ;else check to see if active.
                lda         (idxl),y
                beq         nxentr                  ;branch if deleted file.
                and         #$f                     ;strip file 'kind'.
```

```
                sta         name                        ;save name's length.
*
*************** see rev note #en3 **************
*
                ifeq        os-ednet
                ldy         atflg                       ; has atalk already been loaded?
                beq         at.done                     ; yes, so branch.
                cmp         #at.name.len                ; is the file name the right length?
                bne         nxentr                      ; no, so branch and check next entry.
                tay                                     ; could be, so now make sure it is.
tst.atalk       equ         *                           ; do the file names match?
                lda         (idxl),y                    ;
                ora         #$80                        ; set hi bit for comparison...
                cmp         at.name-1,y                 ; is it a match?
                bne         nxentr                      ; no, so check next entry.
                dey                                     ;
                bne         tst.atalk                   ; go chech next char for match.
                jsr         load.atalk                  ; go load the appletalk driver file.
                jsr         setup.atalk                 ; move and setup the atalk drivers.
                lda         #<gointerp                  ; put the load address for the interp
back
                sta         rdparm+3                    ; in the read parms. (low byte assumed
0)
                dec         atflg                       ; signify atalk now loaded.
                jmp         lodintrp                    ; now proceed with loading interp.
at.done         equ         *
                fin
*******************************************************
                cmp         #8                          ;must be at least 'x.system'
                bcc         nxentr                      ;otherwise, ignore it.
                bcs         skip2                       ;branch always...
noint1          beq         nointrp                     ;(branch extender)
*
skip2           tay                                     ;compare last 7 characters for
'.system'.
                ldx         #$6
lkintrp         lda         (idxl),y
                eor         iterp,x
                asl         a
                bne         nxentr                      ;branch if something else.
                dey
                dex
                bpl         lkintrp
                skp         1
load.atalk      equ         *-licode+lodintrp           ; (*********** en3 ************)
                ldy         #0                          ;move name to pathname buffer.
mvintrp         iny
                lda         (idxl),y
                sta         name,y
                ora         #$80                        ;make it printable in case of error.
                sta         iomess+$11,y
                cpy         name                        ;all characters moved?
                bne         mvintrp
                lda         #$a0                        ;save a space after name.
                sta         iomess+$12,y
                tya                                     ;update error message length.
                adc         #$13                        ;(carry was set)
                sta         ierlen
                skp         1
                jsr         gopro                       ;open interpreter file.
                dfb         iopen
                dw          opparm
                bne         badlod
```

```
                jsr         gopro                   ;get file's length
                dfb         igeof
                dw          efparm
                bne         badlod
                lda         eof+2                   ;make sure file will fit.
                bne         toolong
                lda         eof+1
                cmp         #$98                    ;max size is 38k
                bcs         toolong
                sta         rdlen+1
                lda         eof
                sta         rdlen                   ;read entire file.
                jsr         gopro
                dfb         iread
                dw          rdparm
                beq         goclos                  ;branch if sucessful read.
                cmp         #errbuf                 ;memory conflict?
                beq         toolong
                bne         badlod                  ;report i/o error.
goclos          jsr         gopro
                dfb         iclos
                dw          clparm
                bne         badlod                  ;(branch never, we hope).
*
**************** see rev note #en3 ****************
*
                ifeq        os-ednet
                lda         atflg                   ; is atalk being loaded?
                beq         goint                   ; no, so branch.
                rts                                 ; yes, so return and load the interp!
goint           equ         *
                fin
************************************************
* if we are booting on a //c and an escape is in the keyboard buffer
* then clear it so we dont interfere with start application
* (pizza accelerator chip requires ESC to shift speed down)
                lda         cflag-licode+lodintrp   ;booting on a 2c?
                beq         going                   ; branch if not
                lda         $c000                   ; fetch last key in board (pending or
not)
                cmp         #$9b                    ; ESCAPE character? (with bit 7 on)
                bne         going                   ; branch if not
                sta         $c010                   ; clear keyboard strobe
going           equ         *
                lda         romin
                jmp         gointerp
cflag           dfb         $00                     ; a one if an apple 2 c
nointrp         equ         *
*
*********** see rev note #en3 ********
*
                ifeq        os-ednet
                lda         atflg                   ; is atalk being loaded?
                beq         nitp                    ; no, so it is the interp not found.
                jmp         no.atalk                ; yes, and we can't find it!
nitp            equ         *
                fin
************************************
                ldx         #$27                    ;report no interpreter.
nitrp           lda         nomess,x
                sta         iscrn,x
                dex
                bpl         nitrp
```

```
                bmi         hang10
*
badlod          ldy         ierlen                  ;center the bad news.
                lda         #$27
                sec
                sbc         ierlen
                lsr         a
                adc         ierlen
                tax
bdlod1          lda         iomess,y
                sta         iscrn,x
                dex
                dey
                bpl         bdlod1
                bmi         hang10
*
toolong         ldy         #$1e
tlong1          lda         lgmess,y
                sta         iscrn+5,y
                dex
                bpl         tlong1
hang10          bmi         hang10
*
nomess          equ         *-licode+lodintrp
                asc         '**  UNABLE TO FIND A ".SYSTEM" FILE  ** ' '
lgmess          equ         *-licode+lodintrp
                asc         '**  SYSTEM PROGRAM TOO LARGE  **'
iomess          equ         *-licode+lodintrp
                asc         '** UNABLE TO LOAD X.SYSTEM *********' '
ierlen          equ         *-licode+lodintrp
                dfb         0
                skp         2
opparm          equ         *-licode+lodintrp
                dfb         $03
                dw          name
                dw          $1400
                dfb         $01
                skp         1
efparm          equ         *-licode+lodintrp
                dfb         $02
                dfb         01
eof             equ         efparm+2
                dfb         0,0,0
                skp         1
rdparm          equ         *-licode+lodintrp
                dfb         $04
                dfb         $01
                ifeq        os-ednet
                dw          at.load.addr            ; changed back to $2000 at
runtime.(*en3*)
                fin
                ifeq        os-prodos
                dw          $2000
                fin
rdlen           equ         rdparm+4
                dw          $0000
                dw          $0000
                skp         1
clparm          equ         *-licode+lodintrp
                dfb         01
                dfb         00
                skp         1
                skp         1
```

```
iterp           equ         *-licode+lodintrp
                asc         '.SYSTEM'
*
* note: atflg must be here so it also will get relocated!
*
                ifeq        os-ednet
atflg           equ         *-licode+lodintrp
                db          1                       ; <>0 --> loading atalk drivers
                fin
*
pclen           equ         *-licode
*
***************** see rev note #en3 *****************
*
                ifeq        os-ednet
setup.atalk     equ         *
*
                sta         $c009                   ; switch in aux lc (bank 1 in already).
                lda         eof+1                   ; check if necessary to overlap to bank
2.
                cmp         #$3f                    ; allow only $3fe4 bytes in alc, b1.
                beq         check.low               ; go see if low byte exceeds limit.
                bcs         needs.b2                ; branch if bank 2 is needed.
                bcc         one.chunk               ; branch if it fits in one chunk.
check.low       lda         eof                     ;
                cmp         #$e5                    ; does the low byte exceed the limit?
                bcs         needs.b2                ; branch if it does.
one.chunk       lda         eof                     ; move low # of bytes to relocate.
                sta         at.rel.tbl+3            ; store in relocate table.
                lda         eof+1                   ;
                sta         at.rel.tbl+4            ;
                jsr         reloc.at                ; move up the drivers.
                jmp         copy.at.tbl             ;
needs.b2        lda         #$e4                    ; relocate first $3fe4 bytes.
                sta         at.rel.tbl+3            ;
                lda         #$3f                    ;
                sta         at.rel.tbl+4            ;
                jsr         reloc.at                ; move 'um out..
                lda         eof                     ; now calculate overlap going into bank
2.
                sec                                 ;
                sbc         #$e4                    ; subtract low bytes..
                sta         at.rel.tbl+3            ; and save low byte overlap.
                lda         eof+1                   ; subtract hi bytes..
                sbc         #$3f                    ;
                sta         at.rel.tbl+4            ; and save overlap.
                sta         ramin2                  ; switch in bank 2.
                jsr         reloc.at                ; go move overlap.
*
*
* copy the atalk entry point table up into the main lc.
*
copy.at.tbl     equ         *
                sta         $c008                   ; switch back to main lc.
                jsr         lc1in                   ; and make sure bank 1 is in.
                ldx         #32                     ; 32 bytes in address table.
copytbl         lda         at.load.addr-1,x        ;
                sta         atalktbl-1,x            ; file, to the beginning of the mli
(file
                dex                                 ; xdosmli).
                bne         copytbl                 ;
*
* now do an open atp driver call to initialize the driver and
```

```
* bring up the appletalk interface.
*
                sei                                     ; no interrupts during driver init.
                jsr         gopro                       ;
                db          $42                         ; appletalk command number.
                dw          at.open.list                ; open driver parameter list pointer.
                bcc         open.ok                     ; branch if open was ok.
                jmp         at.open.err                 ; otherwise system error death time...
open.ok         cli                                     ; allow processor to get interrupted
again
*
* note that the appletalk interrupt handler is internal to prodos
* and is resident in the file xdosmli.
*
* now the atalk printer hook must be set up to get to the rpm stub.
*
                lda         at.slot                     ; $cn where n = atalk card slot number.
                tax                                     ; screenhole offset is also $cn...
                sta         $3b8,x                      ; atalks cards 1st screen hole = $cn.
                lda         #>rpm.entry-1               ; get low addr byte of rpm.stub
routine.
                sta         $6b8,x                      ; store in atalk card's 7th screen
hole.
                lda         #<rpm.entry-1               ; get hi addr byte of rpm.stub routine.
                sta         $738,x                      ; store in atalk card's 8th screen
hole.
                rts                                     ; set up for atalk complete!
*
reloc.at        equ         *
                ldx         atrelptr                    ; now go and relocate the
                ldy         atrelptr+1                  ; appletalk drivers...
                jsr         reloc                       ;
                rts                                     ; and return.
*
at.open.err     equ         *
                sta         romin                       ; rom in.
                jsr         home                        ; clear video.
                ldy         #atmsglen-1
badnews         lda         atmsg,y                     ; print "unable to open appletalk
driver"
                sta         qscrn,y                     ;
                dey                                     ;
                bpl         badnews
freeze          bmi         freeze                      ; and hang.....
*
no.atalk        equ         *
                sta         romin                       ; rom in.
                jsr         home                        ; clear video.
                ldy         #no.at.msg.len-1
badnews1        lda         noatalkmsg,y                ; print "unable to find appletalk
driver"
                sta         qscrn,y                     ;
                dey                                     ;
                bpl         badnews1
freeze1         bmi         freeze1                     ; and hang.....
*
atmsg           equ         *
                asc         '*** UNABLE TO OPEN APPLETALK DRIVER ***'
atmsglen        equ         *-atmsg
*
noatalkmsg      equ         *
                asc         '*** UNABLE TO FIND APPLETALK DRIVER ***'
no.at.msg.len   equ         *-noatalkmsg
```

```
*
at.open.list    db          1                       ; token length of prodos parameter
list.
                db          1                       ; atalk open driver command number.
at.slot         db          0                       ; 0 has driver return $cn location of
card here.
                db          1                       ; driver will return node number here.
at.int.handler  dw          $0000                   ; address of drivers's interrupt
routine.
                dw          $0000                   ; address of direct driver entry (using
x,y etc.).
*
at.name         equ         *
                asc         'ATALK.DRIVERS'
at.name.len     equ         *-at.name
*
                fin
*
************************************************
                skp         2
*
pg3stuf         equ         *                       ;this stuff goes on page 3
* ------------------- see rev note 15 -------------------------
*
* locate between vectors in page 3 starting at $3d6
*
*  note: since this is treated as a subroutine from the mli,
*        nothing may use the stack in main ram area!!
*
*  x = 5 from calling routine to move parameter bytes in the call
lp1             equ         *                       ;address after relocation = $3d6
                sta         $c008                   ;swap in main z.p. & stack
                lda         $42,x                   ;get the parameter bytes
                sta         $c009                   ;switch in aux stack & z.p.
                sta         $42,x                   ;save them in aux zero page
                dex
                bpl         lp1
*
* at this time, a jsr is no good since we have the wrong stack pointer
*  the first, and last thing a ram based driver must do is take care
*  of the stack pointers saved in $100 & $101 of alt stack.
*
* this operational address (after relocation) is $3e3.  it must be
*   patched to a jmp $d100 when the ram based driver is installed.
*
                lda         #$28                    ;no device connected error
                sec                                 ;show an error
*
comebak         equ         *                       ;ram drivers in aux lc must jmp back to
here
                sta         $c008
                rts                                 ;swap main z.p. & stack, and return
                dw          ramdrvr                 ;vector addr for patching into global
page
                ds          4,0                     ;spare bytes to align following vectors
to $3f0
* ------------------------------------------------------------
                dw          m.break                 ;brk vector
                dw          m.reset                 ;reset vector
                dfb         $5a                     ;power-up byte
                jmp         m.and                   ;'&' vector
                jmp         m.ctl.y                 ;ctrl-y vector
                jmp         m.nmi                   ;nmi vector
```

```
                dw          irqent                  ;interrupt vector to global page
                skp         2
lc1in           equ         *                       ;swap lc bank1 in
                lda         ramin
                lda         ramin
                rts
                page
whchrom         lda         #0                      ;assume standard apple ii first.
                sta         apple
                ldx         $fbb3                   ;look at the approved location...
                cpx         #$38                    ;apple ii? (actually is it autostart
rom?)
                beq         muchram                 ;branch if it is..
                lda         #$80                    ;else try for apple iie.
                cpx         #6
                beq         muchram
                lda         #$40                    ;if that fails, try ii+.
                cpx         #$ea                    ;must be one of these values...
                bne         whatsit
                ldx         $fb1e                   ;if it passes as ii+, then
                cpx         #$ad                    ; it might be /// in emulation.
                beq         muchram
                lda         #$d0                    ;mark it as 48k /// emulation!
                cpx         #$8a                    ; if it passes the test.
                bne         whatsit                 ;branch always, well, maybe.
nsm             sec                                 ;48k not allowed so apple ///
                rts                                 ;  emulation is not sufficient memory
whatsit         lda         #$02                    ;machine unknown if we land here.
                sta         (dst),y
                bne         finram                  ;branch always
                skp         1
muchram         sta         apple                   ;save rom id.
                jsr         lc1in                   ;test for the presents of 'language'
                lda         #$aa                    ; card ram.
                sta         $d000
                eor         $d000                   ;if it is there, result is zero.
                bne         nsm                     ;branch if it is not.
                lsr         $d000                   ;else check twice just to be sure.
                lda         #$55
                eor         $d000
                bne         nsm
                lda         #$20                    ;indicate at lc ram available.
is64k           ora         apple
finram          jmp         tst128                  ;go test for 128k.
                skp         1
tst128          equ         $80                     ;use zpage for this routine
strt128         sta         apple                   ;save accumulated value.
                bpl         not128                  ;branch if sure it's less than 128k.
                skp         1
                lda         #$ee                    ;first try storing in aux mem
                sta         $c005                   ;write to aux while on main zp
                sta         $c003                   ;set to read aux ram
                sta         $c00                    ;check for sparse mem mapping
                sta         $800
                lda         $c00                    ;see if sparse memory -same value
                cmp         #$ee                    ;1k away
                bne         noaux
                asl         $c00                    ;may be sparse mem so change value
                asl         a                       ;& see what happens
                cmp         $c00
                bne         noaux
                cmp         $800
                bne         auxmem
```

```
noaux           sec                             ;sparse mapping so no aux mem
                bcs         back
auxmem          clc                             ;there is aux mem
back            sta         $c004               ;switch  back to write main ram
                sta         $c002               ;switch back main ram read
                bcs         not128              ;branch if not 128k
                lda         apple               ;else update identity of machine.
                ora         #$30                ;indicate 128k present.
                sta         apple
                skp         1
not128          lda         look+1              ;futs with pointer for apple test.
                sec
                sbc         #$05                ;should result in $fb if zpage is ok.
                sta         look+1
                bcs         *+4                 ;(to the clc)
                dec         look
                clc
                rts
end128          equ         *-strt128           ;byte count for routine move to zpage.

; ###############################################################################################
; #   END OF FILE:  PROLDR
; #   LINES      :  1095
; #   CHARACTERS :  56440
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.READ.WRITE.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  READ.WRITE
; ###############################################################################################

                page
read            jsr         mvdbufr                 ;first transfer  buffer adr & request
count
                jsr         mvcbytes                ; to a more accessable location, also
get fcbattr, clc.
                pha                                 ;save attributes for now.
                jsr         calcmrk                 ;calc mark after read, test mark>eof
                pla                                 ;carry set indicates end mark>eof
                and         #readen                 ;test for read enabled first.
                bne         read1                   ;branch if ok to read
                lda         #accserr                ;report illegal access.
                bne         gofix1                  ;branch always taken.
                skp         1
read1           bcc         read2                   ;branch if result mark<eof.
;adjust request to read up to (but not including) end of file.
                ldy         fcbptr
                lda         fcb+fcbeof,y            ;result= (eof-1)-position
                sbc         tposll
                sta         cbytes
                sta         rwreql
                lda         fcb+fcbeof+1,y
                sbc         tposlh
                sta         cbytes+1
                sta         rwreqh
                ora         cbytes                  ;if both bytes are zero, report eof
error.
                bne         read3
                lda         #eoferr
gofix1          jmp         errfix1
                skp         1
read2           lda         cbytes
                ora         cbytes+1
                bne         read3                   ;branch if read request definitely non-
zero.
gorddne         jmp         rwdone                  ;do nothing.
                skp         1
read3           jsr         valdbuf                 ;validate user's data buffer range.
                bcs         gofix1                  ;branch if memory conflict.
                jsr         gfcbstyp                ;get storage type.
                cmp         #tretyp+1               ;now find if it's a tree or other.
                bcc         tread                   ;branch if a tree file.
                jmp         dread                   ;othewise assume it's a directory.
                skp         1
tread           jsr         rdposn                  ;get data pointer set up.
                bcs         gofix1                  ;report any errors
                jsr         preprw                  ;test for newline, sets up for partial
read.
                jsr         readpart                ;move current data buffer contents to
user area
                bvs         gorddne                 ;branch if request is satisfied.
                bcs         tread                   ;carry set indicates newline is set.
                lda         rwreqh                  ;find out how many blocks are to be
read
```

```
                lsr         a                       ;if less than two, then do it the slow
way.
                beq         tread
                sta         bulkcnt                 ;save bulk block count.
                jsr         gfcbstat                ;make sure current data area doesn't
need writing before
                and         #datmod                 ; resetting pointer to read directly
into
                bne         tread                   ; user's area. branch if data need to
be written
                sta         ioaccess                ; to force first call thru all device
handler checking.
                lda         usrbuf                  ;make the data buffer the user's space.
                sta         datptr
                lda         usrbuf+1
                sta         datptr+1
                page
rdfast          jsr         rdposn                  ;get next block directly into user
space.
                bcs         errfix                  ;branch on any error.
rdfast0         inc         datptr+1                ;bump all pointers by 512 (one block)
                inc         datptr+1
                dec         rwreqh
                dec         rwreqh
                inc         tposlh
                inc         tposlh
                bne         rdfast1                 ;branch if position does not get to a
64k boundary.
                inc         tposhi                  ;otherwise, must chect for a 128k
boundary
                lda         tposhi                  ; set carry if mod 128k has been
reached
                eor         #1
                lsr         a
rdfast1         dec         bulkcnt                 ;have we read all we can fast?
                bne         rdfast2                 ;branch if more to read.
                jsr         fxdatptr                ;go fix up data pointer to xdos buffer.
                lda         rwreql                  ;test for end of read.
                ora         rwreqh                  ;are both zero?
                beq         rwdone
                bne         tread                   ;no, read last partial block.
*
rdfast2         bcs         rdfast
                lda         tposhi                  ;get index to next block address
                lsr         a
                lda         tposlh
                ror         a
                tay                                 ;index to address is int(pos/512)
                lda         (tindx),y               ;get low address
                sta         bloknml
                inc         tindx+1
                cmp         (tindx),y               ;are both hi and low address the same?
                bne         realrd                  ;no, it's a real block address.
                cmp         #0                      ;are both bytes zero?
                bne         realrd                  ; nope -- must be real data
                sta         ioaccess                ; don't do repeatio just after sparse
                beq         nostuf                  ; branch always (carry set)
realrd          lda         (tindx),y               ;get high address byte
                clc
nostuf          dec         tindx+1
                bcs         rdfast                  ;branch if no block to read
                sta         bloknmh
                lda         ioaccess                ;has first call gone to device yet?
```

```
                beq         rdfast                  ;nope, go thru normal route...
                clc
                php                                 ;interupts cannot occur while calling
dmgr
                sei
                lda         datptr+1                ;reset hi buffer address for device
handler
                sta         dbufph
                jsr         dmgr
                bcs         errfx00                 ;branch if error
                plp
                bcc         rdfast0                 ;no errors, branch always.
                page
errfx00         plp                                 ;restore interupts.
errfix          pha                                 ;save error code
                jsr         fxdatptr                ;go restore data pointers, etc...
                pla
errfix1         pha                                 ;save error code
                jsr         rwdone                  ;pass back number of bytes actually
read.
                pla
                sec                                 ;report error
                rts
*
                skp         1
rwdone          ldy         #c.trnscnt              ;return total number of bytes actually
read
                sec                                 ; this is derived from cbytes-rwreq
                lda         cbytes
                sbc         rwreql
                sta         (par),y
                iny
                lda         cbytes+1
                sbc         rwreqh
                sta         (par),y
                jmp         rdposn                  ;leave with valid position in fcb.
*
preprw          ldy         fcbptr                  ;adjust pointer to user's buffer to
                sec                                 ;make the transfer
                lda         usrbuf
                sbc         tposll
                sta         usrbuf
                bcs         preprw1                 ;branch if no adjustment to hi addr.
needed.
                dec         usrbuf+1
preprw1         lda         fcb+fcbnlmsk,y          ;test for new line enabled.
                clc
                beq         nonewlin                ;branch if newline is not enabled
                sec                                 ;carry set indicates newline enabled.
                sta         nlmask
                lda         fcb+fcbnewl,y           ;move newline character to more
                sta         nlchar                  ; accessable spot.
nonewlin        ldy         tposll                  ;get index to first data
                lda         datptr                  ;reset low order of posptr to beginning
of page.
                sta         posptr
                ldx         rwreql                  ;and lastly get low order count of
requested bytes.
                rts                                 ;return statuses...
*
readpart        txa                                 ;(x contains low count of bytes to
move)
                bne         rdpart0                 ;branch if request is not a even pages
```

```
                lda         rwreqh                  ;a call of zero bytes should never get
here!
                beq         setrdne                 ;branch if nothin' to do.
                dec         rwreqh
rdpart0         dex
rdpart          lda         (posptr),y              ;move data to user's buffer
                sta         (usrbuf),y              ; one byte at a time.
*
*************** see rev note #48 *************
*
* the following 4 commented out lines are the original code...
* txa ;note: this routine is coded to be
* beq endrqchk ; fastest when newline is disabled.
*rdpart1 bcs tstnewl ;branch if new line needs to be tested.
*rdpart2 dex
                bcs         tstnewl                 ; let's test for newline first!
rdpart2         txa                                 ; note: x must be unchanged from
tstnewl!
                beq         endrqchk                ; see if read request is satisfied...
rdpart1         equ         *                       ; continue
*********************************************
                dex                                 ; dec # of bytes left to move.
                iny                                 ;page crossed?
                bne         rdpart                  ;no. move next byte.
                lda         posptr+1                ;test for end of buffer
                inc         usrbuf+1                ; but first adjust user buffer pointer
                inc         tposlh                  ; and position.
                bne         rdpart3
                inc         tposhi
rdpart3         inc         posptr+1                ;and sos buffer high address.
                eor         datptr+1                ;(carry has been cleverly undisturbed.)
                beq         rdpart                  ;branch if more to read in buffer.
                clv                                 ;indicate not finished.
                bvc         rdprtdne                ;branch always.
*
endrqchk        lda         rwreqh
                beq         rdrqdne                 ;branch if reqest satisfied.
                iny                                 ;done with this block of data?
                bne         endrchk1                ;no, adjust high byte of request.
                lda         posptr+1                ;maybe- check for end of block buffer.
                eor         datptr+1                ;(don't disturb carry)
                bne         endrchk2                ;branch if hi count can be dealt with
next time.
endrchk1        dec         rwreqh
endrchk2        dey                                 ;restore proper value to 'y'
                jmp         rdpart1
*
tstnewl         lda         (posptr),y              ;get last byte transfered again.
                and         nlmask                  ;only bits on in mask are significant.
                eor         nlchar                  ;have we matched newline character?
                bne         rdpart2                 ;no, read next.
rdrqdne         iny                                 ;adjust position.
                bne         setrdne
                inc         usrbuf+1                ;bump pointers.
                inc         tposlh
                bne         setrdne
                inc         tposhi
setrdne         bit         setvflg                 ;(set v flag)
rdprtdne        sty         tposll                  ;save low position
                bvs         rdone1
                inx                                 ;leave request as +1 for next call
rdone1          stx         rwreql                  ;and remainder of request count.
                php                                 ;save statuses
```

```
                clc                                 ;adjust user's low buffer address
                tya
                adc         usrbuf
                sta         usrbuf
                bcc         rdpart4
                inc         usrbuf+1                ;adjust hi address as needed.
rdpart4         plp                                 ;restore return statuses
setvflg         rts                                 ;(this byte <$60> is used to set v
flag)
*
fxdatptr        lda         datptr                  ;put current user buffer
                sta         usrbuf                  ; address back to normal
                lda         datptr+1
                sta         usrbuf+1                ;bank pair byte should be moved also.
                ldy         fcbptr                  ;restore buffer address
                jmp         fndfcbuf
*
                page
*
* read directory file...
*
dread           jsr         rdposn
                bcs         errdrd                  ;pass back any errors
                jsr         preprw                  ;prepare for transfer.
                jsr         readpart                ;move data to user's buffer
                bvc         dread                   ;repeat until request is satisfied.
                jsr         rwdone                  ;update fcb as to new position.
                bcc         dredone                 ;branch if all is well.
                cmp         #eoferr                 ;was last read to end of file?
                sec                                 ;anticipate some other problem
                bne         drederr                 ;branch if not eof error.
                jsr         svmark
                jsr         zipdata                 ;clear out data block.
                ldy         #0                      ;provide dummy back pointer for future
re-position
                ldx         fcbptr                  ;get hi byte of last block.
dred1           lda         fcb+fcbdatb,x
                sta         (datptr),y
                lda         #0                      ;mark current block as imposible.
                sta         fcb+fcbdatb,x
                inx
                iny                                 ;bump indexes to do both hi & low
bytes.
                cpy         #2
                bne         dred1
dredone         clc                                 ;indicate no error
drederr         rts
*
errdrd          jmp         errfix1                 ;report how much we could transfer
before error.
*
mvcbytes        ldy         #c.reqcnt               ;move request count
                lda         (par),y                 ;to a more accessable location.
                sta         cbytes
                sta         rwreql
                iny
                lda         (par),y
                sta         cbytes+1
                sta         rwreqh
                ldy         fcbptr                  ;also return y=val(fcbptr),
                lda         fcb+fcbattr,y           ; and a=attributes
                clc                                 ;and carry clear...
                rts
```

```
*
mvdbufr         ldy         #c.datbuf               ;move pointer to users buffer to bfm
                lda         (par),y
                sta         usrbuf                  ; z-page area.
                iny
                lda         (par),y
                sta         usrbuf+1
gfcbstyp        ldy         fcbptr                  ;also return storage type.
                lda         fcb+fcbstyp,y
                rts
*
calcmrk         ldx         #0                      ;this subroutine adds the requested
byte
                ldy         fcbptr
                clc
calcmrk1        lda         fcb+fcbmark,y           ; count to mark, and returns sum
                sta         tposll,x                ; in scrtch and also returns mark in
tpos.
                sta         oldmark,x               ; and oldmark.
                adc         cbytes,x
                sta         scrtch,x                ;on exit: y, x, a=unknown
                txa                                 ; carry set indicates scrtch>eof.
                eor         #2                      ;(cbytes+2 always = 0)
                beq         eoftest
                iny
                inx
                bne         calcmrk1                ;branch always
eoftest         lda         scrtch,x                ; new mark in scrtch!
                cmp         fcb+fcbeof,y            ;is new position > eof?
                bcc         eoftst1                 ;no, proceed.
                bne         eoftst1                 ;yes, adjust 'cbytes' request
                dey
                dex                                 ;have we compared all tree bytes?
                bpl         eoftest                 ;no, test next lowest.
eoftst1         rts
                skp         2
werreof         jsr         plus2fcb                ;reset eof to pre-error position.
weof1           lda         oldeof,x                ;place oldeof back into fcb.
                sta         fcb+fcbeof,y
                lda         oldmark,x               ;also reset mark to last best write
position.
                sta         fcb+fcbmark,y
                sta         scrtch,x                ;and copy mark to scrtch for test of
                dey                                 ; eof less than mark.
                dex
                bpl         weof1
                jsr         plus2fcb                ;get pointers to test eof<mark
                jsr         eoftest                 ;carry set means mark>eof!!
* drop into wadjeof to adjust eof to mark if necessary.
                skp         1
wadjeof         jsr         plus2fcb                ;get y=fcbptr+2, x=2, a=y
wadj1           lda         fcb+fcbeof,y            ;copy eof to oldeof
                sta         oldeof,x
                bcc         wadj2                   ;and if carry set...
                lda         scrtch,x                ;copy scrtch to fcb's eof
                sta         fcb+fcbeof,y
wadj2           dey
                dex                                 ;copy all three bytes.
                bpl         wadj1
                rts
                skp         1
plus2fcb        lda         #2                      ;on exit a&y=(fcbptr)+2
                tax                                 ;x=2
```

```
                ora         fcbptr
                tay
                rts
                page
write           equ         *                       ;first determine if reqested
                jsr         mvcbytes                ; write is legal
                pha
                jsr         calcmrk                 ;save a copy of eof to oldeof, set/clr
carry
                jsr         wadjeof                 ; to determine if new mark > eof.
                pla                                 ;get attributes again.
*
                and         #writen                 ;is write enabled?
                bne         write1                  ;yes, continue...
erraccs         lda         #accserr                ;report illegal access.
                bne         writerror               ;branch always taken.
*
write1          jsr         tstwprot                ;otherwise, make sure device is not
write protected.
                bcs         writerror               ;report write potected and abort
operation.
                lda         cbytes
                ora         cbytes+1                ;anything to write?
                bne         write2                  ;branch if write request definitely
non-zero.
                jmp         rwdone                  ;do nothing.
*
                page
write2          jsr         mvdbufr                 ;move pointer to users buffer to bfm
                cmp         #tretyp+1               ;zpage area, also get storage type.
                bcs         erraccs                 ;if not tree, return an access error!
twrite          jsr         rdposn                  ;read block we're
                bcs         writerror
                jsr         gfcbstat
*
                and         #datalc+idxalc+topalc
                beq         trewrt1
                ldy         #0                      ;find out if enough disk space is
available for
twrtalc         iny                                 ; indexes and data block
                lsr         a
                bne         twrtalc
                sty         reql
                sta         reqh
                jsr         tsfrblk
                bcs         writerror               ;pass back any errors.
                jsr         gfcbstat                ;now get more specific.
                and         #topalc                 ;are we lacking a tree top?
                beq         tstsapwr                ;no, test for lack of sapling level
index.
                jsr         topdown                 ;go allocate tree top and adjust file
type.
                bcc         dblokalc                ;continue with allocation of data
block.
writerror       pha                                 ; save error
                jsr         werreof                 ;adjust eof and mark to pre-error state
                pla                                 ;restore error code.
                jmp         errfix1                 ; error return
                page
tstsapwr        jsr         gfcbstat                ;get status byte again.
                and         #idxalc                 ;do we need a sapling level index
block?
```

```
                beq         dblokalc                ;no, assume it's just a data block
needed.
                jsr         sapdown                 ;go allocate an index block and update
tree top.
                bcs         writerror               ;return any errors.
dblokalc        jsr         alcwblk                 ;go allocate for data block.
                bcs         writerror
                jsr         gfcbstat                ;clear allocation required bits in
status.
                ora         #idxmod                 ;but first tell 'em index block is
dirty
                and         #$ff-datalc-idxalc-topalc
                sta         fcb+fcbstat,y
                lda         tposhi                  ;calculate position within index block.
                lsr         a
                lda         tposlh
                ror         a
                tay                                 ;now put block address into index block
                inc         tindx+1                 ; high byte first.
                lda         scrtch+1
                tax
                sta         (tindx),y
                dec         tindx+1                 ;(restore pointer to lower page of
index block)
                lda         scrtch                  ;get low block address
                sta         (tindx),y               ;now store low address.
                ldy         fcbptr                  ;also update file control block to
indicate
                sta         fcb+fcbdatb,y           ; that this block is allocated.
                txa                                 ;get high address again.
                sta         fcb+fcbdatb+1,y
trewrt1         jsr         preprw                  ; write on
                jsr         wrtpart
                bvc         twrite
                jmp         rwdone                  ;update fcb with new position.
*
                page
wrtpart         txa
                bne         wrpart                  ;branch if request is not a even pages
                lda         rwreqh                  ;a call of zero bytes should never get
here!
                beq         setwrdne                ;do nothing!
*
                dec         rwreqh
wrpart          dex
                lda         (usrbuf),y              ;move data from user's buffer
                sta         (posptr),y              ; one byte at a time.
                txa
                beq         endwqchk
wrpart2         iny                                 ;page crossed?
                bne         wrpart                  ;no. move next byte.
                lda         posptr+1                ;test for end of buffer
                inc         usrbuf+1                ; but first adjust user buffer pointer
                inc         tposlh                  ; and position.
                bne         wrpart3
                inc         tposhi
; don't wrap around on file!
                bne         wrpart3
                lda         #posnerr                ; say out of range if >32 meg
                bne         writerror
wrpart3         inc         posptr+1                ;and sos buffer high address.
                eor         datptr+1                ;(carry has been cleverly undisturbed.)
                beq         wrpart                  ;branch if more to write to buffer.
```

```
                clv                                     ;indicate not finished.
                bvc         wrprtdne                    ;branch always.
*
endwqchk        lda         rwreqh
                beq         wrtrqdne                    ;branch if reqest satisfied.
                iny                                     ;are we done with this block of data?
                bne         endwchk1                    ;branch if not.
                lda         posptr+1
                eor         datptr+1                    ;while this is redundant, it's
necessary for
                bne         endwchk2                    ; proper adjustment of request count.
endwchk1        dec         rwreqh                      ;(not finished- ok to adjust hi byte.)
endwchk2        dey                                     ;reset modified y
                jmp         wrpart2
*
wrtrqdne        iny                                     ; and position.
                bne         setwrdne
                inc         usrbuf+1                    ;bump pointers.
                inc         tposlh
                bne         setwrdne
                inc         tposhi
setwrdne        bit         setvflg                     ;(set v flag)
wrprtdne        sty         tposll                      ;save low position
                stx         rwreql                      ;and remainder of request count.
                php                                     ;save statuses
                jsr         gfcbstat
                ora         #datmod+usemod
                sta         fcb+fcbstat,y
                clc                                     ;adjust user's low buffer address
                lda         tposll
                adc         usrbuf
                sta         usrbuf
                bcc         wrpart4
                inc         usrbuf+1                    ;adjust hi address as needed.
wrpart4         jsr         fcbused                     ; set directory flush bit
                plp                                     ;restore return statuses
                rts
                page
topdown         jsr         swapdown                    ;first make current 1st block an entry
in new top.
                bcs         tpdwnerr                    ;return any errors
                jsr         gfcbstyp                    ;find out if storage type has been
changed to 'tree'.
;(if not, assume it was originally a seed and
                cmp         #tretyp                     ; both levels need to be built.
                beq         topdwn1                     ; otherwise, only an index need be
allocated)
                jsr         swapdown                    ;make previous swap a sap level index
block.
                bcs         tpdwnerr
topdwn1         jsr         alcwblk                     ;get another block address for the sap
level index.
                bcs         tpdwnerr
                lda         tposhi                      ;calculate position of new index block
                lsr         a                           ; in the top of the tree.
                tay
                lda         scrtch                      ;get address of newly alocated index
block again
                tax
                sta         (tindx),y
                inc         tindx+1
                lda         scrtch+1
                sta         (tindx),y                   ;save hi address
```

```
                dec         tindx+1
                ldy         fcbptr                  ;make newly allocated block the current
index block.
                sta         fcb+fcbidxb+1,y
                txa
*
                sta         fcb+fcbidxb,y
                jsr         wfcbfst                 ;save new top of tree.
                bcs         tpdwnerr
*
***************** see rev note #40 *************************
*
                jmp         zeroindex               ; zero index block in user's i/o
buffer.
* ldy #0 ;end by re-clearing current (new) index block.
* tya
*zindx1 sta (tindx),y ;do first page.
* iny
* bne zindx1
* inc tindx+1 ;bump addr to do second page.
*zindx2 sta (tindx),y
* iny
* bne zindx2
* dec tindx+1 ;restore original addr.
* rts
************************************************************
*
sapdown         jsr         gfcbstyp                ;find out if we're dealing with a tree
                cmp         #seedtyp                ;if seed then an adjustment to file
type is necessary.
                beq         sapdwn1                 ;branch if seed.
                jsr         rfcbfst                 ;otherwise read in top of tree.
                bcc         topdwn1                 ;branch if no error.
tpdwnerr        rts                                 ;return errors
*
                page
sapdwn1         equ         *                       ;make current seed into a sapling
*
swapdown        jsr         alcwblk                 ;allocate a block before swap
                bcs         swaperr                 ;return errors immediately.
                ldy         fcbptr                  ;get previous first block
                lda         fcb+fcbfrst,y           ; address into index block.
                pha                                 ;save temporarly while swapping in new
top index
                lda         scrtch                  ;get new block address (low)
                tax
                sta         fcb+fcbfrst,y
                lda         fcb+fcbfrst+1,y
                pha
                lda         scrtch+1                ;and high address too.
                sta         fcb+fcbfrst+1,y
                sta         fcb+fcbidxb+1,y         ;make new top also the current index in
memory.
                txa                                 ;get low address again
                sta         fcb+fcbidxb,y
                ldy         #0                      ;make previous the first entry in sub
index
                inc         tindx+1
                pla
                sta         (tindx),y
                dec         tindx+1
                pla
                sta         (tindx),y
```

```
                jsr         wfcbfst                 ;save new file top.
                bcs         swaperr
                jsr         gfcbstyp                ;now adjust storage type
                adc         #1                      ; by adding 1 (thus seed becomes
sapling becomes tree)
                sta         fcb+fcbstyp,y
                lda         fcb+fcbstat,y           ;mark storage type modified.
                ora         #stpmod
                sta         fcb+fcbstat,y
                clc                                 ;return 'no error' status.
swaperr         rts
*
                page
alcwblk         jsr         alc1blk
                bcs         aluserr
                jsr         gfcbstat                ;mark usage as modified
                ora         #usemod
                sta         fcb+fcbstat,y
                lda         fcb+fcbuse,y            ;bump current usage count by 1.
                clc
                adc         #1
                sta         fcb+fcbuse,y
                lda         fcb+fcbuse+1,y
                adc         #0
                sta         fcb+fcbuse+1,y
wrok            clc                                 ;indicate no error
aluserr         rts                                 ;all done
*
tstwprot        jsr         gfcbstat                ;check for a 'never been modified'
condition
                and         #usemod+datmod+idxmod+eofmod
                bne         wrok                    ;ordinary rts if known write ok.
                lda         fcb+fcbdevn,y           ;get file's device number.
                sta         devnum                  ;get current status of block device
twrprot1        sta         unitnum                 ; make the device status call.
                lda         bloknmh
                pha
                lda         bloknml                 ;save the current block values
                pha
                lda         #statcmd
                sta         dhpcmd
                sta         bloknml                 ;zero the block #
                sta         bloknmh
                php
                sei
                jsr         dmgr
                bcs         twerr                   ; branch if write protect error
                lda         #$00                    ; otherwise, assume no errors
twerr           plp                                 ; restore interrupt status
                clc
                tax                                 ; save error
                beq         twno                    ; branch if no error
                sec                                 ; else, set carry to show error
twno            equ         *
                pla
                sta         bloknml                 ;restore the block #
                pla
                sta         bloknmh
                txa
wprotret        rts                                 ; carry is indeterminate
*

; ###############################################################################
```

```
; #   END OF FILE:  READ.WRITE
; #   LINES      :  615
; #   CHARACTERS :  35513
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.RELOC.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  RELOC
; ###############################################################################################

               page
****************************************************
* program/data relocation routine is driven by a table
* describing program, vectors, and data segments.  the
* table has the general format:
*  (1)command:     0= zero destinaton range.
*                  1= move data to from src to dst.
*                  2= high address ref tbl, relocate & move.
*                  3= lo-hi addr ref tbl, relocate & move.
*                  4= program, relocate & move.
*                 >4= end of table.
*  (2)dest addr:   address of where segment is to be moved.
*  (2)byte count:  length of segment.
*  (2)source addr: start address segment to be operated on,
*                  n/a if type=0, code does not have to be
*                  assembled at this address.
*  (1)segments:    number of address ranges to be tested
*                  and altered, n/a if type=0 or 1.
*                  limit and offset lists should each
*                  contain segments+1 (s) bytes.
*  (s)limitlow:    list of low page addresses to be tested.
*  (s)limithigh:   list of high page addresses to be tested.
*  (s)offset:      list of amounts to be added if
*                  low & high limits have been met.
*
* on entry: x=table address low, y=table address high
* on exit:  carry clear if no error; else carry set,
*           x=addrlo, y=addrhi of error causing source.
*           a=0 if input table error, =$ff if illegal opcode.
****************************************************
               skp          1
reloc          stx          rtbl                    ;save address of control table
               sty          rtbl+1
*
*************** see note #68 *****************
*
* patch for the gs rom... force the intcxrom bit off
*  to guarantee that the internal/external status of
*  the slots are set per the control panel.
*
* the patch is placed here not because it belongs here;
*  it is here because it must be executed before whchrom and
*  the code between $2000 and the end of greet must stay
*  where it is (to avoid altering the location of the splashscreen
*  bytes which are patched by prodos/16!).
*
statereg       equ          $c068
*
               lda          statereg
               and          #$fe
               sta          statereg                ;we have forced off intcxrom
*
***********************************************
```

```
*
                skp         1
rloop           ldy         #0                  ;get relocation command
                lda         (rtbl),y
                cmp         #5                  ;if 5 or greater then done...
                bcs         rlend               ;branch if done.
                tax
                iny                             ;move destination address to
                lda         (rtbl),y            ; zero page for indirect access.
                sta         dst
                iny
                lda         (rtbl),y
                sta         dst+1
                iny
                lda         (rtbl),y            ;also the length (byte count)
                sta         cnt                 ; of the destination area.
                iny
                lda         (rtbl),y
                sta         cnt+1
                bmi         rlerr               ;branch if >=32k
                txa                             ;request to zero out destination?
                beq         zero                ;branch if it is.
                page
                iny
                lda         (rtbl),y            ;now get the source address.
                sta         src
                sta         cde                 ;src is used for the move, cde is
                iny                             ; used for relocation.
                clc                             ;add length to get final address
                adc         cnt
                sta         ecde
                lda         (rtbl),y
                sta         src+1
                sta         cde+1
                adc         cnt+1
                sta         ecde+1
                dex                             ;now that set-up is done, test for
'move'
                beq         movem               ;branch if move only (no relocation).
                stx         wsize               ;save element size (1,2,3).
                iny
                lda         (rtbl),y            ;now get the number of ranges
                sta         sgcnt               ; that are valid relocation target
addresses.
                tax                             ;separate serial range groups into
tables.
rlimlo          iny
                lda         (rtbl),y            ;transfer low limits to 'limlo' table.
                sta         limlo,x
                dex
                bpl         rlimlo
                ldx         sgcnt
rlimhi          iny
                lda         (rtbl),y            ;transfer high limits to 'limhi' table.
                sta         limhi,x
                dex
                bpl         rlimhi
                ldx         sgcnt
rofset          iny
                lda         (rtbl),y            ;transfer offsets to 'ofset' table.
                sta         ofset,x
                dex
                bpl         rofset
```

```
                skp         1
                jsr         adjtbl                  ;adjust 'rtbl' to point at next spec.
                ldx         wsize                   ;test for machine code relocation.
                cpx         #3
                beq         rlcode                  ;branch if program relocation
                jsr         reladr                  ;otherwise, relocate addresses in
rlend1          jsr         move                    ; one or two byte tables, then move to
destination.
                jmp         rloop                   ;do next table entry...
                skp         1
rlend           clc
                rts
                skp         1
rlerr           jmp         tblerr
                skp         1
rlcode          jsr         rlprog                  ;go relocate machine code references.
                jmp         rlend1
                page
zero            jsr         adjtbl                  ;adjust 'rtbl' pointer to next entry.
                lda         #0                      ;fill destination range with zeros.
                ldy         cnt+1                   ;is it at least a page?
                beq         zpart                   branch if less than 256 bytes.
                tay
zero1           sta         (dst),y
                iny
                bne         zero1
                inc         dst+1                   ;bump to next page.
                dec         cnt+1
                bne         zero1                   ;branch if more pages to clear.
zpart           ldy         cnt                     ;any bytes left to zero?
                beq         zeroed                  ;branch if not.
                tay
zpart1          sta         (dst),y
                iny
                cpy         cnt
                bcc         zpart1
zeroed          jmp         rloop                   ;do next thing in table.
                skp         1
movem           jsr         adjtbl
                jmp         rlend1
                skp         2
adjtbl          tya                                 ;add previous table length to 'rtbl'
                sec                                 ; to get position of next entry in
table.
                adc         rtbl
                sta         rtbl
                bcc         adjtbl1
                inc         rtbl+1
adjtbl1         rts
                skp         2
move            lda         src+1                   ;determine if move is up, down,
                cmp         dst+1                   ; or not at all.
                bcc         movup                   ;branch if definitely up...
                bne         movdn                   ;branch if definitely down...
                lda         src
                cmp         dst
                bcc         movup                   ;branch if definitely up...
                bne         movdn                   ;branch if definitely down...
                rts                                 ;otherwise, don't move nutin.
                page
movup           ldy         cnt+1                   ;calc highest page of move up.
                tya
                clc
```

```
                adc         src+1
                sta         src+1                   ; and adjust src & dst accordingly.
                tya
                clc
                adc         dst+1
                sta         dst+1
                ldy         cnt                     ;move partial page first.
                beq         mvup2                   ;branch if no partial pages.
mvup1           dey
                lda         (src),y
                sta         (dst),y
                tya                                 ;end of page transfer?
                bne         mvup1
mvup2           dec         dst+1
                dec         src+1
                dec         cnt+1                   ;done with all pages?
                bpl         mvup1                   ;branch if not.
                rts
                skp         1
movdn           ldy         #0
                lda         cnt+1                   ;partial page move only?
                beq         mvdn2                   ;branch if less than a page to be
moved.
mvdn1           lda         (src),y
                sta         (dst),y
                iny
                bne         mvdn1
                inc         dst+1                   ;bump addresses
                inc         src+1
                dec         cnt+1                   ;more pages?
                bne         mvdn1                   ;branch if more pages.
mvdn2           lda         cnt                     ;move partial page.
                beq         mvdn4                   ;branch if no more to move.
mvdn3           lda         (src),y
                sta         (dst),y
                iny
                cpy         cnt
                bne         mvdn3
mvdn4           rts                                 ;all done...
                page
reladr          ldy         wsize                   ;determine 1 or 2 byte reference
                dey
                lda         (cde),y
                jsr         adjadr                  ;relocate reference.
                lda         wsize                   ;update and test 'cde' pointer
                jsr         adjcde
                bcc         reladr                  ;branch if more to do.
                rts
                skp         1
rlprog          ldy         #0                      ;fetch next opcode.
                lda         (cde),y
                jsr         oplen                   ;determine if it's a 3 byte
instruction.
                beq         rperr                   ;branch if not an opcode.
                cmp         #3
                bne         rlprg1
                ldy         #2
                jsr         adjadr                  ;relocate address
                lda         #3
rlprg1          jsr         adjcde                  ;update and test 'cde' for done.
                bcc         rlprog                  ;loop if more to do.
                rts
*
```

```
* error handling...
*
rperr           pla                             ;return bad code address
                pla                             ;first un-do stack
                ldx         cde
                ldy         cde+1
                lda         #$ff                ;indicate bad opcode.
                sec                             ;indicate error
                rts
*
tblerr          equ         *                   ;return table address error
                ldx         rtbl
                ldy         rtbl+1
                lda         #0                  ;indicate input table error
                sec
                rts
*
*
adjadr          lda         (cde),y             ;get page address
                ldx         sgcnt               ; and test against limits.
adjad1          cmp         limlo,x             ;is it >= low?
                bcc         adjad2              ;branch if not.
                cmp         limhi,x             ;is it =< highest page limit.
                bcc         adjad3              ;branch if it is.
                beq         adjad3
adjad2          dex                             ;try next limit set.
                bpl         adjad1
                rts                             ;return without adjustment.
adjad3          clc                             ;add offset to form relocated
                adc         ofset,x             ; page address.
                sta         (cde),y             ;and replace old address with result.
                rts
                skp         2
adjcde          clc                             ;update 'cde' pointer
                adc         cde
                ldy         cde+1
                bcc         adcde1              ;branch if not page cross.
                iny                             ;update high order address too.
adcde1          cpy         ecde+1              ;has all code/data been processed?
                bcc         adcde2              ;branch if definitely not.
                cmp         ecde                ;if carry results set, end of code.
adcde2          sta         cde
                sty         cde+1               ;save updated values.
                rts                             ;return result (carry set=done).
                page
oplen           pha                             ;form index to table and which 2-bit
group.
                and         #3                  ;low 2 bits specify group.
                tay
                pla
                lsr         a                   ;upper 6 bits specify byte in table.
                lsr         a
                tax
                lda         opcodln,x
nxgroup         dey                             ;is opcode length in lowest 2 bits of
acc?
                bmi         rtnlen              ;branch if it is.
                lsr         a
                lsr         a                   ;shift to next group
                bne         nxgroup             ;if len=0 then error...
rtnlen          and         #3                  ;strip other garbage
                rts                             ;if z-flg true, then error!!!
*
```

```
*
* the following table contains the length of each
* machine instruction (in two-bit groups).
*
opcodln         dfb          $09,$28,$19,$3c
                dfb          $0a,$28,$0d,$3c
                dfb          $0b,$2a,$19,$3f
                dfb          $0a,$28,$0d,$3c
                dfb          $09,$28,$19,$3f
                dfb          $0a,$28,$0d,$3c
                dfb          $09,$28,$19,$3f
                dfb          $0a,$28,$0d,$3c
                dfb          $08,$2a,$11,$3f
                dfb          $0a,$2a,$1d,$0c
                dfb          $2a,$2a,$19,$3f
                dfb          $0a,$2a,$1d,$3f
                dfb          $0a,$2a,$19,$3f
                dfb          $0a,$28,$0d,$3c
                dfb          $0a,$2a,$19,$3f
                dfb          $0a,$28,$0d,$3c
                skp          2
wsize           dfb          0
sgcnt           dfb          0
limlo           dfb          0,0,0,0,0,0,0,0
limhi           dfb          0,0,0,0,0,0,0,0
ofset           dfb          0,0,0,0,0,0,0,0
zzzend          ds           $2b00-*,0
* ram disk routines follow at $2a00 (*en3 - at $2b00*)
* dfb 0 ;this byte is needed to avoid a bug in edasm(*??en3??*)

; ###############################################################################################
; #   END OF FILE:  RELOC
; #   LINES      :  325
; #   CHARACTERS :  16023
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.REVISIONS.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  REVISIONS
; ###############################################################################################

                SBTL         'MLI Revision History'
*****************************************************************
*                                                               *
*                       C A U T I O N S                         *
*                                                               *
* 1.  In the language card area, the $D000 areas overlay.  To   *
*     determine which bank is in requires that the main bank    *
*     has a CLD instruction ($D8) at location $D000 and the     *
*     alternate bank must not.                                  *
*                                                               *
* 2.  Location $E000 is used to determine the state of ROM vs.  *
*     language card.  Therefore, the values of $E000 in the MLI *
*     and in the ROM MUST differ.                               *
*                                                               *
* 3.  In the file MEMMGR, the routines CALLDISP, RAMDRVR and    *
*     DOBUS each must access the other $D000 bank.  Therefore,  *
*     they MUST reside ABOVE $E000 in the language card area.   *
*                                                               *
* 4.  In order to tell the difference between a Disk II with a  *
*     DEVID of 0 and a Ram-based driver with a DEVID of 0, the  *
*     Disk II routines MUST start on a page boundary and the    *
*     ram-based driver caller (routine RAMDRVR in file MEMMGR)  *
*     MUST NOT start on a page boundary.                        *
*                                                               *
* 5.  In the ram disk driver (file RAM), the byte that assembles *
*     into location $FF58 MUST be a $60 (RTS) so that the usual *
*     routine to determine an I/O card's slot (by JSR $FF58)    *
*     will still work even if the Language card is switched in. *
*                                                               *
* 6.  RWTS calling routine at $FF90 is hard coded at that       *
*     address.  If /RAM driver moves or expands, the routine    *
*     must be moved and the dummy RWTS entry routine in XDOSMLI *
*     must be changed.                                          *
*                                                               *
* 7.  If the ram disk driver is to be moved (to make room more  *
*     code), the value of the symbol Srce must be updated to    *
*     reflect the change.  It is found in the file 'RAM' and    *
*     should point to where the basic builder 'build.prodos'    *
*     loads the RAM driver into memory.                         *
*                                                               *
* 8.  Any changes in the size of PRODOS/8 must be accompanied   *
*     by changes in P/16 load information... the size is hard   *
*     coded there.                                              *
*                                                               *
* 9.  WARNING!  There are at least four absolute patches in p8  *
*     from pquit and p16.  (1) Pquit "CallDisp" about p8 FCE3   *
*     (2) P16 ChangePath "TOTENT" about p8 FE8A                 *
*     (3) P16 Entry/IsItOpen about p16 96fc about p8 EFA3       *
*     "TSTOPEN"  (4) Pquit's PLOADER direct patch into p8's     *
*     block READ/WRITE dispatch "godev"                         *
*     All of these must be looked at if there are any changes to *
*     p8.                                                       *
*                                                               *
```

```
*****************************************************************
                PAGE
*****************************************************************
*
* NOTE: After each revision is made, the splash screen and the value
*       of KVERSION in the Global Page must be updated to reflect the
*
* Ver 1.0 - Written by J.R. Huston
*         - Updated and released by Dale M. Arends, Dec. 1983
*
* Ver 1.0.1 - by Dale M. Arends, January 1, 1984
*         1.  Bug in status call when testing write protect fixed.
*               (File: READ.WRITE)
*
* Ver 1.0.1A01 - Dale M. Arends, January 20, 1984
*         2.  Force FLUSH of FCB after EOF cutback.
*               (File: CLOSE.EOF)
*
* Ver 1.0.2 - Dale M. Arends, February 7, 1984
*         3.  Replace LDAs with BITs in Interrupt entry area of
*             Global page so it won't destroy Accumulator.
*               (Files: GLOBALS)
*         4.  Fixed bug in Disk core routines so that motor is
*             certain to shut off after recal on an error.
*               (File: XRW1)
*
* Ver 1.0.2A01 - Dale M. Arends, February 13, 1984
*         5.  Rearranged MLI layout within Language Card.
*               (All Files Affected)
*         6.  Rearranged entry to XRW routines to make room for
*             a CLD as first byte.  This is necessary since XRW
*             routines are now at $D000 and the CLD flags bank as
*             Main $D000 space.  Also added check for valid call
*             number (0-3).  (Both XRW files affected.)
*
* Ver 1.0.2A02 - Dale M. Arends, April 4, 1984
*         7.  Rewrote machine ID routine to allow for Phylums of
*             machines.    Ranges:  0n - //c
*                                   4n - ???
*               (File: PROLDR)      8n - ???
*                                   En - //e
*
*         8.  Removed support for boot in 48K machines.
*               (File: PROLDR)
*
* Ver 1.1XA01 - Dale M. Arends, May 3, 1984
*         9.  Modified Interrupt routine to allow for new ROMs in //c
*             and //e machines.
*               (Files: PROLDR, EQUATES, GLOBALS, XDOSMLI)
*        10.  Removed check for OS version number in file OPEN routine.
*               (File: POSN.OPEN)
*        11.  Incremented KVERSION number in Global page.
*               (Files: GLOBALS)
*        12.  Modified DEVSRCH routine to correctly reflect the
*             presence of any 80-col card following established
*             signature byte protocol.
*               (File: DEVSRCH)
*
* Ver 1.1XA02 - Dale M. Arends, May 11, 1984
*        13.  Added routine to allow for firmware in slot 3 to be
*             enabled. In //e, internal $C300 firmware is enabled
*             if no ROM in slot 3.  This is necessary for interrupts
*             and BRKs to work.  NOTE: any ROMs in slot 3 MUST support
```

```
*             the 80-column interrupt and BRK entry points or an
*             interupt or BRK will cause a system hang.
*               (File: PROLDR)
*        14.  Added routine to turn off all disk motor phases prior
*             to SEEKing a track on a Disk II call.
*               (File: XRW2)
*
* Ver 1.1XA02 renamed Ver 1.1 - Dale M. Arends, June 14, 1984
*
* Ver 1.1 - by Dale M. Arends (New date in splash screen 7-19-84)
*        15.  Added routine for accessing user installed drivers
*             located in AUX Language Card.
*               (Files: PROLDR, MEMMGR)
* Ver 1.1 - by Dale M. Arends (New date in splash screen 08-17-84)
*        16.  Added work-around for bug when accessing /RAM after it
*             has been removed from DEVLST.
*               (File: NEWFNDVOL)
*        17.  Reduced the RAM DISK by 1 block to protect the interrupt
*             vector area of the Aux Language Card.  Interrupt vectors
*             installed in the Aux Language Card at boot time.
*               (File: RAM, PROLDR)
*
* Ver 1.1.1 - by Dale M. Arends (New date in splash screen 9-18-84)
*        18.  Fixed routine in RSETPHSE to map into Drive 1.
*               (File: XRW2)
*        19.  Rewrote Machine ID routine to give I/O precedence to
*             an identifiable 80-column card in Slot 3.
*               (File: PROLDR)  (See #12 above.)
*
* Ver 1.2XA01 - by Dale M. Arends - 10/3/84
*        20.  Added MLI Call for AppleBus and driver calling code.
*              (Files: EQUATES, PROLDR, XDOSMLI, MEMMGR, DATATBLS)
*        21.  Fixed bug in error return of Ram Disk Driver.
*              (File: RAM)
*        22.  Assigned a SYSTEM ERROR when error occurs in deallocating
*             blocks from a TREE file.
*              (File: DETREE)
*        23.  Fixed bug in zeroing of a VCB when trying reallocating
*             a previously used VCB.
*              (File: NEWFNDVOL)
*        24.  Modified Loader to automatically install up to 4 drives if
*             the Smart Driver follows the defined $CnFE bit protocol.
*              (File: DEVSRCH)
*        25.  Modified MLI entry routines to disable interrupts until
*             after MLIACTV flag is set and CMDADR is set up.
*             (Note: change is not noted in source code.)
*              (Files: GLOBALS, XDOSMLI)
*
* Ver 1.2XA02 - Lost in the Wilds of the San Jose Triangle.
*        (Go to the Bermuda Triangle and turn left.)
*
* Ver 1.2XA03 - by Dale M. Arends - 1/24/85
*        26.  Modified Dispatcher to treat the DELETE key as a
*              Back Space (Left Arrow).
*              (File: SEL)
*
* Ver 1.2XA04 - by Dale M. Arends - 1/28/85
*        27.  Modified splash screen to reflect version control loss.
*              (File: DEVSRCH)
*
* Ver 1.2XA05 - by Kerry Laidlaw - 2/11/85
*        28.  Modified the ONLINE call to zero out a VCB corresponding
*             to a disconnected device.  Previously if a user followed
```

```
*             our method on disconnecting a device by removing the
*             volume's device number in the device list, and removing
*             the device handler's address from the device vector table,
*             ProDOS never removed the corresponding VCB entry.  Now
*             in addition to the above, the user can then issue a
*             specific ONLINE call to the user disconnected device.  A
*             no device connected error ($28) will be returned in the
*             accumulator, but the VCB will be taken off line.
*              (File: BFMGR)
*
* Ver 1.2XA06 - by Kerry Laidlaw - 3/20/85
*       29.  Fixed bug in PROLDR file that did not swap back correct
*             bank.
*              (File:PROLDR)
*       30.  Added the "real" AppleTalk drivers that will reside in
*             bank 2, $D400-$DFFF, in the Main Language Card.
*              (File:BUS)
*
* Ver 1.2xa08 - by Kerry Laidlaw - 5/15/85
*       31.  Updated Copyright Notices in source & in splash screen.
*              (Files:MLI,GLOBALS)
*       32.  Set KVERSION byte in global page to $82.  This version
*             ID byte is interpreted as follows:
*                The low 7 bits contain the ProDOS version number.
*                The high bit is set if this version of ProDOS
*                supports AppleTalk.
*              (Files:GLOBALS)
*       33.  Removed 4 drive install code (see #24 above) to prevent
*             problems with applications.  Some applications would
*             interpret unit numbers as mapping to physical slots.  Thus
*             even though unit numbers may only have a logical mapping,
*             these applications would assume a disk card is in the
*             physical slot.  Therefore if an I/O card is in that slot,
*             the application wouldn't look for it.
*              (File:DEVSRCH)
*
*       34.  Commented out one to many pla's from the dispatcher
*             calling routine.
*              (File:MEMMGR)
*
*       35.  Added an unclaimed (spurious) interrupt handler. It allows
*             255 unclaimed interrupts before going to system death.
*              (File:XDOSMLI)
*
*       36.  Removed low res mode evocation when displaying system
*             death messages.  (It was buggy to begin with and it saves
*             me 6 bytes:lda c083,lda c086).
*             Also removed ability to display system death message
*             without '-ERR xx' as part of the message.  This was done
*             to save space.  The system itself never used this ability
*             and hopefully no one else did either....
*              (Files:XDOSMLI,DATATBLS)
*
*       37.  Since the JSPARE vector in the global page was the only
*             vector to the SYS.RTS routine which will halt the system
*             with an 'INSERT SYSTEM DISK AND RESTART' message), and
*             JSPARE is overwritten by the loader with the dispatcher
*             calling vector, the 6 bytes starting at the SYS.RTS label
*             are now considered free space.
*              (File:GLOBALS)
*
*       38.  Deleted needless code that relocated Atalk drivers twice.
*             Also modified parameters that relocate interrupt file
```

```
*             MLI.3.
*              (File:PROLDR)
*
* Ver 1.2xa09 - by Kerry Laidlaw - 5/30/85
*        39.  Fixed bug in dispatcher that would trash screen holes
*             when code is called while in 80 column mode.  (Previously
*             HOME was called without the window size being updated to
*             a 40 column size.  Therefore the wrong BASCALC's were
*             done and zap!).  Now, I call SETTXT before calling home
*             which will set text mode and a 40 column window.
*             I also commented out a lda #0 to reclaim needed space.
*              (File:SEL)
*
*        40.  Made code that zeroed user's index block in his I/O
*             buffer into a subroutine.  Saved 8 bytes.
*              (Files:READ.WRITE,POSN.OPEN)
*
*        41.  Set a flag in the DELETE code to signal the DETREE routine
*             NOT to zero index blocks upon deallocation.  This was done
*             so developers could write file recovery routines.
*              (Files:DESTROY,DETREE,WRKSPACE)
*
* Ver 1.2xa10 - by Kerry Laidlaw - 6/4/85
*        42.  Replaced two lda's and bit's with one sta to hit the bank
*             switches (c083,c08b).  Note: I am counting on the fact
*             that once into ProDOS, the LC is already in a r/w state.
*             Thereafter, you only have to hit the bank switch once to
*             switch banks and maintain a r/w state in the new bank.
*             Also note that this was done to preserve zero flag status
*             upon return from the network drivers.  The previous BIT's
*             would destroy that status.
*              (File:MEMMGR)
*
*        43.  Forced the /RAM driver to return zeros on block read
*             requests to block 7, and do nothing on block writes to
*             block 7.  Previously, a block write to block 7 (an
*             operation that is invalid anyway), would bomb out into
*             the monitor.  Read and writes to blocks 0,1,4,5,6, and
*             now 7 are all treated in this manner.
*              (File:RAM)
*
* Version 1.2xa11 - by Kerry Laidlaw 6/12/85
*        44.  Removed disabling of interrupts from loader.  Therefore
*             the whether the processor allows interrupts or not is
*             left up to the monitor.
*              (File:DEVSRCH)
*
* Version 1.2xa12 - by Kerry Laidlaw 6/21/85
*        45.  Initialized main stack to $ff and set up $101 in Aux
*             memory (if it is there) to $ff so programs using the
*             aux stack will be started off correctly.
*              (File:PROLDR)
*
*        46.  Fixed 2 bugs in XDOSMLI file:
*             1) After processor is pulled to re-enable interrupts,
*                a CLD is now done to keep out of decimal mode.
*             2) Turning "off" MLIACTIVE flag is now done correctly
*                with an ASL after and initial ROR instead of the
*                original LSR.
*
* New version id for fear of version confusion 12/13/85...
* Version 1.2 exp 01 - by Kerry Laidlaw 12/13/85
*        47.  The OPEN call will now return error $4B (unsupported or
```

```
*             incorrect storage type) instead of $4A (incompatible file
*             format for this version) when determining if it is legal
*             to open a file of a particular storage type.
*               (File: POSN.OPEN)
*
* Version 1.2 exp 02 - by Kerry Laidlaw 2/3/86
*        48.  Splash screen now shows "PRODOS 8" instead of "PRODOS".
*             Fixed a bug in the read call that would return the
*             incorrect number of bytes transferred when newline was
*             enabled.  If the requested number of bytes was greater
*             than $00FF, AND the number of bytes in the file AFTER the
*             newline character was READ was a multiple of $100, then
*             the number of bytes reported transferred by ProDOS was
*             equal to the correct number of transferred bytes + $100.
*               (Files: DEVSRCH, READ.WRITE)
*
*        49.  Removed fix from rev #41 that prevented index blocks
*             from being zeroed after files were deleted.
*              (Files: DESTROY, DETREE, WRKSPACE)
*
*        50.  Added new clock driver to read the Cortland builtin clock.
*             The loader will now check for a Cortland to decide which
*             clock driver (slot or built-in) to install.
*             Stopped messing with slot 3 rom space if running on a
*             Cortland.  The Control Program lets the user determine
*             internal vs. external slot space.
*             Also commented out bus references in equates file, since
*             it has now been decided that there be no Appletalk drivers
*             in this version of ProDOS8.
*              (Files: EQUATES, PROLDR, CCLOCK)
*
* Version 1.2 exp 03 - by Kerry Laidlaw 2/20/86
*        51.  Updated slot clock driver's year lookup table.
*             The year can now be calculated thru 1991.
*              (File: TCLOCK)
*
* Version 1.2 exp 04 - by Kerry Laidlaw 3/07/86
*        52.  Updated Cortland clock driver to adjust to new
*             miscellaneous toolset call number.
*
* Version 1.2 exp 05 - by Kerry Laidlaw 3/13/86
*        53.  Updated DEVSRCH to special case a Smartport card
*             in slot 5 to allow 4 Smartport units. Driver entry
*             addresses for these Smartport units would be placed in
*             the ProDOS global page's slot 5 and slot 2 driver vectors.
*             ProDOS unit numbers $50 & $D0 (slot 5, drives 1 and 2)
*             will correspond to Smartport units 1 and 2.  ProDOS unit
*             numbers $20 and $A0 (Slot 2 drives 1 and 2), will
*             correspond to Smartport units 3 and 4.
*              (File: DEVSRCH)
*
* Version 1.2 exp 06 - by Kerry Laidlaw 04/03/86
*        54.  Updated Smartport unit special casing (see rev note #53)
*             to ensure the extra two unit numbers to have a $B in
*             the low nibble so the FILER won't have any problems
*             with formatting.  Also set version id byte (KVERSION) in
*             global page to $02 representing version 1.2.
*              (Files: DEVSRCH, GLOBALS)
*
* Version 1.2 D1 - by Kerry Laidlaw 06/22/86
*        55.  Crunched the dispatcher code to make room so the correct
*             soft switches and monitor routines could be called to
*             go to 40 columns and display text.
```

```
*              (File: SEL)
*
*       56.  Added feature to loader that will load and JSR to the
*            file "ATINIT" in the booting volume's root directory.
*            If the file is not found, no error is reported, and
*            the ".SYSTEM" file is executed as normal.  But if an
*            I/O error other than file not found is encountered
*            or if the file type is not the ATINIT type of $f8
*            while loading "ATINIT", a fatal error is displayed.
*              (File: PROLDR)
*
*       57.  Added an alternate loader entry point.
*            The new entry is at $2003 and is intended to be called
*            be called only by the network boot code when booting
*            over the net, and by ProDOS 16 v1.0 on the Apple //-16.
*            The loader through this entry point will
*            NOT load the ATINIT file and will NOT load the
*            .SYSTEM file.  It will set up ProDOS and RTS back
*            to the caller to this entry point.
*            Because of this feature, the initialization of the
*            stack done in rev Note #45, was taken out....
*              (File: PROLDR)
*
*       58.  Added code to set MACHID byte to indicate a clock
*            was present when ProDOS is run on an Apple //-16.
*              (File: PROLDR)
*
* Version 1.2A1 - by Kerry Laidlaw 06/30/86
*       59.  ATINIT file official file type now checked for is
*            $E2.
*              (File: PROLDR)
*
* Version 1.2A2 - by Mike Askins   08/25/86
*       60.  The device searching process has changed.  SmartPort
*            devices are only installed if they exist and Disk //s
*            are placed in the device list so that they have the
*            lowest priority in device scans.  This should regularize
*            device numbering and naming in ProDOS/16.  Note that this
*            change caused the loader to grow by 256 bytes, and
*            everything beyond devsrch is moved by one page.
*              (File: DEVSRCH)
*
* Version 1.2B1 - by Mike Askins   08/25/86
*       61.  The loader now passes a flag to Cortland at $FEFF to
*            indicate that we're running on a Cortland.  At system
*            death, if we're on a Cortland, a zero is stuffed into
*            $C029 disabling super hires so that the user can see the
*            system death message.
*              (Files: PROLDR, XDOSMLI, WRKSPACE)
*
*       62.  A parameter os.boot is now passed to Cortland AppleTalk.
*            If it is a stand alone ProDOS/8 (not P8/16 config) a
*            zero is stored in $E100BD.  Two branches had to be
*            extended to accomodate the code which does this (these
*            are marked).
*              (File: PROLDR)
*
*       63.  Two DS directives were altered (" ,0 " added) so that
*            they generated bytes with a fixed value.  Successive
*            builds should now compare byte for byte EXCEPT for the
*            date string (around 1550 decimal into PRODOS file).
*              (File: SEL)
*
```

```
*       64.  The logic of the newly added SmartPort handler in the
*            loader has been changed to fix a bug in V1.2A2 which
*            prevents installation of SmartPort devices in any slot
*            OTHER than 5.  The check for slot five was moved down
*            past the installation of drives 1 and/or 2.
*             (File: DEVSRCH)
*
* Version 1.2B2 - by Mike Askins   09/06/86
*       65.  A bug is fixed in the loader which caused disk devices
*           in slot 2 to conflict ungracefully with more than two
*           devices in slot 5.  Now, if any disk device is detected
*           in slot 2, NO MORE THAN 2 units are installed in slot 5.
*            (File: DEVSRCH)
*
*       66.  The Cortland Clock Driver now sets main memory before
*           doing the call to the miscellaneous tools to read the
*           clock hardware.  The main/aux state is then restored.
*
* Version 1.2B3 - by Kerry Laidlaw 09/07/86
*       67. Updated cclock routine to correct the anding of the
*           state register.
*            (File:CCLOCK)
*
* Version 1.2   - by Mike Askins   09/08/86
*       No changes
*
* Version 1.2   - by Mike Askins   09/23/86
*       68. Inserted a patch into RELOC to fix a bug in the 1.0 GS
*           ROM which caused the internal CX ROM space to be left
*           mapped in after a boot failure on SmartPort.  See the
*           documentation in RELOC to find out why the patch was
*           placed here (it has nothing to do with the RELOCation
*           process!).
*
* Version 1.2   - by Kerry Laidlaw 09/23/86
*       69. Fixed a bug in the dispatcher that trashes screen when
*           executed with the prefix set to nil.
*           Previously when an error occured when the pathname
*           was checked, the message at the top of the screen
*           was redisplayed. Now we are jumping so the message
*           is not redisplayed.  It saves us 2 bytes....
*            (File:SEL)
*
* Version 1.3   - by Mike Askins, Fern Bachman - 10/17/86
*       70. The code that resets the phase lines for Disk //s has been
*           changed so that the phase clearing is done with a load, not
*           a store.  Stores to even numbered locations on the disk
*           hardware cause data bus contention which is just not cool.
*           The routine has been changed to access all 8 even locations
*           which not only clears the phases but forces read mode (as
*           well as 1st drive and motor off) on entry to the disk code.
*           DOS used to do this and ProDOS did not.  If for some reason
*           (copy protection, runaway programs) l7 was left high on
*           entry to the disk routines, and the disk routines checked
*           write protect with l6 high:  BINGO -- we're in write mode.
*           Forcing read mode initially leaves less to chance.
*           to assemble ProDOS8 with cross reference tables.
*       71. The file MLI was changed to allow ASM816 version 1.4.2 to
*           allow for cross reference listings to be generated.
*           Assembling the file MLI generates ProDOS8 without listings.
*           Assembling the file L.MLI generates PorDOS8 with a listing.
*           NOTE: A blank RAM disk card called /RAM1 must be installed
*                 in the system to cross references to be generated.
```

```
*        72. The 'date' directive was removed in the file DEVSRCH
*            and an ASC 'xx-xxx-xx' directive replaced it.
*
* Version 1.3   - by Ray Chiang   11/07/86
*        73. In reference to rev notes 41 and 49, index blocks now
*            have their contents flipped when the file is deleted
*            and their contents are fully zeroed when EOF cutback
*            occurs.  This technique allows scavenging programs to
*            possibly recover files that were accidently deleted.
*            Also the possibility of unclaimed blocks 'looking'
*            like valid index blocks is reduced.  Previous to this
*            revision, whole index blocks from tree files were
*            being released back into the disk.
*            (files: DESTROY,DETREE,WRKSPACE)
*
* Version 1.3   - by Fern Bachman 11/17/86
*        74. The SmartPort interface card (our Liron interface card)
*            does not set up its device chain unless a ProDOS call
*            is made to it.  Therefore in DEVSRCH where Mike makes
*            SmartPort calls the card is not setup yet so it returns
*            00 devices in its chain.  This results in the card/drives
*            never being seen by ProDOS.  To fix that we will now make
*            a ProDOS status call before SmartPort status calls.  This
*            should guarantee valid device counts.
*            (file: DEVSRCH)
*
* Version 1.4   - by Ray Chiang/Fern Bachman 02/03/87
*        75. Fix illegal 65C02 instruction in 6502 code.  ie change
*            the BRA to a BEQ instruction.
*
* Version 1.4   - by Mike Askins/Fern Bachman 02/12/87
*        76. Changed RSETPHSE routine to only clear the phase lines.
*            And we clear Q7 so no writes can accidently occur.
*
* Version 1.4  - by Ray Chiang/Mike Askins 03/23/87
*        77. Simple programming error in fix 76 .
*
* Version 1.4 - Ray Chiang 04/17/87
*        78. No changes, released to SCM
*
* Version 1.5d1 - Grady Ward 2 Sep 87
*         79.  Fixed overrun problem in READ.WRITE
*              Also, changed p16 patch addresses
*
*         80.  Made call to DEVMGR surrounded by NOP's so that
*              it can be patched by p16 and pquit more easily
*              (for both cache and AppleTalk booting)
*
*
* Version 1.5d7 - Grady Ward 20 Nov 87 Summary of Changes
*         81.  Change to READ.WRITE in tstwprot to disable interrupts
*              if going to use DMGR (Caching)
*
*         82.  P8 code has been extensively moved about to make room
*              for patches to DMGR dispatch (approximately 14 bytes
*              were needed for AppleTalk booting patches caching patches)
*
* Version 1.5d8 4-Dec-87
*
*          83.  Change some asc strings, like "system" uppercase.
*
* Version 1.5d11 20-Jan-88
*
```

```
*           84.  Took out $42 generic AppleTalk Call
*
* Version 1.5b2 16-Feb-88
*
*            85.  Restored asl mliactv to lsr in "exitmli"
*
* Version 1.5b3 1-Mar-88
*
*            86. Clear ESC in keyboard if booting on Apple //c
*                ("slow" down for Zip Chip)
* Version 1.6 14-Jun-88
*
*            87. Set up parallel pfixptr to fix
*                AppleTalk PFI misinterpretation
*                of two's comp prefix pointer
*
* Version 1.7 22-July-88
*
*            88.  When setting eof backwards, make sure
*                 volumes bitmaps are counted before inc
*                 free blocks
*
*            89.  When installing ramdisk, make sure slot3rom
*                 is preserved rather than overwritten
*
*            90.  In system death, check "cortflag" rather than
*                 "cortland" to ascertain what system we're on
*
*             91.  Waited to restore slot3 firmware until just before
*                  gquit is called at jspare
*
*             92.  Now permit invisible bit to be set with set file info
*                  (bit 2 of d.attr byte)
*
*********************************************************************
*
*
* EdNet v1.0x01 - by Kerry Laidlaw - 6/5/85
*+++++++     The following changes are made to support transparent
*            file access for the AppleTalk network.  This version
*            will be the user (student) stations ProDOS.  It has been
*            decided that this version of ProDOS will only support
*            128k systems.
*
*      EN1.  The RWTS routines are now running in LC, b2 above
*            the dispatcher code ($D400-$DAE9).  To do this the
*            loader was changed to load the RWTS routines into bank 2
*            starting at $d400, and a dummy RWTS entry at $D000 b1,
*            was entered which only does a CLD followed by a JMP $FF90.
*            The routine at $FF90 is at the end of the RAM code file
*            where a few extra bytes were dangling.
* NOTE: FOR ANY FUTURE MAINTAINER of this code.  As of 5/88, all
* ProDOS 8, 16, and BASIC.SYSTEM bugs are archived as ProDOS bug
* in the B.R.C. archive, priority 5.  Please look there before
* making any changes to p8 or p16.  Grady Ward
*
*      EN2.  Modified the block file manager to go to the AppleTalk
*            command when the pathname corresponds to /ATALK.
*            The new DOATALK file will house all the AppleTalk interface
*            command handling.
*            The MLI.BUILD BASIC program was updated to handle 7 more
*            pages of code in the MLI.2 file.
*             (Files:BFMGR,DATATBLS,DOATALK,WRKSPACE)
```

```
*
*      EN3.  Changed loader to relocate Atalk drivers to Aux LC.
*            Also changed ATP caller to go thru the "bridge" mechanism.
*            This bridge routine is so named because its purpose is to
*            get an entry address to ATP, FAP etc., as contained in
*            the address table labled ATALKTBL, and jsr to the
*            routine in the Aux LC.  This address table is
*            located at the beginning of the ATP driver file COMMANDS.
*            At load time, this table is copied into space reserved
*            for it in the file XDOSMLI.
*            Also, the ATP interrupt handler has been pre-installed
*            into ProDOS in the file XDOSMLI.
*            The RPM STUB that will be called when characters are
*            written to the AppleTalk card was placed in the file
*            MEMMGR.  The Loader now installs the hooks to the RPM
*            STUB in the AppleTalk card's screen holes.
*             (Files:PROLDR,MEMMGR,XDOSMLI,COMMANDS)
*
**************************************************

; ###############################################################################################
; #   END OF FILE:  REVISIONS
; #   LINES      :  639
; #   CHARACTERS :  31226
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.ROM.pretty
================================================================================

; ###########################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  ROM
; ###########################################################################################

               sbtl         'lang. cd. irq, nmi, & reset'
               org          $ff9b
*
nmi            equ          $3fb
acc            equ          $45
*
lanirq         equ          *
               pha
               lda          acc
               sta          old45
               pla
               sta          acc
               pla                                ;get status register from stack.
               pha                                ;(and restore it!)
               and          #$10                  ;is it a break or interupt
               bne          lbreak                ;branch if break
               lda          $d000                 ;get bankid
               eor          #$d8                  ;is the system active?
               beq          sysactv               ;branch if it is.
               lda          #$ff
sysactv        sta          intbankid
               sta          afbank
               lda          #<aftirq
               pha                                ;set up return address
               lda          #>aftirq
               pha
               lda          #4                    ;status reg w/int flag set
               pha
lbreak         lda          #<romirq              ;push rom entry also
               pha
               lda          #>romirq
               pha
gorom          sta          romin                 ;switch to rom (hits rts immediately).
*
lreset         lda          rreset+1
               pha                                ;since reset, acc can be destroyed.
               lda          rreset
               pha
               jmp          gorom
*
rreset         dw           $fa61
fix45          sta          intareg               ;preserve the acc
               lda          old45
               sta          acc
               lda          ramin
               lda          ramin                 ;switch ram in for write & read
               lda          afbank
               jmp          irqxit0
*
*
xxx            equ          *
stypfx         sty          newpfxptr             ; fix AppleTalk PFI bug
               sty          pfixptr
```

```
                rts
stapfx          sta          newpfxptr
                sta          pfixptr
                rts
*fil equ $fffa-xxx
* ds fil,0
*
                dw           nmi
                dw           lreset
                dw           lanirq

; ###############################################################################################
; #   END OF FILE:  ROM
; #   LINES      :  63
; #   CHARACTERS :  2440
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.WRKSPACE.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  WRKSPACE
; ###############################################################################################

               page
*
own.blok       dw          $0000
own.ent        db          0
own.len        db          0
*
h.credt        dw          $0000                    ;directory creation date
               dw          $0000                    ;directory creation time
               db          0                        ;version under which this directory was
created
               db          0                        ;earliest version that it's compatable
with
h.attr         db          0                        ;attributes (protect bit, etc.)
h.entln        db          0                        ;length of each entry in this
directory.
h.maxent       db          0                        ;maximum number of entries per block
h.fcnt         dw          $0000                    ;current number of files in this
directory
h.bmap         dw          $0000                    ;address of first allocation bit map
h.tblk         dw          $0000                    ;total number of blocks on this unit
*
d.dev          db          0                        ;device number of this directory entry
d.head         dw          $0000                    ;address of <sub> directory header
d.entblk       dw          $0000                    ;address of block which contains this
entry
d.entnum       db          0                        ;entry number within block.
dfil           equ         *
d.stor         equ         *-dfil                   ;storage type * 16 + file name length
               db          0
; *-dfil ; file name
               ds          15,0
d.filid        equ         *-dfil                   ;user's identification byte
               db          0
d.frst         equ         *-dfil                   ;first block of file
               dw          $0000
d.usage        equ         *-dfil                   ;number of blocks currently allocated
to this file
               dw          $0000
d.eof          equ         *-dfil                   ;current end of file marker
               dfb         0,0,0
d.credt        equ         *-dfil                   ;date of file's creation
               dw          $0000
d.cretm        equ         *-dfil                   ;time of file's creation
               dw          $0000
d.sosver       equ         *-dfil                   ;sos version that created this file
               db          0
d.comp         equ         *-dfil                   ;backward version compatabilty
               db          0
d.attr         equ         *-dfil                   ;'protect', read/write 'enable' etc.
               db          0
d.auxid        equ         *-dfil                   ;user auxillary identifacation
               dw          $0000
d.moddt        equ         *-dfil                   ;file's last modification date
```

```
                dw          $0000
d.modtm         equ         *-dfil                  ;file's last modification time
                dw          $0000
d.dhdr          equ         *-dfil                  ;header block address of file's
directory
                dw          $0000
*
scrtch          ds          4,0                     ;scratch area for allocation address
conversion
oldeof          dfb         0,0,0                   ; temp used in w/r
oldmark         dfb         0,0,0                   ; used by 'rdposn' and 'write'
*
xvcbptr         dfb         0                       ;used in 'cmpvcb' as a temp.
vcbptr          db          0
fcbptr          db          0
fcbflg          db          0
*
*
reql            db          0
reqh            db          0
levels          db          0
totent          db          0
entcntl         db          0
entcnth         db          0
cntent          db          0
nofree          db          0
bmcnt           db          0
saptr           db          0
pathcnt         db          0
p.dev           db          0
p.blok          dw          $0000
bmptr           db          0
basval          db          0
half            db          0
*
*
                page
*
* bit map info tables (a & b)
*
bmastat         db          0
bmadev          db          0
bmadadr         dw          $0000
bmacmap         db          0                       ; similar to vcbcmap
*
tposll          db          0
tposlh          db          0
tposhi          db          0
rwreql          db          0
rwreqh          db          0
bulkcnt         db          0
nlchar          db          0
nlmask          db          0
ioaccess        db          0                       ;has a call been made to disk device
handler?
cmdtemp         db          0                       ;test refnum, time, and dskswtch for
(pre)processing.
*
bkbitflg        db          0                       ;used for drevise to set or clear back
up bit.
duplflag        db          0                       ;difference between vnferr and dupvol
by synpath
vcbentry        db          0                       ; pointer to current vcb entry
```

```
*
*
*
* xdos temporaries added....
*
command         db          0
namcnt          db          0
rnptr           db          0
pnptr           equ         *
namptr          db          0
vnptr           db          0
prfxflg         db          0
tempx           db          0
cferr           db          0
*
*
* the following are used for deallocation temps.
*
firstbl         db          0
firstbh         db          0
stortype        db          0
deblock         dw          $0000
dtree           db          0
dsap            db          0
dseed           dw          $0000
topdest         db          0
dtmpx           db          0
dealbufl        ds          8,0
dealbufh        ds          8,0
*
*
loklst          equ         dealbufl
*
cbytes          dw          $0000
                db          $00                     ;cbytes+2 must always be zero!!! see
"calcmark"
*
bufaddrl        db          0
bufaddrh        db          0                       ;buffer allocation, getbuffr, and
release buffer temps.
*
svstack         ds          $10,0
svzerop         ds          $6,0
goadr           dw          $0000
*
****************** see rev note #73 ********************
******************** see rev note #49 ***************
************** see rev note #41 *************
delflag         ds          1,0                     ; used by detree to know if called from
delete
*****************************************
*
************** see rev note #en2 *************
                ifeq        os-ednet
atpfxflg        db          0                       ; = 0 --> prefix not /atalk, <> 0 is.
                fin
*****************************************
                skp         2
                asc         '(C)1983-87APPLE'
                skp         2
                ds          $feff-*,0
                skp         2
*********************************************************
```

```
*
* this flag is set in the loader indicating whether the
*  machine is a cortland.
*
*
cortflag        dfb          0                        ;0=>non cortland / 1=>cortland
*
**************** see rev note #61 **********************

; ###############################################################################################
; #   END OF FILE:  WRKSPACE
; #   LINES      :  172
; #   CHARACTERS :  7133
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.XDOSMLI.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  XDOSMLI
; ###############################################################################################

                sbtl         'prodos machine language interface'
                org          orig1
* * * * * * * * * * * * * * * * * *
*    the xdos machine language  *
*         interface (mli)       *
*      system call processor    *
* * * * * * * * * * * * * * * * * *
*
                ifeq         os-ednet
*
************** see en#1 ***************
*
                cld                                     ; this is the entry for the disk ii
rwts.
                jmp          $ff90                      ;
****************************************
*
************ see rev note #en3 ****************
*
                msb          off
                asc          "(C)APPLE'83"
*
** the following address table consists of entry points into appletalk
** resident in the aux lc.  at load time, it is copied from a table in
** the file commands in the atp drivers where it resides at $d000, alc.
*
atalktbl        equ          *
                ds           32,0                       ; reserve space for 16 entry points.
*
***********************************************
*
                fin
entrymli        cld                                     ;cannot deal with decimal mode!!!
                pla                                     ;get processor status
                sta          spare1                     ;save it on the global page temporarily
                sty          savey                      ;preserve the y and x registers.
                stx          savex
                pla                                     ;find out the address of the caller.
                sta          par
                clc                                     ;and preserve the address of the call
spec.
                adc          #4
                sta          cmdadr
                pla
                sta          par+1
                adc          #0
                sta          cmdadr+1                   ;cmdadr is in the globals.
                lda          spare1
                pha
                plp                                     ;pull processor to re-enable interrupts
                cld                                     ; no decimal! (** #46 **)
                ldy          #0
                sty          serr                       ;clear any previous errors...
```

```
                iny                             ;find out if we've got a valid command.
                lda         (par),y             ;get command #
                lsr         a                   ;hash it to a range of 0-$1f.
                lsr         a
                lsr         a
                lsr         a
                clc
                adc         (par),y
                and         #$1f
                tax
                lda         (par),y             ;now see if result is a valid command
number.
                cmp         scnums,x            ;should match if it is.
                bne         scnerr              ;branch if not.
                iny                             ;index to call spec parm list addr.
                lda         (par),y             ;make par point to parameter count byte
                pha                             ;  in parameter block.
                iny
                lda         (par),y
                sta         par+1
                pla
                sta         par
                ldy         #c.pcnt             ;now make sure parameter list has the
                lda         pcntbl,x            ;  proper number of parameters
                beq         goclock             ;clock has 0 parameters
                cmp         (par),y
                bne         scperr              ;report error if wrong count.
                page
                lda         scnums,x            ;get call number again.
                cmp         #$65
                beq         special
                asl         a                   ;carry set if bfm or dev mgr.
                bpl         godevmgr
                bcs         gobfmgr
                lsr         a                   ;shift back down for interupt mgr.
                and         #$3                 ;valid calls are 0 & 1
                jsr         intmgr
                jmp         exitmli             ;command processed, all done.
*
special         equ         *
                jmp         jspare
*
goclock         jsr         datetime            ;go read clock.
                jmp         exitmli             ;no errors posible!
*
godevmgr        lsr         a                   ;save command #
                adc         #1                  ;valid commands are 1 & 2
                sta         dhpcmd
                jsr         devmgr              ;execute read or write request.
                jmp         exitmli
*
gobfmgr         lsr         a
                and         #$1f                ;valid commands in range of 0-$13
                tax
                jsr         bfmgr               ;go do it...
*
exitmli         lda         #0                  ;first clear bubit.
                sta         bubit               ;(back up bit)
                ldy         serr                ;y holds error code thru most of exit.
                cpy         #1                  ;if >0 then set carry
                tya                             ; and set z flag.
                php                             ;disable interupts until exit complete.
                sei
```

```
                lsr         mliactv                 ;indicate mli done.(** #46 **) (** #85
**)
                pla                                 ;save status register elswhere
                tax                                 ; until return address is placed
                lda         cmdadr+1                ; on the stack.
                pha
                lda         cmdadr                  ;returning is done via 'rti'
                pha                                 ; so that the status register is
                txa                                 ; restored at the same time.
                pha                                 ;place status back on the stack.
                tya                                 ;return error, if any.
                ldx         savex                   ;restore x & y registers.
                ldy         savey
exitrpm         pha                                 ; (exit point for rpm **en3**)
                lda         bnkbyt1                 ;restore language card status & return.
                jmp         exit
*
gnodev          lda         #xnodrv                 ;report no device connected.
                jsr         syserr
*
scnerr          lda         #badscnum               ;report no such command.
                bne         scerr1                  ;branch always
*
scperr          lda         #badscpcnt              ;report parameter count is invalid.
scerr1          jsr         gosyserr
                bcs         exitmli                 ;branch always taken.
*
                sbtl        'prodos device manager'
devmgr          ldy         #5                      ;the call spec for devices must
                php                                 ;do not allow interupts.
                sei
dvmgr1          lda         (par),y                 ; be passed to drivers in zero page.
                sta         dhpcmd,y                ;dhpcmd,unitnum,bufptr,blocknum.
                dey
                bne         dvmgr1
                ldx         dbufph
                stx         usrbuf+1
                inx
                inx                                 ;add 2 for 512 byte range
                lda         dbufpl                  ;is buffer page alligned?
                beq         dvmgr2                  ;branch if it is.
                inx                                 ;else account for 3-page straddle...
dvmgr2          jsr         vldbuf1                 ;make sure user is not conflicting
                bcs         dvmgrerr                ; with protected ram
                jsr         dmgr                    ;call internal entry for device
dispatch.
                bcs         dvmgrerr                ;branch if error occured.
                plp
                clc                                 ;make sure carry is clear (no error)
gl.rts          rts
dvmgrerr        plp                                 ;restore interupt status.
gosyserr        jsr         syserr
*
*
*  NOTE: interrupts must always be off when entering here
*
dmgr            lda         unitnum                 ;get device number.
                and         #$f0                    ;strip misc lower nibble
                sta         unitnum                 ;  and save it back.
                lsr         a                       ;use as index to device table.
                lsr         a
                lsr         a
                tax
```

```
                lda         devadr01,x              ;fetch driver address.
                sta         goadr
                lda         devadr01+1,x
                sta         goadr+1
                jmp         (goadr)                 ;goto driver (or error if no driver)
                nop
                nop
                nop
                rts
*
                sbtl        'prodos interrupt manager'
intmgr          equ         *
                sta         intcmd                  ;allocate intrupt or deallocate?
* -------------------- see rev note 20 ---------------------------
***************** see rev note #en3 *****************
                cmp         #2                      ;alloc & dealloc = 0 & 1; 2 = atalk
                bcc         intmgr1
                ifeq        os-prodos
                jsr         dobus                   ; go set up and call atp drivers.
                fin
                ifeq        os-ednet
                ldx         par                     ; x and y must be used to call atp
since
                ldy         par+1                   ; main zero page is not in when atp
runs.
                inx                                 ; inc parameter list pointer by one
                bne         par1                    ; to bypass the dummy length field.
                iny                                 ;
par1            equ         *
                php                                 ; now we must set up the stack the way
                pha                                 ; the bridge routine wants it....
                txa                                 ;
                pha                                 ;
                lda         #0                      ; offset into atalk address table for
atp.
                jsr         bridge                  ; go call atp drivers...
                fin
                bcs         interr1                 ;signify error
                rts
intmgr1         equ         *
* ------------------------------------------------------------
                lsr         a                       ;(acc=0, carry set=dealloc)
                bcs         dealcint                ;branch if deallocation.
                ldx         #3                      ;test for a free interupt space in
table.
alcint          lda         intrupt1-2,x            ;test high addr for zero.
                bne         alcint1                 ;branch if spot occupied.
                ldy         #c.intadr+1             ;fetch address of routine.
                lda         (par),y                 ;must not be in zero page!!!!
                beq         badint                  ;branch if the fool tried it.
                sta         intrupt1-2,x            ;save high address
                dey
                lda         (par),y
                sta         intrupt1-3,x            ;and low address.
                txa                                 ;now return interupt # in range of 1 to
4.
                lsr         a
                dey
                sta         (par),y                 ;pass back to user.
                clc                                 ;indicate success!
                rts
*
alcint1         inx
```

```
                inx                             ;bump to next lower priority spot.
                cpx         #$b                 ;are all four allocated already?
                bne         alcint              ;branch if not.
*
                lda         #intblful           ;return news that four devices are
active.
                bne         interr1
*
badint          lda         #badlstcnt          ;report invalid parameter.
interr1         jsr         syserr
*
*
dealcint        ldy         #c.intnum           ;zero out interupt vector.
                lda         (par),y             ;but make sure it is valid #
                beq         badint              ;branch if it's <1
                cmp         #5                  ; or >4
                bcs         badint
                asl         a
                tax
                lda         #0                  ;now zip it.
                sta         intrupt1-2,x
                sta         intrupt1-1,x
                clc
                rts
                page
irqrecev        equ         *
                lda         $45                 ;get acc from 0-page where old rom put
it
                sta         intareg
                stx         intxreg             ;entry point on ram card interupt
                sty         intyreg
                tsx
                stx         intsreg
                lda         irqflag             ;irq flag byte = 0 if old roms
                bne         irqrcv1             ;  and 1 if new roms
                pla
                sta         intpreg
                pla
                sta         intaddr
                pla
                sta         intaddr+1
irqrcv1         txs                             ;restore return addr & p-reg to stack
                lda         $7f8                ;set up to re-enable $cn00 rom.
                sta         irqdev+2
                tsx                             ;make sure stack has room for 16 bytes
                bmi         nostsve             ;branch if stack safe.
                ldy         #$f
stksve          pla
                sta         svstack,y
                dey
                bpl         stksve
nostsve         ldx         #$fa                ;save 6 bytes of zero page
zpgsve          lda         $0,x
                sta         svzerop-$fa,x
                inx
                bne         zpgsve
*
* poll interupt routines for a claimer.
*
                lda         intrupt1+1          ;test for valid routine.
                beq         intr2               ;branch if no routine.
                jsr         goint1
                bcc         irqdone
```

```
intr2           lda         intrupt2+1              ;test for valid routine.
                beq         intr3                   ;branch if no routine.
                jsr         goint2                  ;execute routine.
                bcc         irqdone
intr3           lda         intrupt3+1              ;test for valid routine.
                beq         intr4                   ;branch if no routine.
                jsr         goint3
                bcc         irqdone
intr4           lda         intrupt4+1              ;test for valid routine.
                ifeq        os-prodos
                beq         irqdeath
                fin
                ifeq        os-ednet
                beq         intr5                   ;branch and check if atalk interrupt..
                fin
                jsr         goint4                  ;execute routine.
                bcc         irqdone
*
*************** see rev note #en3 **************
*
* the following code pre-installs the appletalk interrupt handler.
* this is done so as not to take up a user interrupt space.
* note that registers are saved and restored by prodos.
*
                ifeq        os-ednet
intr5           equ         *
                inc         $410                    ; ** for debug purposes only...
                php                                 ; now set up stack for bridge....p....
                pha                                 ; a...
                tax                                 ; x...
                pha                                 ;
                lda         #4                      ; offset into atalk address table for
atp
                bit         $ff58                   ; set overflow to signify bridge called
during interrupts.
                jsr         bridge                  ; interrupt handler.
                bcc         irqdone                 ; if carry clear, then interrupt
processed
                fin
***********************************************
*
*************** see rev note #35 ************************
irqdeath        inc         irqcount                ; allow 255 unclaimed interrupts before
                bne         irqdone                 ; going to system death...
*************************************************************
*
                lda         #badirq                 ;kill the system, no one claimed the
interupt.
                jsr         sysdeath
*
*
irqdone         ldx         #$fa
irqdne1         lda         svzerop-$fa,x           ;restore zero page stuff.
                sta         $0,x
                inx
                bne         irqdne1
                ldx         intsreg                 ;test for necessity of restoring stack
elements.
                bmi         irqdne3
                ldy         #$00
irqdne2         lda         svstack,y
                pha
                iny
```

```
                cpy         #$10
                bne         irqdne2
irqdne3         equ         *
                lda         irqflag                 ;check for old roms
                bne         irqdne                  ;branch if new roms
                ldy         intyreg                 ;restore registers.
                ldx         intxreg
                lda         $cfff                   ;re-enable i/o card
irqdev          lda         $c100                   ;warning, self modified.
                lda         irqdev+2                ;restore device id
                sta         $7f8                    ;just in case...
irqdne          jmp         irqxit                  ;do necessary bank switches and return.
*
irqflag         dfb         0                       ;irq flag byte. 0=old roms; 1=new roms
irqcount        db          0                       ; unclaimed interrupt counter.(note
#35)
*
goint1          jmp         (intrupt1)
goint2          jmp         (intrupt2)
goint3          jmp         (intrupt3)
goint4          jmp         (intrupt4)
                page
*
syserr1         sta         serr
                pla
                pla                                 ;pop 1 level of return.
                lda         serr
                sec
                rts
*
line23          equ         $750
line24          equ         $7d0
*
*
*************** see rev note #36 *************************
*
*sys.end lda #0 ;no error code, intentional cold start needed
sysdeath1       tax                                 ;system death!!!
                sta         $c00c                   ;force 40 columns on rev-e.
                lda         $c051                   ; text mode on.
*
*************** see rev note #61 *************************
*
                lda         cortflag                ;check if we're on a cortland
                beq         nosuphires              ;if not, don't touch shires switch
                lda         #0                      ;jam all bits off in shires video byte
                sta         $c029                   ;force off super hires
nosuphires      equ         *
*
************************************************************
*
* lda $c053 ; go to mixed mode (if above switch was for graphics!)
                lda         $c054                   ; display page 1 on.
* lda $c056 ; go to low res mode (if graphics enabled).
                ldy         #$27                    ;move 40 characters
dsdeath         lda         #$a0                    ;blank second to the bottom line
                sta         line23,y
                lda         death,y
                sta         line24,y
                dey
                bpl         dsdeath
* txa
* beq halt
```

```
* lda #'-'+$80
* sta line24+$21
* lda #'e'+$80 ;put err in death line if
* sta line24+$22 ;  sysdeath caused by an error
* lda #'r'+$80
* sta line24+$23
* sta line24+$24
* txa
* lsr a
* lsr a
* lsr a
* lsr a
* ora #$b0
* cmp #$ba
* bcc pq1 ;branch if not >9
* adc #$6 ;bump to alpha a-f
*pq1 sta line24+$26
************************************************************
               txa
               and          #$f
               ora          #$b0
               cmp          #$ba
               bcc          pq                      ;branch if not >9
               adc          #$6                     ;bump to alpha a-f
pq             sta          line24+$27
halt           jmp          halt                    ;hold forever.
*

; ###########################################################################################
; #   END OF FILE:  XDOSMLI
; #   LINES      :  440
; #   CHARACTERS :  21459
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.XRW1.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  XRW1
; ###############################################################################################

                sbtl         'xdos read/write routines revised' '
* org $d400 ; ** moved to xrw file **
***************************
*                         *
*     critical timing     *
*   requires page bound   *
*   considerations for    *
*      code and data      *
*     code-----      *
*   virtually the entire  *
*     'write' routine     *
*      must not cross     *
*     page boundaries.    *
*  critical branches in   *
*  the 'write', 'read',   *
*  and 'read adr' subrs   *
*  which must not cross   *
*  page boundaries are    *
*  noted in comments.     *
*                         *
***************************
*                         *
*         equates         *
*                         *
maxcmd          equ          4                          ;commands 0-3 only
wtemp           equ          $3a
midnib1         equ          wtemp+1
midnib2         equ          wtemp+2
lstnib          equ          wtemp+3
slotz           equ          wtemp+4
yend            equ          wtemp+5
dhpcmd          equ          $42
unitnum         equ          $43
buf             equ          $44
bloknml         equ          $46
bloknmh         equ          bloknml+1
*
dvmot           equ          $e8
*
************************
*                      *
*    device address    *
*     assignments      *
*                      *
************************
phaseoff        equ          $c080                      ;stepper phase off.
*phaseon equ $c081 ;stepper phase on.
q6l             equ          $c08c                      ;q7l,q6l=read
q6h             equ          $c08d                      ;q7l,q6h=sense wprot
q7l             equ          $c08e                      ;q7h,q6l=write
q7h             equ          $c08f                      ;q7h,q6h=write store
***************************************
*
```

```
* equates for rwts and block
*
***************************************
motoroff        equ          $c088
motoron         equ          $c089
drv0en          equ          $c08a
*drv1en equ $c08b
ibderr          equ          $27
ibwper          equ          $2b
ibnodrv         equ          $28
                page
************************
*                      *
*       block i/o      *
*                      *
************************
blockio         cld                                  ;this must be first as it is an id
value
                jsr          rsetphse
                skp          1
* this is patch 76 part 1.  part 2 is in xrw2.
* this is patch 77, use to be 'lda q7l+14,x'.
                skp          1
                lda          q7l,x                   ;turn off write enable please!!!
                skp          1
* (pad is just spare bytes but the number is critical for page
pad             equ          >*                      ;alignment of tables later.)
                ds           $09-pad,$ea             ;fill pad w/nops
                jsr          docheck
                bcs          badblk                  ;branch if block # is out of range
blkio1          ldy          #$5
trksec          asl          a
                rol          ibtrk
                dey
                bne          trksec
                asl          a
                bcc          trksc1
                ora          #$10                    ;adjust for upper 4 blks of trk.
trksc1          lsr          a
                lsr          a
                lsr          a
                lsr          a
                pha                                  ;save sector#
                jsr          regrwts
                pla                                  ;restore sector #
                bcs          quit                    ;branch if error encountered.
                inc          buf+1
                adc          #2
                jsr          regrwts                 ;get second half of block.
                dec          buf+1
quit            lda          ibstat
                rts
*
badblk          lda          #ibderr                 ;i/o error
                sec
                rts
                page
*************************
*                       *
*     read/write a      *
*   track and sector    *
*                       *
*************************
```

```
*
regrwts         ldy          #1                      ;retry count
                sty          seekcnt                 ;only one recalibrate per call
                sta          ibsect
                lda          unitnum                 ;get slot # for this operation
                and          #$70
                sta          slotz
                jsr          chkprev                 ;make sure other drives in other slots
are stopped.
*
* now check if the motor is on, then start it
*
                jsr          chkdrv                  ;set zero flag if motor stopped
                php                                  ;save test results
                lda          #dvmot
                sta          montimeh
                lda          unitnum                 ;determine drive one or two
                cmp          iobpdn                  ;same drive used before?
                sta          iobpdn                  ;save it for next time
                php                                  ;keep results of compare
                asl          a                       ;get drive number into carry
                lda          motoron,x               ;turn on the drive
                bcc          drivsel                 ;branch if drive 1 selected
                inx                                  ;select drive 2
drivsel         lda          drv0en,x
                plp                                  ;was it same drive?
                beq          ok
                plp                                  ;must indicate drive off by setting
zero flag)
                ldy          #7                      ;delay 150 ms before stepping
drvwait         jsr          mswait                  ;(on return a=0)
                dey
                bne          drvwait
                php                                  ;now zero flag set
ok              lda          dhpcmd                  ;make sure this command needs seeking.
                beq          ok1                     ;branch if status check.
                lda          ibtrk                   ;get destination track
                jsr          myseek                  ;and go to it.
*now at the desired track.  was the motor
* on to start with?
ok1             plp                                  ;was motor on?
                bne          trytrk                  ;if so, don't delay, get it today!
*
* motor was off, wait for it to speed up.
*
motof           lda          #1                      ;wait exactly 100 us for each count in
montime
                jsr          mswait
                lda          montimeh
                bmi          motof                   ;count up to 0000
                page
****************************************
*
* motor should be up to speed.
* if it still looks stopped then
* the drive is not present.
*
****************************************
                jsr          chkdrv                  ;is drive present?
                beq          hndlerr                 ;branch if no drive
*
* now check: if it is not the format disk command,
*  locate the correct sector for this operation.
```

```
*
trytrk          lda         dhpcmd                  ;get command code #
                beq         statcmd                 ;if $00, then status command.
                lsr         a                       ;set carry=1 for read, 0 for write
                bcs         trytrk2                 ;must prenibblize for write.
                jsr         prenib16
trytrk2         ldy         #$40                    ;only 64 retries of any kind
                sty         retrycnt
tryadr          ldx         slotz                   ;get slot num into x-reg
                jsr         rdadr16                 ;read next address field
                bcc         rdright                 ;if read it right, hurrah!
tryadr2         equ         *
                dec         retrycnt                ;another mistaek!!
                bpl         tryadr                  ; well, let it go this time.,
                lda         #ibderr                 ;anticipate a bad drive error
                dec         seekcnt                 ;only recalibrate once!
                bne         hndlerr                 ;tried to recalibrate a second time,
error!
                lda         curtrk
                pha                                 ;save track we really want
                asl         a
                adc         #$10                    ;pretend track is 8>curtrk
                ldy         #$40
                sty         retrycnt                ;reset retrys to 64 max.
                bne         recal1                  ;branch always
*
* have now read an address field correctly.
* make sure this is the track, sector, and volume desired.
rdright         ldy         track                   ;on the right track?
                cpy         curtrk
                beq         rttrk                   ;if so, good
* recalibrating from this track
                lda         curtrk                  ;preserve destination track
                pha
                tya
                asl         a                       ;(washing machine fix!)
recal1          jsr         settrk
                pla
                jsr         myseek
                bcc         tryadr                  ;go ahead and recalibrate
                page
*
* drive is on right track, check volume mismatch
*
rttrk           lda         sect                    ;check if this is the right sector
                cmp         ibsect
                bne         tryadr2                 ;no, try another sector
                lda         dhpcmd                  ;read or write?
                lsr         a                       ;the carry will tell.
                bcc         writ                    ;carry was set for read operation,
                jsr         read16                  ;cleared for write
                bcs         tryadr2                 ;carry set upon return if bad read
*
alldone         equ         *                       ;was clc
                lda         #$0                     ;no error
                dfb         $d0                     ;branch never (skip 1 byte)
hndlerr         sec                                 ;indicate an error
aldone1         sta         ibstat                  ;give him error#
                ldx         slotz                   ;get the slot offset
                lda         motoroff,x              ;turn it off...
                rts                                 ;all finished!
*
writ            jsr         write16                 ;write nybbles now
```

```
statdne         bcc          alldone                  ;if no errors.
                lda          #ibwper                  ;disk is write protected!!
                bne          hndlerr                  ;branch always
*
statcmd         ldx          slotz
                lda          q6h,x                    ;test for write protected.
                lda          q7l,x
                rol          a                        ;write protect-->carry-->bit 0=1
                lda          q6l,x                    ;keep in read mode...
                jmp          statdne                  ;branch always taken.
                page
*
* this is the 'seek' routine
*  seeks track 'n' in slot #x/$10
* if drivno is negative, on drive 0
* if drivno is positive, on drive 1
*
myseek          asl          a                        ;assume two phase stepper.
                sta          track                    ;save destination track(*2)
                jsr          alloff                   ;turn all phases off to be sure.
                jsr          drvindx                  ;get index to previous track for
current drive
                lda          drv0trk,x
                sta          curtrk                   ;this is where i am
                lda          track                    ;and where i'm going to
                sta          drv0trk,x
                jsr          seek                     ;go there!
alloff          ldy          #3                       ;turn off all phases before returning
nxoff           tya                                   ;(send phase in acc.)
                jsr          clrphase                 ;carry is clear, phases shold be turned
off
                dey
                bpl          nxoff
                lsr          curtrk                   ;divide back down
                clc
                rts                                   ;all off... now it's dark
                page
*************************
*                       *
*  fast seek subroutine  *
*************************
*                       *
*    on entry ----   *
*                       *
*  x-reg holds slotnum   *
*         times $10.    *
*                       *
*  a-reg holds desired   *
*         halftrack.    *
*         (single phase) *
*                       *
*  curtrk holds current  *
*          halftrack.   *
*                       *
*    on exit -----   *
*                       *
*  a-reg uncertain.      *
*  y-reg uncertain.      *
*  x-reg undisturbed.    *
*                       *
*  curtrk and trkn hold  *
*      final halftrack.  *
*                       *
```

```
*  prior holds prior     *
*    halftrack if seek   *
*    was required.       *
*                        *
*  montimel and montimeh *
*    are incremented by  *
*    the number of       *
*    100 usec quantums   *
*    required by seek    *
*    for motor on time   *
*    overlap.            *
*                        *
* --- variables used --- *
*                        *
*  curtrk, trkn, count,  *
*    prior, slottemp     *
*    montimel, montimeh  *
*                        *
**************************
seek            sta         trkn                    ;save target track
                cmp         curtrk                  ;on desired track?
                beq         setphase                ;yes,energize phase and return
                lda         #$0
                sta         trkcnt                  ;halftrack count.
seek2           lda         curtrk                  ;save curtrk for
                sta         prior                   ;delayed turnoff.
                sec
                sbc         trkn                    ;delta-tracks.
                beq         seekend                 ;br if curtrk=destination
                bcs         out                     ;(move out, not in)
                eor         #$ff                    ;calc trks to go.
                inc         curtrk                  ;incr current track (in).
                bcc         mintst                  ;(always taken)
out             adc         #$fe                    ;calc trks to go.
                dec         curtrk                  ;decr current track (out).
mintst          cmp         trkcnt
                bcc         maxtst                  ; and 'trks moved'.
                lda         trkcnt
maxtst          cmp         #$9
                bcs         step2                   ;if trkcnt>$8 leave y alone (y=$8).
                tay                                 ;else set acceleration index in y
                sec
step2           jsr         setphase
                lda         ontable,y               ;for 'ontime'.
                jsr         mswait                  ;(100 usec intervals)
                lda         prior
                clc                                 ;for phaseoff
                jsr         clrphase                ;turn off prior phase
                lda         offtable,y              ; then wait 'offtime'.
                jsr         mswait                  ;(100 usec intervals)
                inc         trkcnt                  ; 'tracks moved' count.
                bne         seek2                   ;(always taken)
seekend         jsr         mswait                  ;settle 25 msec
                clc                                 ;set for phase off
setphase        lda         curtrk                  ;get current track
clrphase        and         #3                      ;mask for 1 of 4 phases
                rol         a                       ;double for phaseon/off index
                ora         slotz
                tax
                lda         phaseoff,x              ;turn on/off one phase
                ldx         slotz                   ;restore x-reg
                rts                                 ;and return
                page
```

```
*
************************
*                      *
*     7-bit to 6-bit     *
*    'deniblize' tabl    *
*   (16-sector format)   *
*                        *
*      valid codes       *
*    $96 to $ff only.    *
*                        *
*  codes with more than  *
*  one pair of adjacent  *
*   zeroes or with no    *
*  adjacent ones (except *
*   bit 7) are excluded. *
*                        *
*  nibls in the ranges   *
*  of $a0-$a3, $c0-$c7,  *
*  $e0-$e3 are used for  *
*  other tables since no *
*  valid nibls are in    *
*  these ranges.         *
************************
*
dnibl           equ         *-$96
                dfb         $00,$04
                dfb         $ff,$ff,$08,$0c
                dfb         $ff,$10,$14,$18
twobit3         dfb         $00,$80,$40,$c0         ;used in fast prenib as lookup for 2-
bit quantities.
                dfb         $ff,$ff,$1c,$20
                dfb         $ff,$ff,$ff,$24
                dfb         $28,$2c,$30,$34
                dfb         $ff,$ff,$38,$3c
                dfb         $40,$44,$48,$4c
                dfb         $ff,$50,$54,$58
                dfb         $5c,$60,$64,$68
twobit2         dfb         $00,$20,$10,$30         ;used in fast prenib.
endmrks         dfb         $de,$aa,$eb,$ff         ;table using 'unused' nibls
($c4,$c5,$c6,$c7)
                dfb         $ff,$ff,$ff,$6c
                dfb         $ff,$70,$74,$78
                dfb         $ff,$ff,$ff,$7c
                dfb         $ff,$ff,$80,$84
                dfb         $ff,$88,$8c,$90
                dfb         $94,$98,$9c,$a0
twobit1         dfb         $00,$08,$04,$0c         ;used in fast prenib.
                dfb         $ff,$a4,$a8,$ac
                dfb         $ff,$b0,$b4,$b8
                dfb         $bc,$c0,$c4,$c8
                dfb         $ff,$ff,$cc,$d0
                dfb         $d4,$d8,$dc,$e0
                dfb         $ff,$e4,$e8,$ec
                dfb         $f0,$f4,$f8,$fc
*
                page
;   page align the following tables.
************************
*                      *
*     6-bit to 2-bit     *
*   conversion tables.   *
*                      *
* dnibl2 abcdef-->0000fe  *
```

```
* dnibl3 abcdef-->0000dc  *
* dnibl4 abcdef-->0000ba  *
*                         *
***************************
*                         *
*     6-bit to 7-bit      *
*  nibl conversion table  *
*                         *
*   codes with more than  *
*   one pair of adjacent  *
*    zeroes or with no    *
*   adjacent ones (except *
*     b7) are excluded.   *
*                         *
***************************
dnibl2          dfb          0
dnibl3          dfb          0
dnibl4          dfb          0
nibl            dfb          $96,2,0,0,$97
                dfb          1,0,0,$9a,3,0,0,$9b
                dfb          0,2,0,$9d,2,2,0,$9e
                dfb          1,2,0,$9f,3,2,0,$a6
                dfb          0,1,0,$a7,2,1,0,$ab
                dfb          1,1,0,$ac,3,1,0,$ad
                dfb          0,3,0,$ae,2,3,0,$af
                dfb          1,3,0,$b2,3,3,0,$b3
                dfb          0,0,2,$b4,2,0,2,$b5
                dfb          1,0,2,$b6,3,0,2,$b7
                dfb          0,2,2,$b9,2,2,2,$ba
                dfb          1,2,2,$bb,3,2,2,$bc
                dfb          0,1,2,$bd,2,1,2,$be
                dfb          1,1,2,$bf,3,1,2,$cb
                dfb          0,3,2,$cd,2,3,2,$ce
                dfb          1,3,2,$cf,3,3,2,$d3
                dfb          0,0,1,$d6,2,0,1,$d7
                dfb          1,0,1,$d9,3,0,1,$da
                dfb          0,2,1,$db,2,2,1,$dc
                dfb          1,2,1,$dd,3,2,1,$de
                dfb          0,1,1,$df,2,1,1,$e5
                dfb          1,1,1,$e6,3,1,1,$e7
                dfb          0,3,1,$e9,2,3,1,$ea
                dfb          1,3,1,$eb,3,3,1,$ec
                dfb          0,0,3,$ed,2,0,3,$ee
                dfb          1,0,3,$ef,3,0,3,$f2
                dfb          0,2,3,$f3,2,2,3,$f4
                dfb          1,2,3,$f5,3,2,3,$f6
                dfb          0,1,3,$f7,2,1,3,$f9
                dfb          1,1,3,$fa,3,1,3,$fb
                dfb          0,3,3,$fc,2,3,3,$fd
                dfb          1,3,3,$fe,3,3,3,$ff
                page
*
nbuf2           ds           $56,0                    ;nibl buffer for read/write of low 2-
bits of each byte.
*
*
ibtrk           dfb          $00
ibsect          dfb          $00
ibstat          dfb          $00
iobpdn          dfb          $00
curtrk          dfb          $00
drv0trk         equ          *-2
                dfb          0,0,0,0,0,0,0            ;for slots 1 thru 7
```

```
                dfb         0,0,0,0,0,0,0           ;drives 1 & 2
retrycnt        ds          1,0
seekcnt         ds          1,0
*
************************
*                      *
*    readadr----   *
*                      *
************************
count           equ         *                       ;'must find' count.
last            ds          1,0                     ;'odd bit' nibls.
csum            ds          1,0                     ;used for address header cksum
csstv           ds          4,0                     ;four bytes,
*       checksum, sector, track, and volume.
sect            equ         csstv+1
track           equ         csstv+2
volume          equ         csstv+3
*
trkcnt          equ         count                   ;halftrks moved count.
prior           ds          1,0
trkn            ds          1,0
*
************************
*                      *
*    mswait ----   *
*                      *
************************
montimel        equ         csstv+2                 ;motor-on time
montimeh        equ         montimel+1              ;counters.
*
                page
*************************
*                       *
*  phase on-, off-time  *
*   tables in 100-usec  *
*   intervals. (seek)   *
*                       *
*************************
ontable         dfb         1,$30,$28
                dfb         $24,$20,$1e
                dfb         $1d,$1c,$1c
offtable        dfb         $70,$2c,$26
                dfb         $22,$1f,$1e
                dfb         $1d,$1c,$1c
*
*************************
*                       *
*   mswait subroutine   *
*                       *
*************************
*                       *
*  delays a specified   *
*   number of 100 usec  *
*   intervals for motor *
*   on timing.          *
*                       *
*    on entry ----   *
*                       *
*  a-reg: holds number  *
*        of 100 usec    *
*        intervals to   *
*        delay.         *
*                       *
```

```
*    on exit -----   *
*                    *
*  a-reg: holds $00.     *
*  x-reg: holds $00.     *
*  y-reg: unchanged.     *
*  carry: set.           *
*                    *
*  montimel, montimeh    *
*   are incremented once *
*   per 100 usec interval*
*   for moton on timing. *
*                    *
*    assumes ----    *
*                    *
*   1 usec cycle time    *
*                    *
************************
mswait          ldx          #$11
msw1            dex          delay                    86 usec.
                bne          msw1
                inc          montimel
                bne          msw2                     double-byte
                inc          montimeh                 increment.
msw2            sec
                sbc          #$1                      done 'n' intervals?
                bne          mswait                   (a-reg counts)
                rts

; ###########################################################################################
; #   END OF FILE:  XRW1
; #   LINES      :  574
; #   CHARACTERS :  24221
; ###########################################################################################
```

```
================================================================================
DOCUMENT MLI.SRC.XRW2.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  XRW2
; ###############################################################################################

                page
****************************
*                          *
*    read address field    *
*        subroutine        *
*    (16-sector format)    *
*                          *
****************************
*                          *
*    reads volume, track   *
*        and sector        *
*                          *
*    on entry ----     *
*                          *
*  xreg: slotnum times $10 *
*                          *
*  read mode (q6l, q7l)    *
*                          *
*    on exit -----     *
*                          *
*  carry set if error.     *
*                          *
*  if no error:            *
*    a-reg holds $aa.      *
*    y-reg holds $00.      *
*    x-reg unchanged.      *
*    carry clear.          *
*                          *
*    csstv holds chksum,   *
*      sector, track, and  *
*      volume read.        *
*                          *
*    uses temps count,     *
*      last, csum, and     *
*      4 bytes at csstv.   *
*                          *
*     expects ----     *
*                          *
*   original 10-sector     *
*  normal density nibls    *
*   (4-bit), odd bits,     *
*   then even.             *
*                          *
*     caution ----     *
*                          *
*         observe          *
*    'no page cross'       *
*      warnings on         *
*    some branches!!       *
*                          *
*     assumes ----     *
*                          *
*    1 usec cycle time     *
```

```
*                          *
****************************
rdadr16         ldy          #$fc
                sty          count                  ;'must find' count.
rdasyn          iny
                bne          rda1                   ;low order of count.
                inc          count                  ;(2k nibls to find
                beq          rderr                  ; adr mark, else err)
rda1            lda          q6l,x                  ;read nibl.
                bpl          rda1                   ;*** no page cross! ***
rdasn1          cmp          #$d5                   ;adr mark 1?
                bne          rdasyn                 ;(loop if not)
                nop                                 ;added nibl delay.
rda2            lda          q6l,x
                bpl          rda2                   ;*** no page cross! ***
                cmp          #$aa                   ;adr mark 2?
                bne          rdasn1                 ; (if not, is it am1?)
                ldy          #$3                    ;index for 4-byte read.
*            (added nibl delay)
rda3            lda          q6l,x
                bpl          rda3                   ;*** no page cross! ***
                cmp          #$96                   ;adr mark 3?
                bne          rdasn1                 ; (if not, is it am1?)
                sei                                 ;no interupts until address is
tested.(carry is set)
                lda          #$0                    ;init checksum.
rdafld          sta          csum
rda4            lda          q6l,x                  ;read 'odd bit' nibl.
                bpl          rda4                   ;*** no page cross! ***
                rol          a                      ;align odd bits, '1' into lsb.
                sta          last                   ; (save them)
rda5            lda          q6l,x                  ;read 'even bit' nibl.
                bpl          rda5                   ;*** no page cross! ***
                and          last                   ;merge odd and even bits.
                sta          csstv,y                ;store data byte.
                eor          csum
                dey
                bpl          rdafld                 ;loop on 4 data bytes.
                tay                                 ;if final checksum
                bne          rderr                  ;nonzero, then error.
rda6            lda          q6l,x                  ;first bit-slip nibl.
                bpl          rda6                   ;*** no page cross! ***
                cmp          #$de
                bne          rderr                  ;error if nonmatch.
                nop                                 ;delay
rda7            lda          q6l,x                  ;second bit-slip nibl.
                bpl          rda7                   ;*** no page cross! ***
                cmp          #$aa
                bne          rderr                  ;error if nonmatch.
                clc                                 ;clear carry on
                rts                                 ; normal read exits.
rderr           sec
                rts
                page
**************************
*                        *
*     read subroutine    *
*   (16-sector format)   *
*                        *
**************************
*                        *
*   reads encoded bytes  *
*  into nbuf1 and nbuf2  *
```

```
*                        *
*  first reads nbuf2     *
*          high to low,  *
*  then reads nbuf1      *
*          low to high.  *
*                        *
*    on entry ----   *
*                        *
*  x-reg: slotnum        *
*         times $10.     *
*                        *
*  read mode (q6l, q7l)  *
*                        *
*    on exit -----   *
*                        *
*  carry set if error.   *
*                        *
*  if no error:          *
*     a-reg holds $aa.   *
*     x-reg unchanged.   *
*     y-reg holds $00.   *
*     carry clear.       *
*    caution -----   *
*                        *
*        observe         *
*    'no page cross'     *
*      warnings on       *
*    some branches!!     *
*                        *
*    assumes ----    *
*                        *
*   1 usec cycle time    *
*                        *
**************************
read16          txa                             ;get slot #.
                ora         #$8c                ;prepare mods to read routine.
                sta         rd4+1               ;warning: the read routine is self
modified!!!
                sta         rd5+1
                sta         rd6+1
                sta         rd7+1
                sta         rd8+1
                lda         buf                 ;modify storage addresses also.
                ldy         buf+1
                sta         ref3+1
                sty         ref3+2
                sec
                sbc         #$54
                bcs         rd16b               ;branch if no borrow.
                dey
rd16b           sta         ref2+1
                sty         ref2+2
                sec
                sbc         #$57
                bcs         rd16c               ;branch if no borrow.
                dey
rd16c           sta         ref1+1
                sty         ref1+2
                ldy         #$20                ;'must find count'
rsync           dey
                beq         rderr2              ;branch if can't find data header
marks.
rd1             lda         q6l,x
```

```
                bpl         rd1
rsync1          eor         #$d5                ;first data mark.
                bne         rsync
                nop                             ;waste a little time...
rd2             lda         q6l,x
                bpl         rd2
                cmp         #$aa                ;data mark 2
                bne         rsync1              ;if not, check for first again.
                nop
rd3             lda         q6l,x
                bpl         rd3
                cmp         #$ad                ;data mark 3
                bne         rsync1              ;if not, check for data mark 1 again.
                ldy         #$aa
                lda         #0
rdata1          sta         wtemp               ;use zpage for checksum keeping.
rd4             ldx         $c0ec               ;warning: self modified.
                bpl         rd4
                lda         dnibl,x
                sta         nbuf2-$aa,y         ;save the two-bit groups in nbuf.
                eor         wtemp               ;update checksum.
                iny                             ;bump to next nbuf position.
                bne         rdata1              ;loop for all $56 two-bit groups.
                ldy         #$aa                ;now read directly into user buffer.
                bne         rd5                 ;branch always taken!!!
*
rderr2          sec
                rts
*
ref1            sta         $1000,y             ;warning: self modified!
*
rd5             ldx         $c0ec
                bpl         rd5
                eor         dnibl,x             ;get actual 6-bit data from dnib table.
                ldx         nbuf2-$aa,y         ;get associated two-bit pattern.
                eor         dnibl2,x            ;and combine to form whole byte.
                iny
                bne         ref1                ;loop for $56 bytes.
                pha                             ;save this byte for now, no time to
store...
                and         #$fc                ;strip low bits...
                ldy         #$aa                ;prepare for next $56 bytes.
rd6             ldx         $c0ec
                bpl         rd6
                eor         dnibl,x
                ldx         nbuf2-$aa,y
                eor         dnibl3,x
ref2            sta         $1000,y             ;warning: self modified.
                iny
                bne         rd6                 ;loop until this group of $56 read in.
*
rd7             ldx         $c0ec
                bpl         rd7
                and         #$fc
                ldy         #$ac                ;last group is $54 long.
rdata2          eor         dnibl,x
                ldx         nbuf2-$ac,y
                eor         dnibl4,x            ;combine to form full byte.
ref3            sta         $1000,y
rd8             ldx         $c0ec               ;warning: self modified.
                bpl         rd8
                iny
                bne         rdata2
```

```
                and         #$fc
                eor         dnibl,x                 ;check sum ok?
                bne         rderr1                  ;branch if not.
                ldx         slotz                   ;test end marks.
rd9             lda         q6l,x
                bpl         rd9
                cmp         #$de
                clc
                beq         rdok                    ;branch if good trailer...
rderr1          sec
rdok            pla                                 ;place last byte into user buffer.
                ldy         #$55
                sta         (buf),y
                rts
*
                page
*
* this subroutine sets the slot dependent track
* location.
*
settrk          jsr         drvindx                 ;get index to drive number.
                sta         drv0trk,x
                rts
****************************************
*
* subr to tell if motor is stopped
*
* if motor is stopped, controller's
* shift reg will not be changing.
*
* return y=0 and zero flag set if it is stopped.
*
****************************************
chkdrv          ldx         slotz
chkdrv0         ldy         #0                      ;init loop counter
chkdrv1         lda         q6l,x                   ;read the shift reg
                jsr         ckdrts                  ;delay
                pha
                pla                                 ;more delay
                cmp         q6l,x                   ;has shift reg changed?
                bne         ckdrts                  ;yes, motor is moving
                lda         #ibnodrv                ;anticipate error.
                dey                                 ;no,dec retry counter
                bne         chkdrv1                 ;and try 256 times
ckdrts          rts                                 ;then return
*
drvindx         pha                                 ;preserve acc.
                lda         unitnum
                lsr         a
                lsr         a
                lsr         a
                lsr         a
                cmp         #$8
                and         #$7
                rol         a
                tax                                 ;into x for index to table
                pla                                 ;restore acc.
                rts
                page
************************
*                      *
*      write subr      *
*  (16-sector format)  *
```

```
*                       *
************************
*                       *
*   writes data from    *
*    nbuf1 and buf      *
*                       *
*  first nbuf2,         *
*      high to low.     *
*  then direct from     *
*  (buf), low to high.  *
* self modified code!!  *
*   on entry ----  *
*                       *
*   x-reg: slotnum      *
*        times $10.     *
*                       *
*                       *
*   on exit -----  *
*                       *
*  carry set if error.  *
*   (w prot violation)  *
*                       *
*  if no error:         *
*                       *
*    a-reg uncertain.   *
*    x-reg unchanged.   *
*    y-reg holds $00.   *
*    carry clear.       *
*                       *
*   assumes ----   *
*                       *
*  1 usec cycle time    *
*                       *
************************
write16         sec                             ;anticipate wprot err.
                lda     q6h,x
                lda     q7l,x                   ;sense wprot flag.
                bpl     wr16
                jmp     wexit                   ;exit if write protected
wr16            lda     nbuf2
                sta     wtemp
                lda     #$ff                    ;sync data.
                sta     q7h,x                   ;(5) goto write mode
                ora     q6l,x                   ;(4)
                ldy     #$4                     ;(2) for five nibls.
                nop                             ;(2)
                pha                             ;(3)
                pla                             ;(4)
wsync           pha                             ;(3) exact timing.
                pla                             ;(4) exact timing.
                jsr     wnibl7                  ;(13,9,6)  write sync.
                dey                             ;(2)
                bne     wsync                   ;(3-)  must not cross page!
                lda     #$d5                    ;(2)  1st data mark.
                jsr     wnibl9                  ;(15,9,6)
                lda     #$aa                    ;(2)  2nd data mark.
                jsr     wnibl9                  ;(15,9,6)
                lda     #$ad                    ;(2)  3rd data mark.
                jsr     wnibl9                  ;(15,9,6)
                tya                             ;(2) zero checksum
                ldy     #$56                    ;(2) nbuf2 index
                bne     wdata1                  ;(3)branch always taken.
wdata0          lda     nbuf2,y                 ;(4) prior 6-bit nibl.
```

```
wdata1          eor          nbuf2-1,y              ;(5) xor with current.
                tax                                 ;(2) index to 7-bit nibl (nbuf2 must be
on page boundary).
                lda          nibl,x                 ;(4) must not cross page boundary
                ldx          slotz                  ;(3) restore slot index.
                sta          q6h,x                  ;(5) store encoded byte.
                lda          q6l,x                  ;(4) time must = 32 us per byte!
                dey                                 ;(2)
                bne          wdata0                 ;(3-) must not cross page boundary
                lda          wtemp                  ;(3) get prior nibl (from nbuf2).
wrefd1          ldy          #0                     ;(2) warning: load value modified by
prenib!
wdata2          equ          *
wrefa1          eor          $1000,y                ;(4) warning: address modified by
prenib!
                and          #$fc                   ;(2)
                tax                                 ;(2) index to nibl table
                lda          nibl,x                 ;(4)
wrefd2          ldx          #$60                   ;(2) warning: load value modified by
prenib
                sta          q6h,x                  ;(5) write nibl.
                lda          q6l,x                  ;(4) handshake.
wrefa2          lda          $1000,y                ;(4) prior nibl. warning: address
modified by prenib.
                iny                                 ;(2) all done with this page?
                bne          wdata2                 ;(3-) loop until page end.
*
                lda          midnib1                ;(3) get next (precalculated and
translated) nibl.
                beq          wdone                  ;(2+) branch if code writen was page
aligned.
                lda          yend                   ;(3) get byte address of last byte to
be written.
                beq          wdata4                 ;(2+) branch if only 1 byte left to
write.
                lsr          a                      ;(2) test for odd or even last byte
(carry set or clear)
                lda          midnib1                ;(3) restore nibl to acc.
                sta          q6h,x                  ;(5)
                lda          q6l,x                  ;(4)
                lda          midnib2                ;(3) =byte 0 of second page. xor'd with
byte 1 if above test set carry.
                nop                                 ;(2) waste time.
                iny                                 ;(2) y=1
                bcs          wrtodd                 ;(2+) branch if last byte to be odd.
*
wdata3          equ          *
wrefa3          eor          $1100,y                ;(4) warning: address modified by
prenib.
                and          #$fc                   ;(2) strip low 2 bits.
                tax                                 ;(2) index to nibl table.
                lda          nibl,x                 ;(4) get nibl.
wrefd3          ldx          #$60                   ;(2) restore slot index. warning:
modified by prenib.
                sta          q6h,x                  ;(5)
                lda          q6l,x                  ;(4)
wrefa4          lda          $1100,y                ;(4) warning: modified by prenib.
                iny                                 ;(2) got prior nibl, bump to next.
wrefa5          eor          $1100,y                ;(4) warning: modified by prenib.
wrtodd          cpy          yend                   ;(3) set carry if this is last nibl.
                and          #$fc                   ;(2)
                tax                                 ;(2)
                lda          nibl,x                 ;(4)
```

```
wrefd4          ldx          #$60                    ;(2) restore slot. warning: modified by
prenib.
                sta          q6h,x                   ;(5)
                lda          q6l,x                   ;(4)
wrefa6          lda          $1100,y                 ;(4) get prior. warning: these warnings
are all the same.
                iny                                  ;(2)
                bcc          wdata3                  ;(3-) branch if that was not the last.
                bcs          wdne1                   ;(3) waste 3 cycles, branch always
wdne1           bcs          wdone                   ;(3) branch always
*
wdata4          dfb          $ad,midnib1,$00         ;(4) absolute reference to zero page.
                sta          q6h,x                   ;(5)
                lda          q6l,x                   ;(4)
                pha                                  ;(3) waste 14 us total.
                pla                                  ;(4)
                pha                                  ;(3)
                pla                                  ;(4)
wdone           ldx          lstnib                  ;(3) use last nibl (anded with $fc) for
checksum.
                lda          nibl,x                  ;(4)
wrefd5          ldx          #$60                    ;(2) restore slot. warning: see above
warnings...
                sta          q6h,x                   ;(5)
                lda          q6l,x                   ;(4)
                ldy          #0                      ;(2) set y to index end mark table.
                pha                                  ;(3) waste another 11 us.
                pla                                  ;(4)
                nop                                  ;(2)
                nop                                  ;(2)
wrtendmk        lda          endmrks,y               ;(4) dm4, dm5, dm6, and turn off byte.
                jsr          wnibl                   ;(15,6)    write it.
                iny                                  ;(2)
                cpy          #4                      ;(2) have all end marks been written?
                bne          wrtendmk                ;(3)
                clc                                  ;(2,9)
wexit           lda          q7l,x                   ;out of write mode.
                lda          q6l,x                   ;to read mode.
                rts                                  ;return from write.
*
****************************
*                          *
*   7-bit nibl write subrs  *
*                          *
*   a-reg or'd prior exit   *
*       carry cleared       *
*                          *
****************************
wnibl9          clc                                  ;(2)  9 cycles, then write.
wnibl7          pha                                  ;(3)  7 cycles, then write.
                pla                                  ;(4)
wnibl           sta          q6h,x                   ;(5)  nibl write sub.
                ora          q6l,x                   ;(4)  clobbers acc, not carry.
                rts                                  ;(6)
*
                page
****************************
*                          *
*    preniblize subr       *
*   (16-sector format)     *
*                          *
****************************
*                          *
```

```
*  converts 256 bytes of   *
*  user data in (buf) into *
*  6 bit nibls into nbuf2  *
*  high 6 bits are trans-  *
*  lated directly by the   *
*  write routines.         *
*                          *
*     on entry ----    *
*                          *
*  buf is 2-byte pointer   *
*    to 256 bytes of user  *
*    data.                 *
*                          *
*     on exit -----    *
*                          *
*  a,x,y undefined.        *
*  write routine modified  *
*  to do direct conversion *
*  of high 6 bits of users *
*  buffer data.            *
*************************
prenib16        lda         buf                 ;first self modify addresses so we can
be fast!
                ldy         buf+1               ;y contains high order address.
                clc                             ;all offsets are -$aa...
                adc         #2                  ;the highest set is buf+$ac
                bcc         prenib1             ;branch if no carry.
                iny                             ;otherwise add carry to high address.
prenib1         sta         prn3+1              ;self mod 3
                sty         prn3+2
                sec
                sbc         #$56                ;middle set is buf+$56
                bcs         prenib2             ;branch if no borrow
                dey                             ;otherwise deduct from high..
prenib2         sta         prn2+1              ;self mod 2.
                sty         prn2+2
                sec
                sbc         #$56                ;low set is exactly buf.
                bcs         prenib3
                dey
prenib3         sta         prn1+1              ;self mod 1.
                sty         prn1+2
*
                ldy         #$aa                ;count up to 0.
prenib4         equ         *
prn1            lda         $1000,y             ;fetch byte from lowest group. warning:
self modified.
                and         #$3                 ;strip high 6 bits.
                tax                             ;index to 2 bit equiv.
                lda         twobit1,x
                pha                             ;save pattern
prn2            lda         $1056,y             ;fetch from middle group
                and         #3
                tax
                pla                             ;restore pattern.
                ora         twobit2,x           ;combine second group with first.
                pha                             ;save new pattern.
prn3            lda         $10ac,y             ;get highest group.
                and         #3
                tax
                pla                             ;restore new pattern.
                ora         twobit3,x           ;and form final nibl.
                pha
```

```
                tya
                eor         #$ff
                tax
                pla
                sta         nbuf2,x                 ;save in nibl buffer!
                iny                                 ;bump to next set.
                bne         prenib4                 ;loop until all $56 nibls formed.
                ldy         buf                     ;now prepare data bytes for write16
routine.
                dey                                 ;prepare end addr.
                sty         yend
                lda         buf
                sta         wrefd1+1                ;warning: the following storage
addresses starting
                beq         wmod1                   ; with 'wref' are referces into code
space,
                eor         #$ff                    ; changed by this routine.
                tay                                 ;index to last byte of page pointed to
by buf.
                lda         (buf),y                 ;pre-niblize the last byte of the page
with
                iny                                 ; the first byte of the next page.
                eor         (buf),y
                and         #$fc
                tax
                lda         nibl,x                  ;get disk 7-bit nibl equivalent.
wmod1           sta         midnib1
                beq         wmod3                   ;branch if data to be written is page
aligned.
                lda         yend                    ;find out if last byte is even or odd
address.
                lsr         a                       ;shift even/oddness into carry
                lda         (buf),y                 ;if even, then leave in tact.
                bcc         wmod2                   ;branch if odd.
                iny                                 ;if even, then pre-xor with byte 1.
                eor         (buf),y
wmod2           sta         midnib2                 ;save result for write routine.
wmod3           ldy         #$ff                    ;index to last byte of data to be
writen
                lda         (buf),y                 ; to be used as checksum.
                and         #$fc                    ;strip extra bits
                sta         lstnib                  ;save it.
                ldy         buf+1                   ;now modify address reference to user
data.
                sty         wrefa1+2
                sty         wrefa2+2
                iny
                sty         wrefa3+2
                sty         wrefa4+2
                sty         wrefa5+2
                sty         wrefa6+2
                ldx         slotz                   ;and lastly index references to
controller.
                stx         wrefd2+1
                stx         wrefd3+1
                stx         wrefd4+1
                stx         wrefd5+1
                rts                                 ;all done.
*
chkprev         eor         iobpdn                  ;same slot as last?
                asl         a
                beq         chkpv2
                lda         #01
```

```
                sta         montimeh
chkpv1          lda         iobpdn
                and         #$70
                tax
                beq         chkpv2                  ;branch if no previous ever (boot only)
                jsr         chkdrv0                 ;find out if previous drive running.
                beq         chkpv2                  ;branch if stopped.
                lda         #1                      ;waste some time.
                jsr         mswait
                lda         montimeh
                bne         chkpv1
chkpv2          rts
*
* ----------------- see rev notes 14, 18 & 70 -------------
rsetphse        equ         *
                lda         unitnum                 ;get unit number
                and         #$7f                    ;map off hi bit
                tax
*
* clear all the phases and force read mode
*
*
* patch 76 part 2. part 1 is in xrw1.
                skp         1
                lda         phaseoff+0,x            ;make sure all motor phases
                lda         phaseoff+2,x            ;  are off.
                lda         phaseoff+4,x
                lda         phaseoff+6,x
                rts
* ---------------------------------------------------
docheck         equ         *
                lda         dhpcmd                  ;get the command number
                cmp         #maxcmd                 ;is the command allowable?
                bcs         dochkbad                ;branch if not!
                lda         bloknml
                ldx         bloknmh
                stx         ibtrk                   ;calculate block's track & sector.
                beq         dochkok                 ;branch if block # in range.
                dex                                 ;else test further
                bne         dochkbad                ;bad range
                cmp         #$18                    ;must be <$118
                bcc         dochkok
dochkbad        sec                                 ;set carry for an error
                rts
*
dochkok         clc                                 ;no error
                rts

; ###############################################################################################
; #   END OF FILE:  XRW2
; #   LINES      :  634
; #   CHARACTERS :  30925
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT PRODOS.EXEC.LST.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  PRODOS.EXEC.LST
; ###############################################################################################

NEW
-/prodos16/asm816/asm816,L.MLI,MLI.0,L=77/prodos16/mli/mli.list
-/prodos16/asm816/asm816,L.XRW,XRW.0,L=77/prodos16/mli/xrw.list
-/prodos16/asm816/asm816,L.TCLOCK,TCLOCK.0,L=77/prodos16/mli/tclock.list
-/prodos16/asm816/asm816,L.CCLOCK,CCLOCK.0,L=77/prodos16/mli/cclock.list
-/prodos16/asm816/asm816,L.RAM,RAM.0,L=77/prodos16/mli/ram.list
-/prodos16/asm816/asm816,L.SEL,SEL.0,L=77/prodos16/mli/sel.list

; ###############################################################################################
; #   END OF FILE:  PRODOS.EXEC.LST
; #   LINES      :  7
; #   CHARACTERS :  399
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT PRODOS.EXEC.pretty
================================================================================

NEW
-/prodos16/asm816/asm816,MLI
-/prodos16/asm816/asm816,XRW
-/prodos16/asm816/asm816,TCLOCK
-/prodos16/asm816/asm816,CCLOCK
-/prodos16/asm816/asm816,RAM
-/prodos16/asm816/asm816,SEL
```

```
================================================================================
DOCUMENT RAM.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  RAM
; ###############################################################################################

                LST          OFF,NOASYM,NOVSYM,NOGEN
                X65816
                XREFSLOT     6
                SEG          $00
*
************************************************************
*                                                          *
*            PRODOS 8 KERNEL 59.5K RAMDISK (REV-E)         *
*                                                          *
*            COPYRIGHT APPLE COMPUTER, INC., 1983-86       *
*                                                          *
*                     ALL RIGHTS RESERVED                  *
*                                                          *
************************************************************
*
**** Note that the "os" variable must be updated depending on
**** the system you are building.
*
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
* Creates 3 object files
                SBTL         'EXTENDED 80 COL RAMDISK'
                DSECT
                ORG          $42
CMD             DS           1,0                     ;command from ProDOS
UNIT            DS           1,0                     ;unitnum from ProDOS
BUFPTR          DS           2,0                     ;User Buffer pointer
BLOCK           DS           2,0                     ;block requested
                DEND
*
XDIOERR         EQU          $27
XDWPERR         EQU          $2B                     ;write protect error
A1L             EQU          $3C
A1H             EQU          $3D                     ;SOURCE OF TRANSFER
A2L             EQU          $3E
A2H             EQU          $3F                     ;END OF SOURCE
A3L             EQU          $40
A3H             EQU          $41
A4L             EQU          $42
A4H             EQU          $43                     ;DESTINATION OF TRANSFER
*
MOVE            EQU          $C311                   ;monitor move data routine
XFER            EQU          $C314                   ;monitor XFER control
*
EnblRAM2        EQU          $C083                   ;enable 2nd bank of LC
EnblRAM1        EQU          $C08B                   ;enable 1st bank of LC
*
Store80Off      EQU          $C000
Store80On       EQU          $C001
Store80Sw       EQU          $C018
WRMAINRAM       EQU          $C004                   ;Write data to main ram
```

```
WRCARDRAM       EQU         $C005                   ;Write data to card ram
ALTZP           EQU         $C009                   ;enable alternate ZP,STK,LC
REGZP           EQU         $C008                   ;enable regular ZP,STK,LC
*
ABUF1           EQU         $C00                    ;temporary buffer
VBLOCK1         EQU         $E00                    ;where the Vdir goes
*
EnterCard       EQU         $200                    ;card entry point
PassIt          EQU         $3ED                    ;XFER param
*
                PAGE
                ORG         EnterCard
*
* After the main routine has determined that the command
* is ok, and the block to be read/written is within
* range, it tranfers control to this routine.  This routine
* remaps the block requested as follows:
*     Request blocks 0,1 :invalid
*                    2 :returns VDIR (card block 3)
*                    3 :returns bitmap (synthesized)
*                    4 :returns card block 0
*                 $5-$5F :returns card blocks $5-$5F
*                $60-$67 :returns blocks $68-$7F in bank 1 of
*                           card's language card
*                $68-$7F :returns blocks $68-$7F in bank 2
*                           of card's language card
*
DoCmd           LDA         Store80Sw               ;Read 80STORE
                PHA                                 ;save for later
                STA         Store80Off              ;turn off 80STORE
*
                LDX         #4                      ;move the params for our use
DoCmd1          LDA         CMD,X                   ;CMD,UNIT,BUFPTR,&BLOCK(lo)
                STA         TCMD,X                  ;->TCMD,TUNIT,R2L,R2,R1
                DEX
                BPL         DoCMD1
                AND         FormatFlag              ;Format the volume first time
                BNE         DoCommand               ;thru, or when requested
*
DoFormat        LDX         BLOCK                   ;save R1 during format
                LDA         #<VBLOCK1               ;block to be cleared
                JSR         CLRBUF1                 ;ClrBuf clears all buffers
                LDY         #$3                     ;Format volume in 2 chunks
DF2             LDA         Vdir,Y
                STA         VBLOCK1+4,Y
                DEY
                BPL         DF2
*
                lda         #$FE                    ;Set last block as unusable to protect
vectors
                sta         bitmap+$F
                TYA                                 ;Set bitmap bits to $FF
                LDY         #$E                     ;15 bytes to set
DF2.1           STA         BITMAP,Y
                DEY
                BNE         DF2.1
                STY         BITMAP                  ;first byte=0
*
                LDY         #$7                     ;do other chunk
DF3             LDA         Access,Y
                STA         VBLOCK1+34,Y
                DEY
                BPL         DF3
```

```
               LDA         FormatFlag              ;if 0, set to FF
               BNE         DFX                     ;else exitcard
               STY         FormatFlag              ;Y=FF, won't format next time
               STX         R1                      ;restore R1
*
* Now use the requested block number to determine
* which routine performs the transfer
*
DoCommand      ASL         R1                      ;block requested->page requested
               LDA         R1                      ;get page requested
               CMP         #$BF                    ;in Language card?
               BCS         TLC1                    ;yes, do it
CB1            CMP         #6                      ;bit map?
               BNE         CB2
               JMP         TBMAP                   ;yes, transfer bitmap
CB2            JMP         TREG                    ;else normal transfer
*
* When a block between $60 and $7F is requested, it must
* be spirited into/from the language card area of the
* 64K card.  This requires a two-stage move: into the temp
* buffer and then to its real destination.
*
TLC1           TAX                                 ;save R1 for later
               JSR         SETPTR                  ;get direction
               PHP                                 ;save direction
               BCS         LCWrt                   ;it is a write
LCRd           TXA                                 ;get R1 back
               CMP         #$CF                    ;which bank is it in
               BCS         TLC2                    ;in main bank
*
               ORA         #$10                    ;in secondary bank
               BNE         TLC3                    ;branch always
*
TLC2           STA         EnblRAM2                ;turn on main $D000
               STA         EnblRAM2
*
TLC3           STA         R1                      ;restore R1
               LDA         R2                      ;save R2 for later
               PHA
               LDX         R2L
               STA         ALTZP                   ;now switch to other ZP
               LDA         #<Abuf1                 ;set R2 to Abuf
               STA         R2
               LDA         #>Abuf1
               STA         R2L
               JSR         SetPtr                  ;set pointers
               TAY                                 ;A=0 from SETPTR
TLC4           LDA         (A1L),Y                 ;move A1,A2 to A4,A3
               STA         (A4L),Y
               LDA         (A2L),Y
               STA         (A3L),Y
               DEY
               BNE         TLC4
*
               STA         REGZP                   ;restore normal ZP
               STX         R2L
               PLA                                 ;restore R2
               STA         R2
               PLP                                 ;get direction
DFX            BCS         XLCWrt                  ;write, done with move
*
               STA         EnblRAM1                ;now switch MLI part of LC in
               STA         EnblRAM1
```

```
                JSR         BLOCKDO0                ;read, transfer Abuf to main
XLCWrt          JMP         ExitCard
*
LCWrt           JSR         BLOCKDO0                ;transfer main to Abuf
                JMP         LCRd                    ;transfer Abuf to Lang card
*
* BLOCKDO transfers a block between main memory and the
*  64K card.  R1 contains the page address of the block
*  in the card; R2 contains the page address of the block
*  in main memory.  The address in main memory is always
*  in the language card, so the language card is always
*  switched in.  If CMD is 2, a write is done (R2->R1);
*  if CMD is 1, a read is done (R1->R2).
*
BLOCKDO0        LDA         #<ABUF1                 ;set up R1 = Abuf
BLOCKDO1        STA         R1
BLOCKDO         JSR         SETPTR                  ;set pointers
                BCS         BlockWrite              ;it's a write
                STA         WrMainRAM               ;transfer buffer directly to main ram
                TAY                                 ;0 left from SETPTR
BD1             LDA         (A1L),Y                 ;transfer A1,A2 to A4,A3
                STA         (A4L),Y
                LDA         (A2L),Y
                STA         (A3L),Y
                DEY
                BNE         BD1
                STA         WrCardRAM               ;back the way it was
DoneWrt         RTS                                 ;MainWrt returns here
*
BlockWrite      LDA         #>MainWrt               ;pointers set up,
                STA         PassIt                  ;pass control to main ram
                LDA         #<MainWrt
                JMP         Ex1                     ;set PassIt+1 and transfer
*
* SETPTR is used by other routines to set up pointers
* and to detect read or write.
*
SETPTR          LDA         TCMD                    ;the rest depends on read
                LSR         A                       ;or write.  Which is it?
                BCS         CMDWRT                  ;it's write
*
CMDRD           LDA         R2                      ;get dest page
                STA         A4H                     ;1st dest page (MOVE)
                STA         A3H                     ;2nd dest page
                LDA         R2L                     ;low byte dest page
                STA         A4L                     ;1st dest page low
                STA         A3L                     ;2nd dest page low
                LDA         R1                      ;get source page
                STA         A1H                     ;1st source page
                STA         A2H                     ;2nd source page
                LDA         #0                      ;source page aligned
                STA         A1L                     ;1st source page
                STA         A2L                     ;2nd source page
                BEQ         CMDBOTH                 ;update second pages
*
CMDWRT          LDA         R2                      ;get source page
                STA         A1H                     ;1st source page
                STA         A2H                     ;2nd source page
                LDA         R2L                     ;get source page low
                STA         A1L                     ;1st source page low
                STA         A2L                     ;2nd source page low
                LDA         R1                      ;get dest page
                STA         A4H                     ;1st dest page
```

```
                STA         A3H                     ;2nd dest page
                LDA         #0                      ;dest page aligned
                STA         A4L                     ;1st dest page
                STA         A3L                     ;2nd dest page
*
CMDBOTH         INC         A2H                     ;update 2nd source page
                INC         A3H                     ;update 2nd dest page
                RTS
*
* TZIP is called if Blocks 0,1,4,5 are requested.
* On write it does nothing, on read, it returns 0's
*
TZIP            JSR         CLRBUF0                 ;fill ABUF with 0's
                JSR         BLOCKDO                 ;transfer them 0's
ZIPOUT          JMP         ExitCard                ;and return
*
* CLRBUF fills the buffer indicated by R1 to 0's
* Should only be called on a read or format.
*
CLRBUF0         LDA         #<ABUF1                 ;ABUF is temp buffer
CLRBUF1         STA         R1                      ;assign to BLOCK
CLRBUF2         JSR         SETPTR                  ;set pointers
                TAY                                 ;A set to 0 by setptr
CLRBUF3         STA         (A1L),Y
                STA         (A2L),Y
                DEY
                BNE         CLRBUF3
                RTS
*
* TREG maps the requested block into the aux card
* so that 8K data files will be contiguous (the index
* blocks will not be placed within data).
*
TREG            CMP         #4                      ;page 4 = vdir
                BNE         T1                      ;not vdir, continue
                LDA         #$7                     ;else xfer block 7
                BNE         GoTimes2
*
***************** See Rev Note #43 ********************
*
T1              CMP         #$f                     ;if any page<f (<block 8) requested
*****************************************************
                BCC         TZIP                    ;it is invalid
*
TREG1           LDX         #0                      ;X contains number of iterations
                LDA         BLOCK                   ;use true block number
                CMP         #$5D                    ;beyond 8K blocks?
                BCC         TR1                     ;no, do normal
                SBC         #$50                    ;else subtract offset
GoTimes2        JMP         Times2                  ;and multiply by 2
*
* Determine which 8K chunk it is in, place in X;
* block offset into chunk goes into Y.
*
TR1             SEC
                SBC         #$8                     ;block = block -6
TR2             CMP         #$11                    ;if <=17, done
                BCC         TR3                     ;yup, got iteration
                SBC         #$11                    ;else block =block -17
                INX                                 ;count iteration
                BPL         TR2                     ;branch always
                DFB         $00                     ;just in case
TR3             TAY                                 ;remainder in Y
```

```
*
* If remainder is 1 it's an index block: start index
* blocks at $1000,$2000..$19FF)
*    If remainder is 0, it is first data block in 8K
* chunk.  Page is 32 + (16 * X).
*    Otherwise, it is some other data block.
* Page is 32 + (16 * X) + (2 * Y)
*
                CPY         #1                      ;is it index block?
                BNE         TR4                     ;no
                TXA                                 ;index = 2*(8+X)
                CLC
                ADC         #8
                BNE         Times2                  ;multiply by 2
TR4             INX                                 ;need iteration+1
                TXA                                 ;page = 2 * (16 + 8X)
                ASL         A
                ASL         A
                ASL         A
                ASL         A
                STA         R1
                TYA                                 ;get offset into 8K chunk
                BEQ         TR5                     ;if 0, no offset
                DEY                                 ;else offset = 2 * Y
                TYA
TR5             CLC
                ADC         R1
Times2          ASL         A                       ;acc=2*acc
                JSR         BLOCKDO1                ;store in R1 and xfer
                JMP         ExitCard                ;and return
*
* When Block 3 is requested, the bitmap is returned.  The
* Real bitmap is only 16 bytes long (BITMAP); the rest of
* the block is synthesized.  The temporary buffer at $800
* is used to build/read a full size bitmap block.
*
TBMAP           LDA         #<ABUF1                 ;use temporary buffer as BLOCK
                STA         R1
                JSR         SETPTR                  ;Set pointers/test read-write
                BCS         BITWRT                  ;its a write!
*
BITRD           JSR         CLRBUF2
*
                LDY         #$F                     ;now put real bitmap there
BITRD2          LDA         BITMAP,Y
                STA         (A1L),Y
                DEY
                BPL         BITRD2
                JSR         BLOCKDO                 ;move temp buf to user buf
                JMP         ExitCard
*
BITWRT          JSR         BLOCKDO                 ;move user buf to temp buf
                JSR         SETPTR                  ;Set pointers
                LDY         #$F                     ;move temp buf to bitmap
BITWRT1         LDA         (A4L),Y                 ;(pointer set by SETPTR)
                STA         BITMAP,Y
                DEY
                BPL         BITWRT1
                JMP         ExitCard
*
FormatFlag      DFB         0                       ;not formatted yet
*
TCMD            DS          1,0                     ;command byte
```

```
TUNIT           DS          1,0                     ;unit byte (not used)
R2L             DS          1,0                     ;low byte of user buffer
R2              DS          1,0                     ;hi byte of user buffer
R1              DS          1,0                     ;page requested
*
BITMAP          DFB         $00,$FF,$FF,$FF         ;blocks 0-7 used
                DFB         $FF,$FF,$FF,$FF
                DFB         $FF,$FF,$FF,$FF
                DFB         $FF,$FF,$FF,$FE
*
Vdir            EQU         *                       ;start of vdir
Type_NameL      DFB         $F3                     ;storage type F, namelength 3
                MSB         OFF
VName           ASC         "RAM"
*
Access          DFB         $C3                     ;Destroy, Rename, Read enabled
Entry_Len       DFB         $27                     ;entry length
Entries_Per_Block DFB         $0D
File_Count      DFB         0,0
Map_Pointer     DFB         3,0                     ;Block 3
Total_Blocks    DFB         $7f                     ;128 blocks
*
ExitCard        LDA         EnblRAM1                ;restore lang card
                LDA         EnblRAM1
                PLA                                 ;get 80STORE
                BPL         Ex0                     ;80STORE wasn't on
                STA         Store80On
Ex0             JMP         $3EF                    ;Jump around PassIt (3ED,3EE)
                DS          $3EF-*,0                ;pad thru $3EE
                LDA         #>NoErr                 ;set up return to NoErr
                STA         PassIt
                LDA         #<NoErr
Ex1             STA         PassIt+1                ;also used by BlockWrite
                CLC                                 ;transfer card to main
                CLV                                 ;use standard zp/stk
                JMP         XFER                    ;there's no place like home...
*  NOTE: The previous section of code MUST NOT use $3FE & $3FF
*        since the Interrupt Vector must go there if AUX interrupts
*        are to be used.
                PAGE
*
* Transfer card part of the driver to the card
* Transfer front part of the driver to lang.card
*
RAMDest         EQU         $0200
RAMDest1        EQU         RAMDest+$100
LCDest          EQU         $FF00
*
                ifeq        os-ProDOS
*
**************** See Rev Note #60 *****************
*
Srce            equ         $2b00                   ; Note: THIS MUST CHANGE WHEN THINGS
GROW!
************************************************
                fin
                ifeq        os-EdNet
Srce            equ         $2b00                   ; The loader is bigger in EdNet's
Version.
                fin
RAMSrc          EQU         Srce+$100
RAMSrc1         EQU         Srce+$200
LCSrc           EQU         Srce+$300
```

```
*
DEVNUM          EQU          $B0                        ;Slot 3, Drive 2 ($30 + hi bit set)
DEVADR          EQU          $BF10                      ;Base of Device addresses
DEVCNT          EQU          $BF31
DEVLST          EQU          $BF32
*
                ORG          Srce
*
* Move LCSrc to LCDest
*
                LDY          #$99                       ;move $9A bytes
MvLC            LDA          LCSrc,Y                    ;get a byte of source
                STA          LCDest,Y
                DEY
                CPY          #$FF
                BNE          MVLC
*
* Move RAMSrc to RAMDest
*
                LDX          #>RAMSrc
                STX          A1L                        ;source low
                DEX                                     ;source end low
                STX          A2L
                LDX          #<RAMSrc                   ;source high
                STX          A1H
                INX
                STX          A2H                        ;end high
                LDA          #>RAMDest
                STA          A4L
                LDA          #<RAMDest
                STA          A4H
                SEC                                     ;RAM to Card
                JSR          MOVE
*
* Now install it into the system
*
                LDA          #>LCDest                   ;put LC address into
                STA          DEVNUM/8+DEVADR            ;slot 3, drive 2
                LDA          #<LCDEST
                STA          DEVNUM/8+DEVADR+1
                INC          DEVCNT
                LDX          DEVCNT
                LDA          #DEVNUM+$F                 ;unit num of /RAM
                STA          DEVLST,X
                RTS
                PAGE
*
                ORG          LCDest
EnterRAM        EQU          *                          ;/RAM entry point
                CLD
*
                LDX          #$B                        ;save 13 bytes of params
SAVPARMS        LDA          A1L,X
                STA          A1L1,X
                DEX
                BPL          SAVPARMS
*
                LDX          #1                         ;save XFER Vectors too
SVP1            LDA          PassIt,X
                STA          SP1,X
                DEX
                BPL          SVP1
*
```

```
                LDA         CMD                     ;get command
                BEQ         STAT                    ;0=STATUS
                CMP         #4                      ;check for command too high
                BCS         IOERR                   ;if it is, IO ERR
                EOR         #$3                     ;0=FORMAT,2=READ,1=WRITE
                STA         CMD                     ;CMD=>0=Format,2=Read,1=Write
                BEQ         FORMAT                  ;format the volume
                LDY         BLOCK+1                 ;check for enormous blocknum
                BNE         IOERR                   ;io error if too big
                LDA         BLOCK                   ;get the block number
                BMI         IOERR                   ;largest block is $7F
*
* At this point, control is passed to the code in the
* alternate 64K.  It is used for read, write, and
* format.  After the request is completed, control
* is always passed back to NoErr.
*
Format          EQU         *
*
                LDA         #>EnterCard             ;card entry point
                STA         PassIt                  ;figure it out on card
                LDA         #<EnterCard
GoCard          STA         PassIt+1                ;also used by MainWrt
                SEC                                 ;RAM->Card
                CLV                                 ;start with original z.p.
                JMP         XFER                    ;transfer control
*
*
IOERR           LDA         #XDIOERR                ;get err num
                BNE         ERROUT                  ;and return
WPERR           LDA         #XDWPERR                ;write protect err
ERROUT          SEC                                 ;flag error
                BCS         Restore                 ;restore cmd and unitnum
*
STAT            EQU         *
NoErr           LDA         #0                      ;no error
                CLC                                 ;flag no error
Restore         PHP                                 ;save status
                PHA                                 ;save error code
*
                LDX         #$B                     ;restore 13 bytes of params
RSTPRMS         LDA         A1L1,X
                STA         A1L,X
                DEX
                BPL         RSTPRMS
*
                LDA         SP1                     ;restore XFER params
                BIT         $6060                   ;This instruction is to put an RTS at
$FF58 as in ROM
                STA         PassIt
                LDA         SP1+1
                STA         Passit+1
*
; -------------------- See rev note 21 -----------------------
                PLA                                 ;get error
                PLP                                 ;get status
; ------------------------------------------------------------
                RTS                                 ;and return
*
MainWrt         STA         WrCardRAM               ;xfer data to card
                LDY         #0
MW1             LDA         (A1L),Y                 ;pointers set in card by SETPTR
                STA         (A4L),Y
```

```
                LDA           (A2L),Y
                STA           (A3L),Y
                DEY
                BNE           MW1
                STA           WrMainRAM                 ;done writing Card
*
                LDA           #>DoneWrt
                STA           PassIt
                LDA           #<DoneWrt
                JMP           GoCard
*
SP1             DS            2,0
A1L1            DS            $C,0                      ;13 bytes of storage
FrontLen        EQU           *-EnterRAM
*
***************** See Rev Note #EN1 *****************
*
                ifeq          os-EdNet
                lda           $c083                     ; Switch in bank 2.
                jsr           $d400                     ; Go do RWTS.
                sta           $c08b                     ; Switch bank 1 back in preserving
status!
                rts                                     ; Back to caller.
wherearewe      equ           *
                fin
*************************************************

; ###############################################################################################
; #   END OF FILE:  RAM
; #   LINES      :  578
; #   CHARACTERS :  27275
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT RPM.0.hex
================================================================================

File ............. "RPM.0"
Fork ............. DATA
Size (bytes) ..... 1,256 (1KB) / $000004E8
Created .......... Wednesday, April 12, 2006 -- 9:20:28 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:28 AM

D/000000: D84C61E0 4C45E178 0002013D 0B4C6173 [.La.LE.x...=.Las]
D/000010: 65725772 69746572 012AA0A0 A0A0A0A0 [erWriter.*......]
D/000020: A0A0A080 A0A080A0 AEA0A0D0 A0A0B0A0 [................]
D/000030: A0A0A0D4 80A0A080 A0A0A0A0 A0A0A0A0 [................]
D/000040: A0A0B0A0 A0A0A0A0 A0A0A080 A0A0FFA0 [................]
D/000050: A0A0A0A0 A0A0C6A0 A0B0A0B1 80A0A000 [................]
D/000060: 00297F8D 5EE08A48 9848ADB0 E1F00E20 [.)..^..H.H..... ]
D/000070: 0FE1ADB0 E1D03F20 D6E020BE E0AD60E0 [......?.......`.]
D/000080: D034AD07 E08DDFE4 AD08E08D E0E4ADBC [.4..............]
D/000090: E130EF08 78AD5EE0 AE5FE09D CAE1E88E [.0..x.^._.......]
D/0000A0: 5FE0D00B 8EC2E1A9 018DC3E1 203DE128 [_...........=.(]
D/0000B0: AD09E08D E4E468A8 68AAAD5E E060A941 [......h.h..^.`.A]
D/0000C0: 8DE3E4AD 09E08DE4 E4A9008D E5E4A2E3 [................]
D/0000D0: A0E42000 E560A941 8DDEE4AD 07E08DDF [.....`.A........]
D/0000E0: E4AD08E0 8DE0E4A2 DEA0E420 00E56008 [.............`.]
D/0000F0: 78A9468D E3E4A2E3 A0E42000 E5286008 [x.F.........(`.]
D/000100: 78A9468D DEE4A2DE A0E42000 E52860A2 [x.F.........(`.]
D/000110: AFA0E120 00E5A900 8D5FE08D 60E0ADB0 [........._..`...]
D/000120: E130FBAD B3E18DBF E18DC9E1 A9008DBC [.0..............]
D/000130: E18DC2E1 8DBEE1A9 018DC3E1 60A2BBA0 [............`...]
D/000140: E12000E5 6020EFE0 20FFE0A9 018D60E0 [....`.........`.]
D/000150: A9708DBD E1A9E18D BEE1AD5F E08DC2E1 [.p........._....]
D/000160: A9008D5F E08DC3E1 A2BBA0E1 2000E560 [..._...........`]
D/000170: A9828DC7 E1A9E18D C8E1A2C5 A0E12000 [................]
D/000180: E560A9FF 8DB0E160 ADBCE130 17AD5FE0 [.`.....`...0.._.]
D/000190: F0128DC2 E1A9008D 5FE08DC3 E1A2BBA0 [........_.......]
D/0001A0: E12000E5 AD09E08D E4E46020 45E16042 [.........`.E.`B]
D/0001B0: FF0000FF DEE20AE0 01CAE243 00000000 [...........C....]
D/0001C0: CAE10001 00450000 0000A0AF 80A0A0A0 [.....E..........]
D/0001D0: A0B0A0A0 A0A0A0A0 A0A0A0A0 A080A0A0 [................]
D/0001E0: 82A0A0A0 A0D0A0A0 A0A0A0A0 A0A080A0 [................]
D/0001F0: A080A0A0 A0A0C9A0 A0A0A0A0 B0A0A0A0 [................]
D/000200: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000210: A0A0A0A0 A0A0A0A0 AFA0A0B1 A0A0A0A0 [................]
D/000220: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000230: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000240: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000250: A0A0A0A0 A0FFA0A0 A5A0CDA0 A0A0A0A0 [................]
D/000260: A0A0A0B0 A0CEFFA0 B1CAA0A0 A0A0A0A0 [................]
D/000270: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0FFA0A0 [................]
D/000280: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000290: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/0002A0: A0A0A0A0 A0A0A0A0 A0A0A080 A0A0DAA0 [................]
D/0002B0: A0A0A0B0 A0A0A0A0 A080A0A0 AEA0A0A0 [................]
D/0002C0: A0B2A0A0 CAA0A0A0 A0B0A0A0 A0A0A0A0 [................]
D/0002D0: A0A0A0A0 80A0A080 A0A0A0A0 A0A0A0A0 [................]
D/0002E0: FFA0A0A0 A0A0A0A0 C6A0A0B0 A0B180A0 [................]
D/0002F0: A083A0A0 A0A0C3A0 A0A0A0A0 FFA0A0A0 [................]
D/000300: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000310: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000320: A0A0A0A0 A0A0A080 A0A082A0 A0A0A0D0 [................]
D/000330: A0A0A0A0 A0A0A0A0 80A0A080 A0A0A0A0 [................]
```

```
D/000340: C9A0A0A0 A0A0B0A0 B0FFA0A0 A0A0A080 [................]
D/000350: A0A080A0 A0A0A0A0 A0A0C6A0 A0B0A0B1 [................]
D/000360: 80A0A083 A0A0A0A0 C3A0A0C5 A0A0FFA0 [................]
D/000370: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000380: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000390: A0A0A0A0 A0A080A0 A081A0A0 A0A0A0A0 [................]
D/0003A0: A0A0A0A0 C5A0A080 A0D5F0A0 A0A0A0A0 [................]
D/0003B0: A0A0CFA0 A0ADA0B0 FFA0CFBF A0A080A0 [................]
D/0003C0: A0C1A0A0 A0A0A0A0 A0A0A0A0 A0A0A0FF [................]
D/0003D0: A0A080A0 C5A0A0B0 A0A0A0A0 A0FFA0A0 [................]
D/0003E0: FFA0A086 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/0003F0: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000400: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0D0A0A0 [................]
D/000410: CFA0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000420: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000430: A0A0A0A0 A0A0A0A0 A0A0A0A0 80A0A082 [................]
D/000440: A0A0A0A0 A0A0A0A0 A0A080A0 A088A0A0 [................]
D/000450: A0A0B1A0 A0B0A0A0 A0A0AEFF A0C780A0 [................]
D/000460: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000470: A0A080A0 A0FFA0D2 A0A0A0A0 A0A0A0A0 [................]
D/000480: A0A0A0A0 A080A0A0 81A0A0A0 A0A0A0A0 [................]
D/000490: A0A0A0C5 A0A080A0 D5F0A0A0 A0A0A0A0 [................]
D/0004A0: A0CFA0A0 B0A0B0FF A0CFBFA0 A080A0A0 [................]
D/0004B0: C1A0A0A0 A0A0A0A0 A0A0A0CF A0A0FFA0 [................]
D/0004C0: A080A0C5 A0A0B0A0 A0A0A0A0 FFA0CC86 [................]
D/0004D0: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A04128 [..............A(]
D/0004E0: 00ABE141 000088E1                    [...A....        ]

File ............. "RPM.0"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

FINIS
```

```
================================================================================
DOCUMENT SEL.0.hex
================================================================================

File ............. "SEL.0"
Fork ............. DATA
Size (bytes) ..... 715 (0KB) / $000002CB
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: AD82C08D 0CC08D0E C08D00C0 2084FE20 [................]
D/000010: 2FFB2093 FE2089FE A217A901 9D58BFCA [/............X..]
D/000020: A9009D58 BFCA10FA A9CF8D58 BF2058FC [...X.......X..X.]
D/000030: 208EFDA2 0020D611 A9038525 208EFD20 [...........%....]
D/000040: 00BFC7C8 12AE8002 A9009D81 02AE8002 [................]
D/000050: F00BBD80 0209809D FF05CAD0 F5A200C6 [................]
D/000060: 25208EFD 200CFDC9 8DF05248 209CFC68 [%.........RH...h]
D/000070: C99BF0B9 C998F0B5 C989F017 C9FFF004 [................]
D/000080: C988D00D E000F003 C624CA20 9CFC4C64 [.........$....Ld]
D/000090: 10B00620 3AFF4C64 10C9DB90 0229DFC9 [....:.Ld.....)..]
D/0000A0: AE90F0C9 DBB0ECC9 BA9004C9 C190E4E8 [................]
D/0000B0: E027B0C2 9D800220 EDFD4C64 10E000F0 [.'........Ld....]
D/0000C0: 128E8002 2000BFC6 C8129007 203AFFA9 [............:..]
D/0000D0: 00F0A320 58FC208E FDA22820 D611A903 [....X.....(.....]
D/0000E0: 8525208E FDA20020 0CFDC99B D006A524 [.%.............$]
D/0000F0: D0E1F0DD C998F0DB C989F00D C9FFF004 [................]
D/000100: C988D003 4CC011B0 06203AFF 4CE710C9 [....L.....:.L...]
D/000110: 8DF029C9 DB900229 DFC9AE90 ECC9DBB0 [..)....).........]
D/000120: E8C9BA90 04C9C190 E048209C FC6820ED [.........H...h..]
D/000130: FDE8E027 B0C09D80 024CE710 A9A020ED [...'.....L......]
D/000140: FD8E8002 2000BFC4 A1129003 4CE211AD [............L...]
D/000150: A512C9FF F005A901 4CE211A9 008DBA12 [........L.......]
D/000160: 2000BFCC B9129003 4CE211AD A4122901 [........L.....).]
D/000170: D005A927 4CE21120 00BFC8B3 1290034C [...'L..........L]
D/000180: E211ADB8 128DBC12 8DC41220 00BFD1C3 [................]
D/000190: 12B04FAD C712F004 A927D046 ADC5128D [..O......'.F....]
D/0001A0: BF12ADC6 128DC012 2000BFCA BB120820 [................]
D/0001B0: 00BFCCB9 12900428 D02828B0 FA4C0020 [.......(.((..L..]
D/0001C0: A524F00F CAA9A020 EDFDC624 C62420ED [.$.........$.$..]
D/0001D0: FDC6244C E710BD11 12F00620 EDFDE8D0 [..$L..........]
D/0001E0: F56085DE A90C8525 208EFDA5 DEC901D0 [.`.....%........]
D/0001F0: 04A24BD0 16C940F0 10C944F0 0CC945F0 [..K...@...D...E.]
D/000200: 08C946F0 04A262D0 02A27920 D6114CDE [..F...b...y...L.]
D/000210: 10C5CED4 C5D2A0D0 D2C5C6C9 D8A0A8D0 [................]
D/000220: D2C5D3D3 A0A2D2C5 D4D5D2CE A2A0D4CF [................]
D/000230: A0C1C3C3 C5D0D4A9 00C5CED4 C5D2A0D0 [................]
D/000240: C1D4C8CE C1CDC5A0 CFC6A0CE C5D8D4A0 [................]
D/000250: C1D0D0CC C9C3C1D4 C9CFCE00 87CECFD4 [................]
D/000260: A0C1A0D4 D9D0C5A0 A2D3D9D3 A2A0C6C9 [................]
D/000270: CCC50087 C9AFCFA0 C5D2D2CF D2A0A0A0 [................]
D/000280: A0A0A0A0 A0A0A0A0 A00087C6 C9CCC5AF [................]
D/000290: D0C1D4C8 A0CECFD4 A0C6CFD5 CEC4A0A0 [................]
D/0002A0: 000A8002 00000000 00000000 00000000 [................]
D/0002B0: 00000003 80020018 00010004 00002000 [................]
D/0002C0: 00000002 00000000 018002            [...........     ]

File ............. "SEL.0"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997
```

```
FINIS
```

```
================================================================================
DOCUMENT SEL.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  SEL
; ###############################################################################################

                LST          OFF,NOASYM,NOVSYM,NOGEN
                X65816
                XREFSLOT     1
                SEG          $00
*
************************************************************
*                                                          *
*          PRODOS 8 LOBOTOMIZED DISPATCHER ROUTINE         *
*                                                          *
*           COPYRIGHT APPLE COMPUTER, INC., 1983-86        *
*                                                          *
*                     ALL RIGHTS RESERVED                  *
*                                                          *
************************************************************
*
                SBTL         "DISPATCHER I" "
************************************************************
*
*  DISPATCHER 1 - This code ORGs and operates at $1000 but
*    is resident in memory at $D100 in the Alt 4K bank of the
*    Language Card.  The QUIT call vectors to a routine high
*    in the MLI that moves DISPATCHER 1 down and jumps to it.
*    The move routine MUST remain somewhere between $E000-$F7FF.
*
*  NOTE: This entire routine MUST remain no larger than 3 pages.
*
************************************************************
                ORG          $1000
                MSB          ON
PROMPT          EQU          $FF
CV              EQU          $25
CH              EQU          $24
INFOCALL        EQU          $C4
SETPFXCALL      EQU          $C6
GETPFXCALL      EQU          $C7
OPENCALL        EQU          $C8
READCALL        EQU          $CA
CLOSECALL       EQU          $CC
EOFCALL         EQU          $D1
ERRNUM          EQU          $DE                     ;Applesoft ERRNUM but who cares?
PATH            EQU          $280                    ;Second half of input buffer
*
line3           equ          $600                    ; 3rd line of screen (starting from 0).
*
ENTRY           EQU          $2000
MLI             EQU          $BF00
BITMAP          EQU          $BF58
PFIXPTR         EQU          $BF9A
KBD             EQU          $C000
STB             EQU          $C010
RAMIN           EQU          $C08B
ROMIN           EQU          $C082
RAMIN2          EQU          $C083
```

```
CLEOL           EQU          $FC9C                    ;Clear to End of Line
settxt          equ          $fb39                    ; Monitor routine to set 40 col window.
HOME            EQU          $FC58
RDKEY           EQU          $FD0C
COUT            EQU          $FDED
CROUT           EQU          $FD8E
BELL            EQU          $FF3A
*
* Monitor equates, soft switch equates for fix #55....
*
setvid          equ          $fe93                    ; Puts COUT1 in CSW.
setnorm         equ          $fe84                    ; Normal white text on black backround.
init            equ          $fb2f                    ; Text pg1;text mode;sets 40/80 col
window.
setkbd          equ          $fe89                    ; Does an IN#1.
clr80vid        equ          $c00c                    ; Disable 80 column hardware.
clraltchar      equ          $c00e                    ; Switch in primary character set.
clr80col        equ          $c000                    ; Disable 80 column store.
*
*  Reset the Rev e soft switches
*
*********** See rev note #55 ***************
HEREIN          equ          *
                LDA          ROMIN
* JSR $FE93 ;SETVID
* JSR $FE89 ;SETNORM
* STA $C00C ;80 col off
* STA $C00F ;Alt chrset on
* STA $C000 ;Main ram
                sta          clr80vid                 ; Disable 80 column hardware.
                sta          clraltchar               ; Switch in primary char set.
                sta          clr80col                 ; Disable 80 column store.
                jsr          setnorm                  ; Normal white chars on black
backround.
                jsr          init                     ; Text pg1; text mode; set 40 col
window.
                jsr          setvid                   ; Does a PR#0 (puts COUT1 in CSW).
                jsr          setkbd                   ; Does an IN#0 to set Basic input to
kbd.
************************************************
*
*  Clear the memory Bit Map
*
CLRMAP          LDX          #$17                     ;Do all the bytes
                LDA          #1
                STA          BITMAP,X                 ;Protect page $BF00
                DEX
                LDA          #$0                      ;Clear the rest
CLRLOOP         STA          BITMAP,X
                DEX
                BPL          CLRLOOP
                LDA          #$CF
                STA          BITMAP                   ;Protect pages 0,1 & $400-$7FF (Screen)
START           EQU          *
*
***************** See Rev Note #39 *********************
*
* jsr settxt ; No longer necessary with fix #55...
*******************************************************
                JSR          HOME                     ; Clear the screen
                JSR          CROUT
*
******************** Rev Note #55  ********************
```

```
*
                ldx         #>msg0-msgstart          ; Load offset to message into x...
*****************************************************
*
                JSR         PRNTLOOP
                LDA         #3                       ;Set CV to 3rd line
                STA         CV
                JSR         CROUT                    ;  and col 1
                JSR         MLI                      ;Call the MLI (Remember, this code
executes at $1000)
                DFB         GETPFXCALL
                DW          PREFIX
                LDX         PATH                     ;Get PREFIX length
                LDA         #0                       ;Put a 0 at end of Prefix
                STA         PATH+1,X
*
****************** See Rev Note #69 ******************
*
                ldx         path                     ; Get length byte back.
                beq         nilpfx                   ; Branch if no prefix to display!!
*****************************************************
*
********************* Rev Note #55  ******************
*
*LDA #0 ;Put a 0 at end of Prefix
*STA PATH+1,X
*LDA #>PATH+1 ;Put the prefix address in print loop
*STA LOOP+1
*LDA #<PATH
*STA LOOP+2
*JSR PRNTLOOP
m1              lda         path,x                   ; Display prefix directly
                ora         #$80                     ; Set hi bit for NORMAL text.
                sta         line3-1,x                ; to the screen.
                dex                                  ; Next....
                bne         m1                       ; Branch until prefix displayed.
*****************************************************
*
****************** See Rev Note #69 ******************
*
nilpfx          equ         *
*****************************************************
                LDX         #0
                DEC         CV                       ;
                JSR         CROUT                    ;Put the cursor on the first char
GETKEY          JSR         RDKEY                    ;Wait for keyboard input
                CMP         #$8D                     ;Is it CR?
                BEQ         GOTPFX                   ;Yes, and we accept what was entered
                PHA                                  ;No, save the char
                JSR         CLEOL                    ;Clear rest of line
                PLA                                  ;Get char back
                CMP         #$9B                     ;Is it ESC?
                BEQ         START                    ;Yes, start over again
                CMP         #$98                     ;If it is CTRL-X, start over.
RESTRT          BEQ         START                    ;(Used as an extended BEQ from PRMPT)
                CMP         #$89                     ;Is it TAB?
                BEQ         BADKEY                   ;No good if it is!
                CMP         #$FF                     ;Delete?
                BEQ         X2                       ;Branch if it is
                CMP         #$88                     ;Back Space?
                BNE         NOTBS                    ;Branch if not
X2              CPX         #$0                      ;If it is, are we at col 0?
                BEQ         *+5                      ;If col 0, do nothing
```

```
                DEC         CH                      ; else move left 1 char
                DEX                                 ; decrement char count,
                JSR         CLEOL                   ; clear rest of line
                JMP         GETKEY                  ;Go get another char
NOTBS           BCS         MAYBE
BADKEY          JSR         BELL                    ;Ring the speaker (bell) if it isn't.
                JMP         GETKEY
MAYBE           CMP         #$DB                    ;Ok, is it below 'Z'?
                BCC         *+4                     ;Branch if yes
                AND         #$DF                    ;If not, shift it up upper case
                CMP         #$AE                    ;Is it below '.'?
                BCC         BADKEY                  ;If yes, it ain't good!
                CMP         #$DB                    ;Is it above 'Z'?
                BCS         BADKEY                  ;If so, it also ain't good
                CMP         #$BA                    ;Is it below ':'? ('.' - '9' range)
                BCC         GOODKEY                 ;Yes, it's good!
                CMP         #$C1                    ;If not, is it at or above 'A'? ('A' -
'Z')
                BCC         BADKEY                  ;No, reject it
GOODKEY         INX                                 ;It's OK.  Hallelulah!
                CPX         #39                     ;Were there more than 39 chars?
                BCS         RESTRT                  ;Yes, too many!  Go restart.
                STA         PATH,X                  ;No, save the lucky char
                JSR         COUT                    ;Print it
                JMP         GETKEY                  ;  and go get another.
GOTPFX          EQU         *
                CPX         #$0                     ;OK, is our Prefix length (chars
entered)=0?
                BEQ         PRMPT                   ;If yes, don't bother re-setting it
                STX         PATH                    ;Set prefix length
                JSR         MLI                     ;Call the MLI
                DFB         SETPFXCALL
                DW          PREFIX
                BCC         PRMPT                   ;If ok, go get Filename
                JSR         BELL                    ;If not, ring Bell
                LDA         #0                      ;  and try again
BADPFX          BEQ         RESTRT                  ;Z flag must be set for extended Branch
PRMPT           JSR         HOME                    ;Clear the screen for application name
PRMPT1          JSR         CROUT                   ;Output a CR
*
********************* Rev Note #55  *******************
*
                ldx         #>msg-msgstart          ; Load offset to message into x...
*****************************************************
*
                JSR         PRNTLOOP
retryrich       LDA         #3                      ;Set CV to 3rd line
                STA         CV
                JSR         CROUT                   ;  and col 1
                LDX         #0
*
********************* Rev Note #69  *******************
*
*LOOP1 LDA #PROMPT ;Our cursor char
* JSR COUT ;Print it
* DEC CH ;  and point to it so a typed
* LDA KBD ;  character can replace it.
*BPL *-3 ;(We'll wait here till a keypress.)
* STA STB ;Hit the strobe
loop1           equ         *
                jsr         rdkey
                CMP         #$9B                    ;ESC
                BNE         NOTESC
```

```
                LDA         CH
                BNE         PRMPT
                BEQ         BADPFX                  ;If ESC in col 0 go get PREFIX again
NOTESC          CMP         #$98                    ;CTRL-X
EXTNDBR         BEQ         PRMPT                   ;(Used as a branch extender)
                CMP         #$89                    ;TAB
                BEQ         NOTGUD
                CMP         #$FF                    ;Delete?
                BEQ         X3
                CMP         #$88                    ;BACK SPACE
                BNE         X1
X3              JMP         EATEM                   ;Eat the previous character.
X1              BCS         GETIN1                  ;> $88 and the char may be acceptable
NOTGUD          JSR         BELL                    ;Ring the bell (speaker)
                JMP         LOOP1
GETIN1          CMP         #$8D                    ;Is it a CR?
                BEQ         DONE
                CMP         #$DB                    ;> than Z
                BCC         *+4                     ;No.
                AND         #$DF                    ;Make sure its Upper case
                CMP         #$AE                    ;Is it "."?
                BCC         NOTGUD                  ;Branch if less
                CMP         #$DB                    ;Must be less than "[".
                BCS         NOTGUD
                CMP         #$BA                    ;OK if less than or equal to "9"
                BCC         ITSGUD
                CMP         #$C1                    ;Else must be > than "A"
                BCC         NOTGUD
ITSGUD          EQU         *
                PHA
                JSR         CLEOL
                PLA
                JSR         COUT                    ;No, print it
                INX
                CPX         #39
                BCS         EXTNDBR
                STA         PATH,X
                JMP         LOOP1                   ;Go get the next one
DONE            EQU         *
                LDA         #'                      '
                JSR         COUT                    ;After the CR, blank out the cursor.
                STX         PATH                    ;Put the length in front of the name.
*
*  At this point the specified Pathname is in PATH ($280)
*    and we can do a GET_FILE_INFO on it.
*
                JSR         MLI
                DFB         INFOCALL
                DW          INFO
                BCC         INFOOK
                JMP         ERROR
INFOOK          LDA         TYPE
                CMP         #$FF                    ;Is it a type SYS file?
                BEQ         DOIT
                LDA         #1                      ;Not SYS File
                JMP         ERROR
DOIT            EQU         *                       ;It's a type SYS all right!
                LDA         #0
                STA         CLSNUM
                JSR         MLI
                DFB         CLOSECALL               ;CLOSE all open files first
                DW          CLS
                BCC         CHKACS
```

```
                JMP         ERROR
*
*  Now check for the proper access
*
CHKACS          LDA         ACESS                   ;Get the allowed access
                AND         #1                      ;Is READ disabled?
                BNE         ACSOK                   ;No. Access ok.
                LDA         #$27                    ;I/O error
                JMP         ERROR                   ;Never returns!
ACSOK           EQU         *
                JSR         MLI
                DFB         OPENCALL
                DW          OPN                     ;OPEN it.
                BCC         *+5
                JMP         ERROR
                LDA         REFNUM
                STA         REEDNUM                 ;Spread REFNUM around
                STA         EOFNUM
*
*  Ok it's OPEN, let's get the EOF
*
                JSR         MLI
                DFB         EOFCALL
                DW          EOF
                BCS         ERROR
                LDA         EOFB+2                  ;3rd of 3 bytes
                BEQ         EOFOK
                LDA         #$27                    ;I/O ERROR even though the file is
                BNE         ERROR                   ;  simply too large
EOFOK           LDA         EOFB                    ;Move EOF to Read # bytes
                STA         RCOUNT
                LDA         EOFB+1
                STA         RCOUNT+1
                JSR         MLI
                DFB         READCALL                ;Do the READ
                DW          REED
                PHP                                 ;Push the processor status
                JSR         MLI
                DFB         CLOSECALL               ;Close it
                DW          CLS
                BCC         *+6
                PLP                                 ;Get status back (it is irrevalent now)
                BNE         ERROR                   ;(if CLOSE generated an error)
                PLP                                 ;We're here if CLOSE was OK
                BCS         *-4                     ;JMP ERROR
                JMP         ENTRY
*
EATEM           EQU         *
                LDA         CH                      ;Is the cursor in col 0?
                BEQ         EATEMBAK                ;Yes, ignore it.
                DEX
                LDA         #'                      '
                JSR         COUT                    ;Blank out the cursor
                DEC         CH                      ;Point to last character
                DEC         CH                      ;  entered...
                JSR         COUT                    ;  and blank it too.
                DEC         CH                      ;Point to that location
EATEMBAK        JMP         LOOP1                   ;Go back & get the next char
*
****************** See Rev Note #55  *****************
*
*PRNTLOOP LDX #0 ;Index into message
*LOOP LDA *,X ;CAUTION:  SELF-MODIFYING CODE
```

```
* BEQ LOOPRTN ;  ADDR FILLED W/ LOC OF MESSAGE.
* ORA #$80
* JSR COUT ;Print message on screen
* INX
* BNE LOOP ;Loop til done (rotisserie mode)
*LOOPRTN RTS
*
prntloop        equ          *
                lda          msgstart,x               ; Display string; offset is in X.
                beq          done1                    ; Branch if done.
                jsr          cout                     ; Output character...
                inx                                   ; Next..
                bne          prntloop                 ; Branch always.
done1           rts                                   ; Done.
************************************************
ERROR           EQU          *
                STA          ERRNUM
                LDA          #$0C                     ;Put error message on line 13
                STA          CV
                JSR          CROUT
                LDA          ERRNUM
                CMP          #1
                BNE          NEXTERR
*
*************** See Rev Note #55  **************
*
* LDA #>ERR1
* STA LOOP+1
* LDA #<ERR1
* STA LOOP+2
*
                ldx          #>err1-msgstart          ; Load x with offset to message.
*********************************************
                BNE          DOERROR
NEXTERR         CMP          #$40
                BEQ          ERROR3
                CMP          #$44
                BEQ          ERROR3
                CMP          #$45
                BEQ          ERROR3
                CMP          #$46
                BEQ          ERROR3
*
*************** See Rev Note #55  **************
*
* LDA #>ERR2
* STA LOOP+1
* LDA #<ERR2
* STA LOOP+2
                ldx          #>err2-msgstart          ; Load x with offset to message.
*********************************************
                BNE          DOERROR
*
*************** See Rev Note #55  **************
*
*ERROR3 LDA #>ERR3
* STA LOOP+1
* LDA #<ERR3
* STA LOOP+2
error3          equ          *
                ldx          #>err3-msgstart          ; Load x with offset to message.
*********************************************
DOERROR         JSR          PRNTLOOP
```

```
*
******************** Rev Note #69  ******************
*
* LDA #0 ; Printloop will leave A=0...See Rev Note #39
*STA CV
                JMP          retryrich
*
************************************************************
msgstart        equ          *
                MSB          ON
MSG0            ASC          'ENTER PREFIX (PRESS "RETURN" TO ACCEPT)'
                DFB          0
MSG             ASC          "ENTER PATHNAME OF NEXT APPLICATION" "
                DFB          0
ERR1            DFB          $87                        ;BELL
                ASC          'NOT A TYPE "SYS" FILE'
                DFB          0
ERR2            DFB          $87
                ASC          'I/O ERROR            ' '
                DFB          0
ERR3            DFB          $87
                ASC          'FILE/PATH NOT FOUND  ' '
                DFB          0
*
************************************************************
*
INFO            DFB          $A                         ;10 PARAMETERS ON GFI
                DW           PATH                       ;Pathname buffer pointer
ACESS           DFB          0                          ;ACCESS
TYPE            DFB          0                          ;File Type
                DS           $0D,0                      ;All the rest are unimportant
*
OPN             DFB          3                          ;3 parameters on an OPEN
                DW           PATH
                DW           $1800                      ;FCB Buffer
REFNUM          DFB          0
*
CLS             DFB          1
CLSNUM          DFB          0                          ;REFERENCE #
*
REED            DFB          4                          ;4 Parameters for a READ
REEDNUM         DFB          0
                DW           ENTRY                      ;SYS files always load at $2000
RCOUNT          DW           0
                DW           0
*
EOF             DFB          2
EOFNUM          DFB          0
EOFB            DS           3,0                        ;Three byte EOF
*
PREFIX          DFB          1
PBUF            DW           PATH
*
ZZSIZ           EQU          *-HEREIN
ZZFRE           EQU          $2FF-ZZSIZ
*

; ##############################################################################################
; #   END OF FILE:  SEL
; #   LINES      :  480
; #   CHARACTERS :  22173
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################################
```

```
================================================================================
DOCUMENT TCLOCK.0.hex
================================================================================

File ............. "TCLOCK.0"
Fork ............. DATA
Size (bytes) ..... 125 (0KB) / $0000007D
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: AE50D7BD 380548A9 A3200BC1 2008C118 [.P..8.H.........]
D/000010: A204A00C B9000229 07853A0A 0A653A0A [.......)..:..e:.]
D/000020: 79010238 E9B0953A 888888CA 10E6A84A [y..8...:.......J]
D/000030: 6A6A6A05 3C8D90BF 08291F79 ABD79002 [jjj.<....).y....]
D/000040: 690338E9 07B0FC69 07E53BB0 026907A8 [i.8....i..;..i..]
D/000050: B9B8D728 2A8D91BF A53D8D93 BFA53E8D [...(*....=....>.]
D/000060: 92BF68AE 50D79D38 0560001F 3B5A7897 [..h.P..8.`..;Zx.]
D/000070: B5D3F214 33515A59 58585756 5B       [....3QZYXXWV[   ]

File ............. "TCLOCK.0"
Fork ............. RESOURCE
Size (bytes) ..... 0 (0KB) / $00000000


Brought to you by: dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

FINIS
```

```
================================================================================
DOCUMENT TCLOCK.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  TCLOCK
; ###############################################################################################

                LST          OFF,NOASYM,NOVSYM,NOGEN
                X65816
                XREFSLOT     6
                SEG          $00
*
*************************************************************
*                                                           *
*          PRODOS 8 CLOCK DRIVER INTERFACE ROUTINE          *
*                                                           *
*           COPYRIGHT APPLE COMPUTER, INC., 1983-86         *
*                                                           *
*                     ALL RIGHTS RESERVED                   *
*                                                           *
*************************************************************
*
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
                ifeq         os-ProDOS
                org          $d742
                fin
                ifeq         os-EdNet
                org          $f842
                fin
                SBTL         "ProDOS Clock Interface"
****************************************
*    CLOCK - PRODOS INTERFACE ROUTINE
*          BY JAMES R. HUSTON
*     WARNING: HARD CODED FOR SLOT 1
****************************************
                SKP          1
TENS            EQU          $3A                      ;NO CONFLICT SINCE MONTH IS LAST
PROCESSED.
MONTH           EQU          $3A
WKDAY           EQU          $3B
DAY             EQU          $3C
HOUR            EQU          $3D
MINUTE          EQU          $3E
                SKP          1
WTTCP           EQU          $C10B
RDTCP           EQU          $C108                    ;CLOCK READ ENTRY POINTS.
CLKMODE         EQU          $538                     ;(+$CN=$5F8+N)
                SKP          1
DATE            EQU          $BF90                    ;PRODOS DATE
TIME            EQU          $BF92                    ;PRODOS TIME.
                SKP          1
INBUF           EQU          $200                     ;INPUT BUFFER.
                SKP          1
READCLK         LDX          CLKSLT                   ;PRESERVE CURRENT MODE FOR THUNDERCLOCK
                LDA          CLKMODE,X
                PHA
                LDA          #$A3                     ;SEND NUMERIC MODE BYTE TO THUNDERCLOCK
```

```
                JSR         WTTCP
CLKSLT          EQU         *+2
                JSR         RDTCP                   ;READ MONTH, DAY OF WEEK, DAY OF MONTH,
AND TIME
                CLC                                 ; INTO INPUT BUFFER.
                LDX         #4                      ;INDEX FOR 5 VALUES
                LDY         #$C                     ;READ MINUTES FIRST, MONTH LAST.
CONVRT          LDA         INBUF,Y                 ;CONVERT VALUES TO BINARY.
                AND         #$7                     ;NO VALUE > 5 DECIMAL.
                STA         TENS                    ;MULTIPLY 'TENS' PLACE VALUE.
                ASL         A
                ASL         A
                ADC         TENS                    ;NOW IT'S TIMES 5.
                ASL         A                       ;NOW IT IS TIMES 10!
                ADC         INBUF+1,Y               ;ADD TO ASCII 'ONES' PLACE
                SEC                                 ;AND SUBTRACT OUT THE ASCII...
                SBC         #$B0
                SKP         1
CNVRT2          STA         MONTH,X                 ;SAVE CONVERTED VALUE.
                DEY                                 ;INDEX TO NEXT LOWER VALUE.
                DEY
                DEY
                DEX                                 ;ARE THERE MORE VALUES?
                BPL         CONVRT                  ;BRANCH IF THERE ARE.
                TAY                                 ;ACC STILL CONTAINS MONTH, SAVE IN Y
FOR NOW.
                LSR         A
                ROR         A
                ROR         A
                ROR         A                       ;(HI BIT OF MONTH HELD IN CARRY)
                ORA         DAY
                STA         DATE                    ;SAVE LOW VALUE OF DATE.
                PAGE
                PHP                                 ;SAVE HI BIT OF MONTH FOR NOW.
                AND         #$1F                    ;ISOLATE DAY AGAIN.
; (WHEN MONTH >7 CARRY SET ACCOUNTED FOR IN FOLLOWING ADD)
                ADC         TDAYS-1,Y               ;REMEMBER THAT Y=MONTH.
                BCC         CNVRT3                  ;BRANCH NOT SEPT 13 THRU 30.
                ADC         #3                      ;ADJUST FOR MOD 7 WHEN DAY > 256.
CNVRT3          SEC
CNVRT4          SBC         #7
                BCS         CNVRT4                  ;LOOP UNTIL LESS THAN 0.
                ADC         #7                      ;NOW MAKE IT IN THE RANGE OF 0-6.
                SBC         WKDAY                   ; THE DELTA PROVIDES YEARS OFFSET.
                BCS         CNVRT5                  ;BRANCH IF POSITIVE.
                ADC         #7                      ;ELSE MAKE IT POSITIVE AGAIN.
CNVRT5          TAY                                 ;LOOK UP YEAR!
                LDA         YRADJ,Y
                PLP                                 ;LASTLY, COMBINE WITH HI BIT OF MONTH.
                ROL         A
                STA         DATE+1                  ;AND SAVE IT.
                LDA         HOUR
                STA         TIME+1                  ;MOVE HOUR AND MINUTE TO PRODOS
GLOBALS.
                LDA         MINUTE
                STA         TIME
                PLA
                LDX         CLKSLT                  ;RESTORE PREVIOUS MODE.
                STA         CLKMODE,X
                RTS                                 ;ALL DONE...
                SKP         2
TDAYS           DFB         $0,$1F,$3B,$5A
                DFB         $78,$97,$B5,$D3
```

```
               DFB          $F2,$14,$33,$51
               SKP          1
*
********** See Rev Note #51 **********
*
yradj          dfb          90,89,88,88               ; New year table.
               dfb          87,86,91                  ; Good thru 1991.
*YRADJ DFB $54,$54,$53,$52
* DFB $57,$56,$55
****************************************

; ###############################################################################################
; #   END OF FILE:  TCLOCK
; #   LINES      :  122
; #   CHARACTERS :  6147
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT XRW.pretty
================================================================================

; ###############################################################################################
; #   PROJECT  :  Apple ][ ProDOS 1.7 OS Source Listing -- Apple Computer -- July 1988
; #   FILE NAME:  XRW
; ###############################################################################################

                LST          OFF,NOASYM,NOVSYM,NOGEN
                X65816
                XREFSLOT     6
                SEG          $00
************************************************
*
*          PRODOS 8 DISK II DRIVER (RWTS)
*
*         COPYRIGHT APPLE COMPUTER INC., 1980-1986
*
*             ALL RIGHTS RESERVED
*
*          REVISED 11/8/82 BY J.R.H.
*
************************************************
ProDOS          equ          0
EdNet           equ          1
os              equ          ProDOS
*
                ifeq         os-ProDOS
                org          $d000
                fin
                ifeq         os-EdNet
                org          $d400
                fin
                INCLUDE      MLI.SRC/XRW1
                INCLUDE      MLI.SRC/XRW2

; ###############################################################################################
; #   END OF FILE:  XRW
; #   LINES      :  27
; #   CHARACTERS :  764
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

THE END