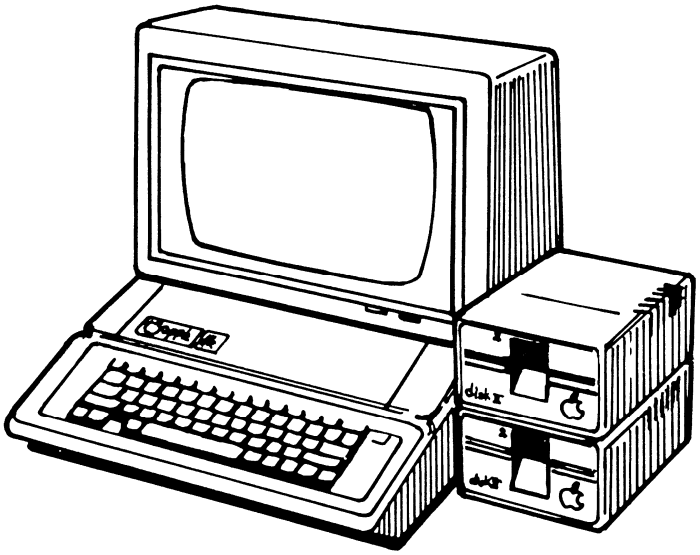
Apple II Computer Information

Apple II DOS 3.3 C Source Code Listing

July 1983



Source File Catalog

Name	Type	Crtr	Size	Flags	Last-Mod-Date	Creation-Date
ASMIDSTAMP.hex	TEXT	MPS	4K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
BLDFTAB.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:49 PM	4/14/06 3:23 PM
BLOCKIO.hex	TEXT	MPS	8K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
BOOTLDR.pretty	TEXT	MPS	24K	lvbspoimad	4/14/06 3:52 PM	4/14/06 3:23 PM
CMDSCAN.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:52 PM	4/14/06 3:23 PM
CMDTBLS.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
COREQUS.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
DOS.TO.DISK	TEXT	ttxt	4K	lvbspoImad	4/12/06 9:20 AM	4/12/06 9:20 AM
DOS33C.OBJ.hex	TEXT	MPS	44K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
DOS33C.pretty	TEXT	MPS	8K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
DOSGOER.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
DOSHOOK.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:49 PM	4/14/06 3:23 PM
DOSINIT.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:49 PM	4/14/06 3:23 PM
DOSPTCH.pretty	TEXT	MPS	8K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
EASM.pretty	TEXT	MPS	8K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
FDELCAT.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
FDOSENT.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:49 PM	4/14/06 3:23 PM
FLOCNXB.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
FLOCSEC.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:52 PM	4/14/06 3:23 PM
FMTRWIO.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
FOPCLRW.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
FORMATR.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
FVCBUFS.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:50 PM	4/14/06 3:23 PM
HELLO.A.hex	TEXT	MPS	4K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
HELLO.B.hex	TEXT	MPS	8K	lvbspoimad	4/14/06 3:37 PM	4/14/06 3:37 PM
MAKE.MASTER.hex	TEXT	MPS	4K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
MASTER.3.3E.hex	TEXT	MPS	8K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
MASTERE.OBJ0.hex	TEXT	MPS	4K	lvbspoimad	4/14/06 2:48 PM	4/14/06 2:48 PM
MSWAITR.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
POSTNRD.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:50 PM	4/14/06 2:53 PM
PRENIBL.pretty	TEXT	MPS	8K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
RDADSEK.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
RELOCTR.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
RWTSONE.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
RWTSTWO.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
TEMPY.pretty	TEXT	MPS	8K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
TRASH.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
WRITADR.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
WRITRTN.pretty	TEXT	MPS	12K	lvbspoimad	4/14/06 3:51 PM	4/14/06 2:53 PM
XLODSAV.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
XMISCMD.pretty	TEXT	MPS	16K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM
XOPNCLS.pretty	TEXT	MPS	20K	lvbspoimad	4/14/06 3:51 PM	4/14/06 3:23 PM

```
================================================================================
DOCUMENT ASMIDSTAMP.hex
================================================================================

File ............. "ASMIDSTAMP"
Fork ............. DATA
Size (bytes) ..... 17 (0KB) / $00000011
Created .......... Wednesday, April 12, 2006 -- 9:22:06 AM
Modified ......... Wednesday, April 12, 2006 -- 9:22:06 AM

D/000000: B1B2ADCA D5CCADB8 B3A0A3C2 B0B0B0B9 [................]
D/000010: B0                                  [.               ]

File ............. "ASMIDSTAMP"
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 BLDFTAB.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  BLDFTAB
; ##############################################################################

               PAGE
;
; BLDFTB - BUILD FILE TABLES
; TABLE MAP:
; HIMEM,SOP
; SBUFF N (256)
; DBUFF N (256)
; FTB N (FCBLEN)
; HEADER N (38)
; .
; .
; SBUFF 1
; DBUFF 1
; FTB 1
; HEADER 1
; THIS PROGRAM
;
; HEADER MAP:
; FILENAME (30)
; FTB PTR (2)
; DBUF PTR (2)
; SBUF PTR (2)
; LINK (2)
;
BLDFTB         EQU         *
               SEC
               LDA         FTAB                    ; START OF FTAB AREA
               STA         ZPGWRK                  ; IS 1ST FTB PTR
               LDA         FTAB+1                  ; HEADER
               STA         ZPGWRK+1
               LDA         CNFTBS                  ; MOVE NO FTABS
               STA         TEMP1A                  ; TO TEMP
;
BFT1           LDY         #0
               TYA
               STA         (ZPGWRK),Y              ; 1ST CHAR FN=0
               LDY         #30                     ; INC Y TO FCB PTR
               SEC
               LDA         ZPGWRK                  ; END OF PTR HEADER
               SBC         #FCBLEN                 ; MINUS FTAB LENGTH
               STA         (ZPGWRK),Y              ; IS START OF FTB
               PHA                                 ; SAVE LOW ADR BYTE
               LDA         ZPGWRK+1
               SBC         #0
               INY
               STA         (ZPGWRK),Y
               TAX
               DEX                                 ; FTB ADR - 256
               PLA                                 ; IS ADR DIR BUFF
               PHA
               INY
               STA         (ZPGWRK),Y              ; SET DIR BUF PTR
               TXA
```

```
                INY
                STA         (ZPGWRK),Y
                TAX
                DEX                                 ; DIR BUFF - 256
                PLA                                 ; IS SBUFF ADR
                PHA
                INY
                STA         (ZPGWRK),Y
                INY
                TXA
                STA         (ZPGWRK),Y
;
                DEC         TEMP1A                  ; DECREMENT TABLE INDEX
                BEQ         BFT2                    ; COUNT AND BR IF DONE
                TAX
                PLA
                SEC
                SBC         #38                     ; SBUFF ADR - 38
                INY
                STA         (ZPGWRK),Y              ; IF ADR OF NEXT TAB
                PHA                                 ; WHICH GOES INTO
                TXA                                 ; LINK
                SBC         #0
                INY
                STA         (ZPGWRK),Y
                STA         ZPGWRK+1                ; AND INTO ZPGWRK
                PLA                                 ; FOR NEXT ENTRY
                STA         ZPGWRK                  ; BUILD
                JMP         BFT1                    ; GO BUILD NEXT
;
BFT2            EQU         *
                PHA
                LDA         #0                      ; SET LAST LINK
                INY                                 ; TO ZERO
                STA         (ZPGWRK),Y
                INY
                STA         (ZPGWRK),Y
;
                LDA         ASIBSW                  ; IF IB THEN GO
                BEQ         BFTIB
;
                PLA                                 ; SET APPLESOFT
                STA         ASHM1+1                 ; UPPER MEM LIMITS
                STA         ASHM2+1
                PLA
                STA         ASHM1
                STA         ASHM2
                RTS
;
BFTIB           EQU         *
                PLA                                 ; SET IB
                STA         IBHMEM+1                ; UPPER MEM LIMITS
                STA         IBSOP+1
                PLA
                STA         IBHMEM
                STA         IBSOP
                RTS
                PAGE
;
; MVISW - MOVE INPUT SWITCH
;
MVCSW           EQU         *
                LDA         INSW+1
```

```
                CMP         CINA+1
                BEQ         MVOSW
                STA         SVINS+1
                LDA         INSW                    ; SAVE CHAR IN SWITCH
                STA         SVINS
;
                LDA         CINA                    ; SET DFB CHAR IN ADR
                STA         INSW
                LDA         CINA+1
                STA         INSW+1
;
;
; MVOSW - MOVE OUTPUT SWITCH
;
MVOSW           EQU         *
                LDA         OUTSW+1
                CMP         COUTA+1
                BEQ         MVSRTN
                STA         SVOUTS+1
                LDA         OUTSW                   ; SAVE CHAR OUT SWITCH
                STA         SVOUTS
;
                LDA         COUTA                   ; SET DFB CHAR OUT ADR
                STA         OUTSW
                LDA         COUTA+1
                STA         OUTSW+1
MVSRTN          EQU         *
                RTS

; ###############################################################################
; #   END OF FILE:  BLDFTAB
; #   LINES      :  144
; #   CHARACTERS :  4908
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT BLOCKIO.hex
================================================================================

File ............. "BLOCKIO"
Fork ............. DATA
Size (bytes) ..... 205 (0KB) / $000000CD
Created .......... Wednesday, April 12, 2006 -- 9:21:54 AM
Modified ......... Wednesday, April 12, 2006 -- 9:21:54 AM

D/000000: A94C8DF8 03A9118D F903A903 8DFA03D0 [.L..............]
D/000010: 2BADE9B7 8DB9038D C703ADEA B78DBA03 [+...............]
D/000020: 8DC803A2 00BD0002 C999F003 E8D0F6BD [................]
D/000030: 0102A002 C9D7F008 C9D2F003 4C65FF88 [............Le..]
D/000040: 8CC403A5 3C8DC003 A53D8DC1 03A53E85 [....<....=....>.]
D/000050: 0EA53F85 0FA5420A 290FA888 8CA70346 [..?...B.)......F]
D/000060: 43664266 426642A5 42293F8D BC03ACA7 [CfBfBfB.B)?.....]
D/000070: 03C8C010 D005EEBC 03A000B9 A8038DBD [................]
D/000080: 038CA703 A903A0B8 20D90390 06202DFF [............-.]
D/000090: 4C69FFE6 3DEEC103 ADC003C5 0EADC103 [Li..=...........]
D/0000A0: E50F90CA 4C69FFA0 000E0D0C 0B0A0908 [....Li..........]
D/0000B0: 07060504 0302010F 01A0A000 E0A0C903 [................]
D/0000C0: E8D2A0A0 F7A0A99E BD0001EF D8       [.............   ]

File ............. "BLOCKIO"
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 BOOTLDR.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  BOOTLDR
; ###########################################################################################

                SBTL         '16-SECTOR DOS BOOT'
HERE3L          EQU          >*
REMDR3          EQU          256-HERE3L
                ORG          *+REMDR3
TRK0LDR         EQU          *
                DO           >TRK0LDR
                ???                                  ;DELIBERATE ERROR IF NOT AT PAGE
BOUNDARY
                FIN
**************************
*                        *
* 16-SECTOR DOS BOOTSTRAP *
*                        *
*     RICK AURICCHIO     *
*       10/10/79         *
*                        *
**************************
*                        *
* THIS PROGRAM RESIDES IN *
*  TRACK 0, SECTOR 0 OF A *
*  DOS DISKETTE. ITS SOLE *
*  PURPOSE IS TO READ THE *
*  DOS LOADER PROGRAM IN  *
*  FROM TRACK 0, SECTORS  *
*  1-9. CONTROL IS THEN   *
*  TRANSFERRED TO THAT    *
*  PROGRAM.               *
*                        *
* NOTE: THE DOS LOADER    *
*  CONTAINS THE ENTIRE    *
*  SET OF 16-SECTOR CORE  *
*  ROUTINES; THOSE CORE   *
*  ROUTINES ARE USED TO   *
*  LOAD THE REST OF THE   *
*  DOS IMAGE INTO MEMORY. *
*                        *
**************************
POINTA          EQU          $26                     ;BUFFER POINTER
BSLOT           EQU          $2B                     ;BOOT BSLOT
BSECTR          EQU          $3D                     ;LAST BSECTR READ
BTEMP           EQU          $3E                     ;ADDRESS BTEMP
BRETRY          EQU          $5C                     ;OFFSET TO READER
*
MONINIT         EQU          $FB2F                   ;MONINIT SCREEN
BHERE3          EQU          >*
BOOTCNT         EQU          $800+BHERE3
                DFB          01
                PAGE
                LDA          POINTA+1                ;WHERE DID BSECTR GET LOADED?
                CMP          #09                     ; (AT 0800)?
                BNE          READNEXT                ;=>NO. WE'RE LOADING SOMETHING
* WE'VE BEEN BOOTED. SET UP
*  PARAMS FOR BOOT PROM SO
```

```
*  THAT WE'LL READ IN TRACK 0,
*  BSECTRS 00-09.
                LDA         BSLOT                   ;GET BOOT BSLOT
                LSR         A                       ;CONVERT TO CX00
                LSR         A
                LSR         A
                LSR         A
                ORA         #$C0
                STA         BTEMP+1
                LDA         #BRETRY                 ;PROM ROUTINE OFFSET
                STA         BTEMP
                CLC                                 ;BUMP LOAD ADDRESS UP TO
                LDA         LOADADDR+1              ; LAST PAGE SO WE
                ADC         BGRPGC                  ;  CAN LOAD 'EM BACKWARDS...
                STA         LOADADDR+1
* READ IN ANOTHER BSECTR FROM
*  TRACK ZERO...
READNEXT        LDX         BGRPGC
                BMI         GOLOADER                ;=>ALL DONE...EXECUTE IT!
                LDA         TABLE,X                 ;GET PHYSICAL BSECTR NUMBER
                STA         BSECTR                  ; AND SET FOR BOOT PROM READ
                DEC         BGRPGC                  ;ONE LESS BELL TO ANSWER..
                LDA         LOADADDR+1              ;GET LOAD ADDRESS
                STA         POINTA+1                ; FOR BSECTR READ
                DEC         LOADADDR+1              ;MOVE LOAD ADDRESS DOWN A PAGE
                LDX         BSLOT                   ;RESTORE BSLOT NUMBER
                JMP         (BTEMP)                 ;READ MORE OF TRACK 0
GOLOADER        INC         LOADADDR+1              ;ENTRY AT SECOND PAGE
                INC         LOADADDR+1
                JSR         SETKBD                  ;CLEAR IN#X
                JSR         SETVID                  ; AND PR#X
                JSR         MONINIT                 ;MONINIT THE SCREEN PARAMS
                LDX         BSLOT                   ;PASS BSLOT NBR TO LOADER
                JMP         (LOADADDR)              ;OFF TO LOOADER!
* TABLE OF PHYSICAL BSECTR NUMBERS
*  WHICH CORRESPOND TO THE LOGICAL
*  BSECTRS 0-F ON TRACK ZERO...
BHERE2          EQU         >*
TABLE           EQU         $800+BHERE2
                DFB         $00,13,11               ;00->00,01->13,02->11
                DFB         09,07,05                ;03->09,04->07;05->05
                DFB         03,01,14                ;06->03,07->01,08->14
                DFB         12,10,08                ;09->12,10->10,11->08
                DFB         06,04,02,15             ;12->6,13->04,14->02,15->15
                PAGE
                REP         40
* APPEND BUG PATCHES
****************************
SC3             EQU         *
EOFFLAG         DFB         0
CLOSFILE        EQU         *
                JSR         FILSRC                  ;FILE BUFFER FOUND?
                BCS         NOTFOUND                ;=> NO, SO SKIP IT.
                LDA         #0                      ;YES, CLOSE IT
                TAY
                STA         EOFFLAG
                STA         (ZPGWRK),Y              ;RIGHT NOW
NOTFOUND        LDA         CCBSTA                  ;ORIGINAL INSTRUCTION
                JMP         ERROR                   ;BACK TO ERROR HANDLER
****************************
BUMPER          EQU         *
                LDA         EOFFLAG                 ;SHOULD WE?
                BEQ         GOBACK                  ;=> NO
```

```
                INC           CCBRRN                    ;BUMP CCB RECORD NUMBER
                BNE           GOBACK
                INC           CCBRRN+1                  ;TO GET TO NEXT SECTOR
GOBACK          LDA           #0
                STA           EOFFLAG                   ;TURN FLAG OFF
                JMP           FIXIT2                    ;Go to FIXIT2 as exit
                REP           40
VPATCH          EQU           *
                STA           CCBRQM                    ;ORIGINAL INSTRUCTION
                JSR           DOSGO                     ;GO SAVE
                JSR           ECLOSE                    ;CLOSE THE FILE
                JMP           EVAR                      ;GO VERIFY IT AFTER SAVE
****************************
EOFFIX          EQU           *
                LDY           #$13                      ;PEEK INTO THE FCB: IF
CHKFILE         LDA           (ZPGFCB),Y                ;DCBCRS,DCBCSB ARE ZEROS,
                BNE           FIXIT                     ; THEN WE HAVE EMPTY FILE
                INY
                CPY           #$17
                BNE           CHKFILE
                LDY           #$19
MOVE            LDA           (ZPGFCB),Y                ; DCBCRR,DCBCRB
                STA           CCBRRN-$19,Y              ;INTO CCBRRN,CCBBYT
                INY
                CPY           #$1D
                BNE           MOVE
BACK            JMP           DOSGO2A                   ;NOW LET APPEND CONTINUE
FIXIT           EQU           *
                LDX           #$FF                      ;SET FLAG SO APPEND WILL
                STX           EOFFLAG                   ;KNOW TO CROSS SECTOR BOUNDARY
                BNE           BACK                      ;ALWAYS TAKEN
                PAGE
                REP           40
* END OF BOOT PAGE DATA SETUP
                REP           40
*
* FIXIT2 was developed to fix the wrap around
* problem APPEND has when trying to APPEND to
* a sequential file which is >255 sectors in length.
*
*                  Fix by
*
*               Fern Bachman
*                Guil Banks
*            September 28, 1982
*
* Fix to fix added to correctly APPEND to a sector
* 255 bytes in length
*
*                     by
*                 Guil Banks
*                July 11, 1983
*
                REP           30
                SKP           1
FIXIT2          EQU           *
                LDA           CCBRLN                    ;Current record length lo
                STA           DCBCSB                    ;Current sector byte
                STA           DCBCRR                    ;Current relative record
                LDA           CCBRLN+1                  ;Do hi as well
                STA           DCBCSB+1
                STA           DCBCRR+1
                STA           DCBCRS                    ;Set current relative sector
```

```
                TSX
                STX         ENTSTK
                JMP         GOODIO
                SKP         2
                REP         30
*
*   Upper/Lower case patch
*   for DOS 3.3C and BASIC
*
*           by
*        Guil Banks
*        Mark Houde
*
                REP         30
*
* This routine converts all characters
* that are not between quotes to
* upper case and returns them to the
* input buffer ($200). This works with
* DOS, Integer & Applesoft.
*
* Upon entry -
*
*              X Reg = 0
*
* Upon exit  -
*
*              Y Reg = $FF
*              ACCUM = $8D
*              X Reg = unknown
*
                REP         30
                DO          ULC
UPRCASE         EQU         *
LUP1            LDA         LBUFF,X                 ;Get a char
                CMP         #'"+$80                 ;Is it a quote?
                BNE         CHK4UC                  ;=> if not
LUP2            INX                                 ;Bump to next char
                LDA         LBUFF,X                 ;Get it
                CMP         #'"+$80                 ;Closing quote?
                BEQ         NEXTCHR                 ;=> if so
                CMP         #$8D                    ;End of line?
                BNE         LUP2                    ;=> if not
ULFINI          LDY         #$FF                    ;Do what DOS wants
                STY         CMDNO
                RTS                                 ;   & exit
CHK4UC          EQU         *
                CMP         #$E0                    ;Upper case?
                BCC         CHK4CR                  ;=> if not
                AND         #$DF                    ;Make upper case
                STA         LBUFF,X                 ;   & restore
CHK4CR          CMP         #$8D                    ;End of input?
                BEQ         ULFINI                  ;=> if so
NEXTCHR         INX                                 ;Bump to next char
                BNE         LUP1                    ;=> always
                FIN
                SKP         1
BHERE1          EQU         >*
                DS          $FD-BHERE1,0
* LOAD ADDRESS FOR CODE (PG BDY)
* ENTRY AFTER BOOT IS AT LOADADDR+256 (SECOND PAGE LOADED)
*
BHERE4          EQU         >*
```

```
LOADADDR        EQU         $800+BHERE4
                DFB         0
GRSPG           DFB         <TRK0LDR                    ;CONTAINS PAGE#-1 OF TRK 0, SEC 1
LOAD ADDRESS
* LAST LOGICAL BSECTOR TO READ STARTING AT $00
BHERE5          EQU         >*
BGRPGC          EQU         $800+BHERE5
GRPGC           DFB         <ENDOFDOS-TRK0LDR-$100
********************
                PAGE
DOSLODR         EQU         *
                DO          >DOSLODR
                ???                                     ;ERROR IF NOT ON PAGE BOUNDARY
                FIN
                REP         40
* FAST BOOT AT 2:1 INTERLEAVE
* FOR 16-SECTOR DISKETTES
                REP         40
                STX         IBSLOT                      ; SET BOOT SLOT
                STX         IBPSLT                      ; SET PREVIOUS SLOT
                LDA         #1                          ; SET PREV DRIVE
                STA         IBPDRV
                STA         IBDRVN
;
                LDA         NDPGS                       ; COPY NO PAGES TO GET
                STA         BRWCNT
                LDA         #2
                STA         IBTRK                       ; SET TRACK 0
                LDA         #4                          ;ENDING SECTOR OF DOS IMAGE
                STA         IBSECT                      ;TO IOB
                LDY         ADOSLD+1                    ;END PAGE OF DOS IMAGE
                DEY                                     ;IS ONE LESS THAN
                STY         IBBUFP+1                    ;START OF DOSLDR+BOOT
;
                LDA         #IBCRTS                     ; SET READ
                STA         IBCMD
;
                TXA                                     ; SET PREV TRACK = 0
                LSR         A
                LSR         A
                LSR         A
                LSR         A
                TAX
                LDA         #0
                STA         $4F8,X
                STA         $478,X
                JSR         BOOTIO                      ; GO READ DOS
;
;DOSINT - INITIALIZE DOS
;
DOSINT          EQU         *
                LDX         #$FF
                TXS
                STX         IBVOL
                JMP         RCPATCH
RCBACK          JSR         SETKBD
;
DI3             JMP         DOSREL                      ; GO TO POST INIT ROUTINE
                PAGE
WBOOT           EQU         *
                LDA         ADOSLD+1
                SEC
                SBC         IBBUFP+1
```

```
                STA         BRWCNT                  ;COMPUT PAGE COUNT
                LDA         ADOSLD+1
                STA         IBBUFP+1                ;BUFFER=LAST PAGE OF RWTS
                DEC         IBBUFP+1
                LDA         #2                      ;ENDING TRACK
                STA         IBTRK
                LDA         #4
                STA         IBSECT                  ;ENDING SECTOR
                LDA         #2
                STA         IBCMD                   ;COMMAND = WRITE
                JSR         BOOTIO                  ;WRITE DOS IMAGE TRK 2,SEC 4
*   ;           BACKWARDS TO TRK 0,SEC C
                LDA         ADOSLD+1
                STA         GRSPG                   ;BOOTSTRAP LOAD ADDRESS
                CLC
                ADC         #9
                STA         IBBUFP+1                ;BUFFER ADDRESS OF END OF BOOT
                LDA         #10
                STA         BRWCNT                  ;SECTOR COUNT TO WRITE
                SEC
                SBC         #1
                STA         GRPGC                   ;BOOT LAST SECTOR #
                STA         IBSECT                  ;START AT END OF RWTS&BOOT
                JSR         BOOTIO                  ;AND WRITE DOWN TO ZERO
                RTS
                DS          6,0                     ;FILL WITH BRKS
                PAGE
BOOTIO          EQU         *
                LDA         BAIOB+1
                LDY         BAIOB
                JSR         DISKIO
;
                LDY         IBSECT                  ; GET SECTOR
                DEY                                 ; DECREMENT TO NEXT
                BPL         BIO1                    ;AT END OF TRACK?
                LDY         #15                     ;SET TO SECTOR 15
                NOP
                NOP
                DEC         IBTRK
BIO1            STY         IBSECT                  ; SET NEXT SECTOR
;
                DEC         IBBUFP+1                ; DECREMENT BUFFER POINTER
                DEC         BRWCNT                  ; DECREMENT PAGE COUNTER
                BNE         BOOTIO                  ; BR IF NOT DONE
                RTS
;
                PAGE
DISKIO          PHP                                 ;SAVE INTERUPT STATUS
                SEI                                 ;INHIBIT INTERUPT WHILE
                JSR         RWTS                    ; ACCESSING DISK
                BCS         DSKERR                  ;MUST PASS BACK CARRY FLAG &
INTERUPT
                PLP
                CLC
                RTS
DSKERR          PLP                                 ;CARRY SET MEANS ERROR
                SEC
                RTS
DLDSUP          LDA         CCBBSA                  ;SET UP FOR DOS LOADER
                STA         IBBUFP+1                ;START ADDRESS
                LDA         #0
                STA         IBBUFP
                LDA         DCBVOL                  ;INVERT VOLUME NUMBER
```

```
                EOR         #$FF
                STA         IBVOL
                RTS
;
CLRSEC          LDA         #0                      ;CLEAR SECTOR
                TAY
CS1             STA         (ZPGFCB),Y
                INY
                BNE         CS1
                RTS
                BRK
EC3             EQU         *
NDPGS           DFB         <TRK0LDR-BEGIN          ;CALC #PAGES IN DOS WITHOUT RWTS
BRWCNT          DFB         0                       ;WRK CTR FOR BOOTIO
                DFB         $0A
                DFB         $1B
BAIOB           DW          IOB
ADOSLD          DW          TRK0LDR
                PAGE
;
;IOB - INPUT / OUTPUT CONTROL BLOCK
;THE IOB IS USED FOR THE INTERFACE
;BETWEEN DOS AND THE DISK I/O ROUTINES
;
IOB             EQU         *
IBTYPE          DFB         1                       ; IOB TYPE CODE
IBSLOT          DFB         6*16                    ; CONTROLLER SLOT NO.
IBDRVN          DFB         1                       ; DRIVE NUMBER
IBVOL           DFB         $00                     ; VOLUME NUMBER
IBTRK           DFB         0                       ; TRACK NUMBER
IBSECT          DFB         0                       ; SECTOR NUMBER
IBDCTP          DW          DCT
IBBUFP          DW          0                       ; POINTER TO BUFFER
IBDLEN          DW          256                     ; DATA LENGTH
IBCMD           DFB         0                       ; COMMAND
IBCNUL          EQU         0                       ; 0-NULL COMMAND
IBCRTS          EQU         1                       ; 1-READ TRACK, SECTOR
IBCWTS          EQU         2                       ; 2-WRITE TRACK, SECTOR
IBFMT           EQU         4                       ; 4-FORMAT DISK
IBBOOT          EQU         8                       ; 8-WRITE BOOT
IBSTAT          DFB         0                       ; STATUS
IBRERR          EQU         $80                     ; READ ERR
IBDERR          EQU         $40                     ; DRIVE ERR
IBVMME          EQU         $20                     ; VOLUME MISMATCH
IBWPER          EQU         $10                     ; WRITE PROTECT ERROR
IBSMOD          DFB         0                       ; STATUS MODIFIER BYTE
IBPSLT          DFB         6*16                    ; PREVIOUS SLOT
IBPDRV          DFB         1                       ; PREVIOUS DRIVE
IBSPAR          DS          2,0                     ; IOB SPARES
DCT             DFB         0,1,$EF,$D8
                DS          1,0                     ;FILL IN 3700 PAGE
                PAGE
;
;FILE DIRECTORY DEFINITION
;
                DSECT
FILDIR          EQU         *
FDUCDE          DS          1                       ; FILE USE CODE
FDLTRK          DS          1                       ; LINK TO NEXT DIR TRACK
FDLSEC          DS          1                       ; LINT TO NEXT DIR SECTOR
FDNSA           DS          1                       ; NO SECTORS ALLOCATED
FDLSDL          DS          1                       ; LAST SECTOR DATA LENGTH
FDFRS           DS          2                       ; 1ST RELATIVE SECTOR IN THIS DIR
```

```
FDSPAR          DS          5                       ; SPARES
;
FDENT           DS          1                       ; START OF FILE ENTRIES (122)
FDTRK           EQU         0                       ; TRACK
FDSEC           EQU         1                       ; SECTOR
;
FDLAST          EQU         FILDIR+256
                DEND

; ###########################################################################################
; #   END OF FILE:  BOOTLDR
; #   LINES      :  436
; #   CHARACTERS :  19320
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT CMDSCAN.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  CMDSCAN
; ###########################################################################################

               PAGE
;
; SCNCMD - SCAN A COMMAND
;
SCNCMD         EQU         *
               DO          ULC
               LDX         #0                      ;Init X for lower case check
               JSR         UPRCASE                 ;Go check
               ELSE
               LDY         #$FF                    ;Set CMDNO to -1
               STY         CMDNO
               FIN
               INY                                 ; INCR TABLE INDEX
               STY         SVCMD
SC0            EQU         *
               INC         CMDNO                   ; INCR CMD NO
               LDX         #0                      ; RESET LINE INDEX TO 0
               PHP                                 ; SAVE EQ STATUS
               LDA         LBUFF,X                 ; GET 1ST LINE CHAR
               CMP         CCHAR                   ; IS IT CONTROL D
               BNE         SC0A                    ; BR /IF NOT
               INX                                 ; INCR OVER CNTLD
SC0A           STX         LBUFD
;
SC1X           EQU         *
               JSR         GNBC                    ; GET NON BLANK INPUT CHAR
               AND         #$7F                    ; MSB OF CHAR OFF
               EOR         CMDNTB,Y                ; EOR WITH INPUT
               INY                                 ; INCREMENT TABLE INDEX
               ASL         A                       ; IF MSB OF EOR RESULT ON
               BEQ         SC1A                    ; IF RESULT NOT NOW ZERO
               PLA                                 ; THEN INPUT DOES NOT
               PHP                                 ; EQUAL ENTRY
SC1A           BCC         SC1X                    ; LOOP FOR END OF ENTRY
;
               PLP                                 ; IF INPUT EQUALS END
               BEQ         SYNTAX                  ; THEN GO SYNTAX
;
               LDA         CMDNTB,Y                ; IF NEXT TABLE CHAR NOT ZERO
               BNE         SC0                     ; THENSCANTHE NEXT TABLE ENTRY
CNF            EQU         *                       ; COMMAND NOT FOUND
               LDA         LBUFF                   ; LINE IS A CNOTROL-D
               CMP         CCHAR                   ; THEN THIS IS A
               BEQ         CNF1                    ; POSSIBLE SYNTAX ERROR, ELSE
               JMP         PRRTN                   ; ITS A BASIC INPUT LINE
CNF1           EQU         *
               LDA         LBUFF+1                 ; GET NEXT CHAR
               CMP         #$8D                    ; IS IT A CR
               BNE         CSERR                   ; BR IF CR
               JSR         CLRSTS                  ; CLEAR THE STATES
               JMP         CMDRTN                  ; CNTL-D ONLY
;
CSERR          JMP         ESYNTX
```

```
                PAGE
;
; SYNTAX - FIGURE OUT WHAT WE GOT HERE
;
SYNTAX          EQU         *
                ASL         CMDNO                   ; CMDNO=CMDNO*2
                LDY         CMDNO
                JSR         TSTRUN                  ;TEST FOR COMMANDS THAT MAY
                BCC         SYN1                    ; NOT BE EXECUTED DIRECTLY
                LDA         #RNONLY                 ;GET RUN ONLY FLAG
                AND         CMDSTB,Y                ;IF FLAG=0 THEN
                BEQ         SYN1                    ; OK TO EXECUTE
                LDA         #$F                     ;OTHERWISE, GIVE "NOT DIRECT"
                JMP         ERROR                   ; MESSAGE.
SYN1            CPY         #6                      ;TEST FOR 'RUN' COMMAND
                BNE         SYN1A
                STY         PROMPT                  ;CHANGE PROMPT TO INSURE APPLESOFT
RUN DETECT.
SYN1A           LDA         #FN1
                AND         CMDSTB,Y                ; IS FN1 REQD
                BEQ         SN10                    ; BR IF NOT
                JSR         CLRFNS
                PHP                                 ; SAVE EQ STATUS
;
SN2             EQU         *
                JSR         GNBC                    ; GET NON BLANK CHAR
                BEQ         SN6                     ; BR IF CR OR COMMA
                ASL         A                       ; TEST FOR ALPHA
                BCC         SN2A                    ; BR IF ALPHA
                BMI         SN2A                    ; BR IF APLHA
                JMP         CNF                     ; LURCH IF NOT ALPHA
SN2A            ROR         A                       ; RESTORE BITS
                JMP         SN4                     ; AWAY WE GO
SN3             JSR         GNXTC                   ; GO GET NEXT CHAR
                BEQ         SN6                     ; BR IF COMMA OR CR
SN4             STA         FNAME1,Y                ; PUT INTO FILENAME
                INY                                 ; INC FN INDEX
                CPY         #60                     ; ATFN CHAR LIMIT
                BCC         SN3                     ; BR IF NOT
SN5             JSR         GNXTC                   ; LOOP UNTIL CR OR COMMA
                BNE         SN5
;
SN6             PLP                                 ; WAS THIS FN2 L OO
                BNE         SN7                     ; BR IF IT WAS
;
                LDY         CMDNO
                LDA         #FN2
                AND         CMDSTB,Y                ; IF FN2 NOT REQD THEN
                BEQ         SN8                     ; BRANCH
;
                LDY         #30                     ; SET FN2 INDEX
                PHP                                 ; INDICATE FN2 SEEK
                BNE         SN2                     ; GO LOOK FOR FN2
;
SN7             LDA         FNAME2                  ; IF 1ST CHAR OF
                CMP         #$A0                    ; FN2 IS BLANK THEN
                BEQ         SERR1                   ; SYNTAX ERROR
;
SN8             LDA         FNAME1                  ; IF 1ST CHAR OF
                CMP         #$A0                    ; FN1 IS NOT BLANK
                BNE         SOPTS                   ; THEN GO LOOK FOR OPTIONS
;
                LDY         CMDNO
```

```
                LDA         #NPB+NPE                ; IF CMD MUST HAVE FILENAME
                AND         CMDSTB,Y                ; THEN
                BEQ         SERR1                   ; THIS IS ERROR, ELSE
;
                BPL         SOPTS                   ; ITS EXCUTABLE WITHOUT
;
SERR1           JMP         CNF
;
CLRFNS          EQU         *
                LDY         #60
CLRFNA          EQU         *
                LDA         #$A0
SN1             STA         FNAME1-1,Y              ; CLEAR FN1, FN2
                DEY
                BNE         SN1
                RTS
                PAGE
SN10            EQU         *                       ; FILE NAMES NOT REQD
                STA         FNAME1
                LDA         #NUM1+NUM2              ; IF NEITHER NUM1
                AND         CMDSTB,Y                ; OR NUM2 IS REQD
                BEQ         SOPTS                   ; THEN GO LOOK AT OPTIONS
;
                JSR         GETNUM                  ; GO GET NUMERICS
                BCS         SERR2
;
                TAY                                 ; IF HIGH DIGIT NOT
                BNE         SERR3                   ; ZERO THEN BAD
;
                CPX         #17                     ; IF LOW DIGIT GT 16
                BCS         SERR3                   ; THEN BAD
;
                LDY         CMDNO
                LDA         #NUM1
                AND         CMDSTB,Y                ; IF WE WANT NUM2
                BEQ         SN11
;
                CPX         #8                      ; IF NUM2>1
                BCS         SERR1                   ; THEN ERROR, ELSE
                BCC         SOPTS                   ; GO SCAN OPTIONS
;
SN11            EQU         *
                TXA                                 ; IF NUM1=0
                BNE         SOPTS                   ; THEN ERROR, ELSE GET OPTIONS
;
SERR3           LDA         #2                      ;RANGE ERROR!
                JMP         ERROR
SERR2           JMP         ESYNTX                  ;DISK CMD SYNTAX ERROR
;
                PAGE
;
; SOPTS - LOOK FOR SYNTAX OPTIONS
;
SOPTS           EQU         *
                LDA         #0
                STA         INOPTS                  ; CLEAR INPUT OPTIONS
                STA         IMBITS
                STA         CV                      ;DEFAULT VOLUME=0
                STA         CL
                STA         CL+1
                JSR         CLRBYTE                 ;PATCH FOR BYTE PARAMETER (WAS STA
TEMP1A)
                LDA         LBUFD                   ; SET PASS 1
```

```
;
SP1             JSR         GNBC                    ; GO GET NON-BLANK CHAR
                BNE         SP2                     ; BR IF NOT COMMA OR CR
                CMP         #$8D                    ; IF CHAR IS COMMA
                BNE         SP1                     ; THEN GO GET CHAR
;
                LDX         CMDNO                   ; OPTIONS INPUT = I
                LDA         INOPTS                  ; ALLOW OPTS = A
                ORA         CMDSTB+1,X              ; IF (A OR I)
                EOR         CMDSTB+1,X              XOR A NOT = 0 THEN
                BNE         SERR1                   ; WE HAVE UNALLOWED OPTIONS
;
                LDX         TEMP1A                  ; IF THIS IS PASS 2
                BEQ         CMDGO                   ; THEN DONE,
                STA         TEMP1A                  ; ELSE SET PASS
                STX         LBUFD                   ; RESTORE LBUFD AND
                BNE         SP1                     ; GO DO PASS 2
;
SP2             LDX         #OPT1L                  ; COMPARE CHAR HAVE WITH
SP3             CMP         OPTAB1-1,X              ; CHARS IN OPT TABLE
                BEQ         SP4                     ; IF FOUND CONTINUE,
                DEX
                BNE         SP3                     ; IF NOT FOUND
SERR2A          BEQ         SERR2                   ; THEN SYNTAX ERROR
;
SP4             LDA         OPTAB2-1,X              ; IF CORRESPONDING OP TAB 2 IS
                BMI         SP8                     ; MINUS THEN IT MONITOR BITS
                ORA         INOPTS
                STA         INOPTS
                DEX
;
                STX         TEMP2A                  ; ELSE A NUMERIC MUST FOLLOW
                JSR         GETNUM                  ; FOLLOW
                BCS         SERR2
;
                LDA         TEMP2A                  ; GET IOTION NUMBER
                ASL         A                       ; MULT BY 4
                ASL         A
                TAY
;
                LDA         CNUM+1                  ; IF RESULT NUM HI IS
                BNE         SP5                     ; GT 0, THEN GT LOW RANGE
                LDA         CNUM                    ; TEST RESULT LOW
                CMP         OPTAB3,Y                ; WITH LOW RANGE (LOW)
                BCC         SERR3                   ; BR IF RESULT < LR
                LDA         CNUM+1
SP5             CMP         OPTAB3+3,Y
                BCC         SP6                     ; BR IF LESS
SERR3A          BNE         SERR3                   ; BR IF GREATER
                LDA         CNUM
                CMP         OPTAB3+2,Y
                BCC         SP6                     ; BR IF LESS
                BNE         SERR3A                  ; BR IF GREATER
;
SP6             LDA         TEMP1A                  ; IF PASS 1, THEN
                BNE         SP1                     ; DONT STORE RESULT
                TYA
                LSR         A
                TAY
;
                LDA         CNUM+1                  ; STORE THE RESULT
                STA         CUROPT+1,Y
                LDA         CNUM
```

```
                STA         CUROPT,Y
SP7             JMP         SP1                     ; GO FOR NEXT OPT
;
SP8             EQU         *                       ; MONITOR REQ
                PHA                                 ; SAVE TYPE REQ
                LDA         #CIO                    ; SET OPTION OF CIO
                ORA         INOPTS
                STA         INOPTS
                PLA                                 ; RESTOERE REQ
                AND         #$7F                    ; CLEAR CIO
                ORA         IMBITS                  ; OR WITH PREV IMBITS
                STA         IMBITS
                BNE         SP7                     ; GO FOR NEXT
;
                BEQ         SERR2A                  ;BRANCH ALWAYS
;
CMDGO           JSR         CMDGO1                  ; CMDGO - EXECUTE COMMAND
                JMP         CERTN
;
CMDGO1          EQU         *
                JSR         CLRSTS
                JSR         CLRCCB                  ; GO CLEAR CCB
ECMD            EQU         *
                LDA         CMDNO                   ; COMMAND NO
                TAX                                 ; IS CMD EXEC TAB INDEX
                LDA         CMDETB+1,X              ; GET CMD ADR
                PHA                                 ; ONTO STACK
                LDA         CMDETB,X
                PHA
                RTS                                 ; AND GOTO COMMAND

; ##############################################################################
; #   END OF FILE:  CMDSCAN
; #   LINES      :  270
; #   CHARACTERS :  13581
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT CMDTBLS.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  CMDTBLS
; ###########################################################################################

                PAGE
;
; COMMAND NAME TABLE
;
EC1             EQU         *
CMDNTB          EQU         *
                DCI         "INIT"
                DCI         "LOAD"
                DCI         "SAVE"
                DCI         "RUN"
                DCI         "CHAIN"
                DCI         "DELETE"
                DCI         "LOCK"
                DCI         "UNLOCK"
                DCI         "CLOSE"
                DCI         "READ"
                DCI         "EXEC"
                DCI         "WRITE"
                DCI         "POSITION"
                DCI         "OPEN"
                DCI         "APPEND"
                DCI         "RENAME"
                DCI         "CATALOG"
                DCI         "MON"
                DCI         "NOMON"
                DCI         "PR#"
                DCI         "IN#"
                DCI         "MAXFILES"
                DCI         "FP"
                DCI         "INT"
                DCI         "BSAVE"
                DCI         "BLOAD"
                DCI         "BRUN"
                DCI         "VERIFY"
                DFB         0
                PAGE
;
; COMMAND SYNTAX OP EQUATES FOR SYNTAX BYTE ONE
;
NPB             EQU         $80                     ; NO PARMS OK, COMMAND GOES TO BASIC
NPE             EQU         $40                     ; NO PARMS OK, COMMAND TO EXECUTION
RTN
FN1             EQU         $20                     ; FILE NAME1 REGD
FN2             EQU         $10                     ; FILE NAME2 REQD
NUM1            EQU         $08                     ; NUMERIC 0-7 REGD
NUM2            EQU         $04                     ; NUMERIC 1-10 REQD
RNONLY          EQU         $02                     ; RUN TIME ONLY FLAG.
CREFLG          EQU         $01                     ; FLAG TO INDICATE CMDS THAT MAY
CREATE FILES
;
; COMMAND SYNTAX OP EQUATES FOR SYNTAX BYTE TWO
;
V               EQU         $40                     ; VOLUME ALLOWED
```

```
D               EQU         $20                     ; DRIVE ALLOWED
S               EQU         $10                     ; SLOT ALLOWED
L               EQU         $08                     ; LENGTH ALLOWED
R               EQU         $04                     ; RECORD NUMBER ALLOWED
B               EQU         $02                     ; BYTE NUMBER ALLOWED
ADR             EQU         $01                     ; ADDRESS
CIO             EQU         $80                     ; C,I, OR O ALLOWED
;
; COMMAND SYNTAX TABLE
; EACH COMMAND HAS TWO BYTE ENTRY
;
CMDSTB          EQU         *
                DFB         FN1+CREFLG,V+D+S        ; INIT
                DFB         NPB+FN1,V+D+S           ; LOAD
                DFB         NPB+FN1+CREFLG,V+D+S    ; SAVE
                DFB         NPB+FN1,V+D+S           ; RUN
                DFB         FN1,V+D+S               ; CHAIN
                DFB         FN1,V+D+S               ; DELETE
                DFB         FN1,V+D+S               ; LOCK
                DFB         FN1,V+D+S               ; UNLOCK
                DFB         NPE+FN1,0               ; CLOSE
                DFB         FN1+RNONLY,B+R          ; READ
                DFB         FN1,R+V+D+S             ; EXEC
                DFB         FN1+RNONLY,B+R          ; WRITE
                DFB         FN1+RNONLY,R            ; POSITION
                DFB         FN1+RNONLY+CREFLG,L+V+D+S ; OPEN
                DFB         FN1+RNONLY,V+D+S        ; APPEND
                DFB         FN1+FN2,V+D+S           ; RENAME
                DFB         NPE,V+D+S               ; CATALOG
                DFB         NPE,CIO                 ; MONITOR
                DFB         NPE,CIO                 ; NO MONITOR
                DFB         NUM1,0                  ; PR#
                DFB         NUM1,0                  ; IN#
                DFB         NUM2,0                  ; MAXFILES
                DFB         NPE,V+D+S               ; APPLESOFT
                DFB         NPE,0                   ; INT
                DFB         FN1+CREFLG,V+D+S+ADR+L  ; BSAVE
                DFB         FN1,V+D+S+ADR           ; BLOAD
                DFB         FN1,V+D+S+ADR           ; BRUN
                DFB         FN1,V+D+S               ; VERIFY
                PAGE
;
; OPTAB - OPTIONAL PARMS SYNTAX TABLES
;
OPTAB1          EQU         *
                DFB         'V'+$80,'D'+$80,'S'+$80,'L'+$80'
                DFB         'R'+$80,'B'+$80,'A'+$80,'C'+$80'
                DFB         'I'+$80,'O'+$80'
OPT1L           EQU         *-OPTAB1
MI              EQU         $20
MO              EQU         $10
OPTAB2          EQU         *
                DFB         V,D,S,L
                DFB         R,B,ADR,CIO+MC
                DFB         CIO+MI,CIO+MO
OPTAB3          EQU         *
                DW          0
                DW          254                     ; VOL RANGE
                DW          1
                DW          2                       ; DRIVE RANGE
                DW          1
                DW          7                       ; SLOT RANGE
                DW          1
```

```
                DW          32767                       ; LENGTH RANGE
                DW          0
                DW          32767                       ; REC NO RANGE
                DW          0
                DW          32767                       ; REC BYTE NO RANGE
                DW          0
                DW          $FFFF                       ; ADDRESS RANGE
                PAGE
;
; ERROR MESSAGE TABLES
;
EMSG            EQU         *
                DFB         $0D,$07,$8D
EM1             EQU         *-EMSG
                DCI         "LANGUAGE NOT AVAILABLE"
EM2             EQU         *-EMSG
EM3             EQU         *-EMSG
                DCI         "RANGE ERROR"
EM4             EQU         *-EMSG
                DCI         "WRITE PROTECTED"
EM5             EQU         *-EMSG
                DCI         "END OF DATA"
EM6             EQU         *-EMSG
                DCI         "FILE NOT FOUND"
EM7             EQU         *-EMSG
                DCI         "VOLUME MISMATCH"
EM8             EQU         *-EMSG
                DCI         "I/O ERROR"
EM9             EQU         *-EMSG
                DCI         "DISK FULL"
EM10            EQU         *-EMSG
                DCI         "FILE LOCKED"
EM11            EQU         *-EMSG
                DCI         "SYNTAX ERROR"
EM12            EQU         *-EMSG
                DCI         "NO BUFFERS AVAILABLE"
EM13            EQU         *-EMSG
                DCI         "FILE TYPE MISMATCH"
EM14            EQU         *-EMSG
                DCI         "PROGRAM TOO LARGE"
;
EM15            EQU         *-EMSG
                DCI         "NOT DIRECT COMMAND"
                DFB         $8D
EMDTB           EQU         *
                DFB         0,EM1,EM2,EM3
                DFB         EM4,EM5,EM6,EM7
                DFB         EM8,EM9,EM10,EM11
                DFB         EM12,EM13,EM14
                DFB         EM15

; ##############################################################################
; #   END OF FILE:  CMDTBLS
; #   LINES      :  164
; #   CHARACTERS :  7084
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT COREQUS.pretty
================================================================================

; ###############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  COREQUS
; ###############################################################################

                SBTL         '16-SECTOR CORE ROUTINES'
**************************
*                        *
*         DISC-II        *
*     16-SECTOR FORMAT   *
*      READ AND WRITE    *
*       SUBROUTINES      *
*                        *
**************************
*                        *
*                        *
*     COPYRIGHT 1979     *
*   APPLE COMPUTER INC.  *
*                        *
*   ALL RIGHTS RESERVED  *
*                        *
**************************
*                        *
*      MAR 18, 1979      *
*          WOZ           *
*                        *
**************************
ASC1            EQU          *                        ;TELL RELOCATOR WHERE CORE STARTS
                PAGE
**************************
*                        *
*     CRITICAL TIMING    *
*   REQUIRES PAGE BOUND  *
*   CONSIDERATIONS FOR   *
*      CODE AND DATA     *
*                        *
*     -----CODE-----     *
*                        *
*   VIRTUALLY THE ENTIRE *
*    'WRITE16' ROUTINE   *
*      MUST NOT CROSS    *
*     PAGE BOUNDARIES.   *
*                        *
*  THE WRITE16, READ16   *
*  AND RDADR16 SUBRS     *
*  WHICH MUST NOT CROSS  *
*  PAGE BOUNDARIES ARE   *
*  NOTED IN COMMENTS.    *
*                        *
*     -----DATA-----     *
*                        *
*  NBUF1 AND NBUF2 ARE   *
*  256-BYTE AND 86-BYTE  *
*  NIBL BUFFERS IN RAM.  *
*  BOTH MUST BEGIN ON    *
*  PAGE BOUNDARIES.      *
*                        *
*  NIBLIZING TABLE 'NIBL' *
```

```
*  (64 BYTES) MAPS 6-BIT  *
*  NIBLS INTO VALID 7-BIT *
*  NIBLS.  THIS TABLE     *
*  MUST NOT CROSS A PAGE  *
*  BOUNDARY.              *
*                         *
*  DENIBLIZING TABLE      *
*  'DNIBL' MAPS 7-BIT     *
*  NIBLS INTO 6-BIT       *
*  NIBLS.  IT MUST BEGIN  *
*  ON A PAGE BOUNDARY,    *
*  BUT ONLY DNIBL,$96 TO  *
*  DNIBL,$FF ARE USED.    *
*                         *
***************************
               PAGE
***************************
*                         *
*         EQUATES         *
*                         *
***************************
*                         *
*    ----PRENIBL16----    *
*      AND POSTNB16       *
*   (16-SECTOR FORMAT)    *
*                         *
***************************
BUF             EQU          $3E                       TWO BYTE POINTER.
*
* POINTS TO 256-BYTE
* USER BUFFER ANYWHERE
* IN MEMORY.  PRENIBL16
* CONVERTS USER DATA
* (IN BUF) INTO 6-BIT
* NIBLS 00ABCDEF IN
* NBUF1 AND NBUF2 PRIOR
* TO 'WRITE'.  POSTNBL16
* CONVERTS 6-BIT NIBLS
* 00ABCDEF BACK TO USER
* DATA IN BUF AFTER 'READ'.
*
T0              EQU          $26                       TEMP FOR POSTNBL16.
************************
*                      *
*    ----RDADR16----   *
*                      *
************************
COUNT           EQU          $26                       'MUST FIND' COUNT.
LAST            EQU          $26                       'ODD BIT' NIBLS.
CSUM            EQU          $27                       CHECKSUM BYTE.
CSSTV           EQU          $2C                       FOUR BYTES,
*       CHECKSUM, SECTOR, TRACK, AND VOLUME.
*
************************
*                      *
*    ---WRITE16---     *
*                      *
*  USES NBUF1, NBUF2,  *
*  AND 64-BYTE TABLE   *
*       'NIBL'.        *
*                      *
************************
WTEMP           EQU          $26                       TEMP FOR DATA AT NBUF2,0.
```

```
SLOTZ           EQU         $27                     SLOTNUM IN Z-PAG LOC.
SLOTABS         EQU         $678                    SLOTNUM IN NON-ZPAG LOC.
*
************************
*                      *
*     ----READ16---    *
*  (16-SECTOR FORMAT)  *
*                      *
*   USES NBUF1,NBUF2.  *
*  USES LAST 106 BYTES *
*  OF A DATA PAGE FOR  *
*  SIGNIFICANT BYTES   *
*  OF DNIBL16 TABLE.   *
*                      *
************************
IDX             EQU         $26                     INDEX INTO (BUF).
*
************************
*                      *
*    ---- SEEK ----    *
*                      *
************************
TRKCNT          EQU         $26                     HALFTRKS MOVED COUNT.
PRIOR           EQU         $27                     PRIOR HALFTRACK.
TRKN            EQU         $2A                     DESIRED TRACK.
SLOTTEMP        EQU         $2B                     SLOT NUM TIMES $10.
CURTRK          EQU         $478                    CURRENT TRACK ON ENTRY.
*
************************
*                      *
*   ---- MSWAIT ----   *
*                      *
************************
MONTIMEL        EQU         $46                     MOTOR-ON TIME
MONTIMEH        EQU         $47                     COUNTERS.
*
************************
*                      *
*   ---- WRADR16 ----  *
*                      *
************************
AA              EQU         $3E                     ;TIMING CONSTANT
NSECT           EQU         $3F                     ;SECTOR NUMBER
NVOL            EQU         $41                     ;VOLUME NUMBER
TRK             EQU         $44                     ;TRACK NUMBER
*
************************
*                      *
*    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

; ###########################################################################################
; #   END OF FILE:  COREQUS
; #   LINES      :  174
; #   CHARACTERS :  5595
```

```
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT DOS.TO.DISK
================================================================================

MONCIO
CALL-151
BLOAD DOS33C.OBJ
6B00<3600.36FFM
6C00<1F00.1FFFM
6D00<1E00.1EFFM
6E00<1D00.1DFFM
6F00<1C00.1CFFM
7000<1B00.1BFFM
7100<3F00.3FFFM
7200<3E00.3EFFM
7300<3D00.3DFFM
7400<3C00.3CFFM
7500<3B00.3BFFM
7600<3A00.3AFFM
7700<3900.39FFM
7800<3800.38FFM
7900<3700.37FFM
7A00<2000.20FFM
7B00<2100.21FFM
7C00<2F00.2FFFM
7D00<2E00.2EFFM
7E00<2D00.2DFFM
7F00<2C00.2CFFM
8000<2B00.2BFFM
8100<2A00.2AFFM
8200<2900.29FFM
8300<2800.28FFM
8400<2700.27FFM
8500<2600.26FFM
8600<2500.25FFM
8700<2400.24FFM
8800<2300.23FFM
8900<2200.22FFM
8A00<3000.30FFM
8B00<3100.31FFM
8C00:0
8C01<8C00.8CFFM
8D00<3500.35FFM
8E00<3400.34FFM
8F00<3300.33FFM
9000<3200.32FFM
9100:0
9101<9000.91FFM
280:AD 0 C0 10 FB 2C 10 C0 60 A9 60 4C ED FD
3D0G
BLOADBLOCKIO
PRINT"INSERT A BLANK DISK IN DRIVE 1":PRINT"AND PRESS 'RETURN' ";:CALL649
CALL640
LOADHELLO,D2
INITHELLO,D1
CALL-151
300G
0<6B00.8CFF□W
15<8C00.91FF□W
3D0G
PRINT"DOS 3.3C NOW ON DISK"
```

```
================================================================================
DOCUMENT DOS33C.OBJ.hex
================================================================================

File ............. "DOS33C.OBJ"
Fork ............. DATA
Size (bytes) ..... 9,472 (9KB) / $00002500
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: 4C841DA9 BF8541A2 008640A0 00A14085 [L.....A...@...@.]
D/000010: 26984526 85269841 408140C5 26D005C8 [&.E&.&.A@.@.&...]
D/000020: D0EFF004 C641D0E3 A54129DF 85438642 [.....A...A)..C.B]
D/000030: A1424885 26984526 85269841 408142C5 [.BH.&.E&.&.A@.B.]
D/000040: 26D009C8 D0EFA443 684C511B 688142A4 [&......ChLQ.h.B.]
D/000050: 41C88C7D 1C3898ED 7E1C8D7C 1C38ED7A [A..}.8..~..|.8.z]
D/000060: 1CF09D8D 7F1CAD7A 1C8D0D1D A91D8D49 [.......z.......I]
D/000070: 37A9848D 4837A200 8640BD29 1CA8BD2A [7...H7...@.)...*]
D/000080: 1C85414C 931B18B1 406D7F1C 9140C8D0 [..AL....@m...@..]
D/000090: 02E641C8 D002E641 A541DD2C 1C90E798 [..A....A.A.,....]
D/0000A0: DD2B1C90 E18A1869 04AAEC28 1C90CBA2 [.+.....i...(....]
D/0000B0: 008E9C33 BD5A1C85 40BD5B1C 8541A200 [...3.Z..@.[..A..]
D/0000C0: A140208E F8A42FC0 02D011B1 40CD7A1C [.@..../.....@.z.]
D/0000D0: 900ACD7B 1CB0056D 7F1C9140 38A52F65 [...{...m...@8./e]
D/0000E0: 408540A9 00654185 41AE9C33 DD5D1C90 [@.@..eA.A..3.]..]
D/0000F0: CDA540DD 5C1C90C6 8A186904 AAEC591C [..@.\.....i...Y.]
D/000100: 90AFA93F 8541AC7D 1C888443 A9008540 [...?.A.}...C...@]
D/000110: 8542A8B1 409142C8 D0F9CE80 1CF006C6 [.B..@.B.........]
D/000120: 41C643D0 EE4C541E 24001D56 1D581D5A [A.C..LT.$..V.X.Z]
D/000130: 1D641D66 1D6C1D70 1D781D7C 1D7E1D80 [.d.f.l.p.x.|.~..]
D/000140: 1DC12AFD 2AE437E8 37EE37F0 37000000 [..*.*.7.7.7.7...]
D/000150: 00000000 00000000 0020841D 8428FD2A [.............(.*]
D/000160: 97335D36 E037563C DF3C0038 113A693A [.3]6.7V<.<.8.:i:]
D/000170: 843A003D A83FC83F FF3F1D40 00002300 [.:.=.?.?.?.@..#.]
D/000180: 23FCFCFC FCFCFCFD FCFCFFFC FDFCFCFF [#.............]
D/000190: FFFCFEFD FEFDFDFE FFFCFEFF FDFEFFFD [.............]
D/0001A0: FCFDFDFC FDFEFDFF FCFEFCFC FCFCFCFC [.............]
D/0001B0: FDFCFCFE FCFDFCFC FFFCFCFE FDFCFDFF [.............]
D/0001C0: FCFFFDFE FFFDFFFC FCFDFEFF FFFFFCFC [.............]
D/0001D0: FFFFFCFE FDFCFDFD FEFFFDFE FFFCFEFE [.............]
D/0001E0: FCFDFCFD FEFDFDFC FDFCFFFD FCFFFCFC [.............]
D/0001F0: FEFDFCFC FDFCFEFC FCFDFFFC FCFDFCFC [.............]
D/000200: D31C811E BD1E752A 932A602A 001BBB35 [......u*.*`*...5]
D/000210: EA1E111F 221F2E1F 511F601F 701F4E25 [....".".Q.`.p.N%]
D/000220: 12249623 D024EF24 62227022 7422E922 [.$.#.$.$b"p"t"."]
D/000230: 1A25C525 0F25DC25 A2229722 80226D25 [.%.%.%.%."."."m%]
D/000240: 32223C22 28222D22 50227925 9D253023 [2"<"("-"P"y%.%0#]
D/000250: 5C238D23 7C2236E8 E524E3E3 00E003E0 [\#.#|"6..$......]
D/000260: 000036E8 E524E3E3 00E003E0 FC24FC24 [..6..$.......$.$]
D/000270: 65D800E0 3CD4F2D4 06250625 6710841D [e...<....%.%g...]
D/000280: 3C0CF20C ADE9374A 4A4A4A8D 6A2AADEA [<.....7JJJJ.j*..]
D/000290: 378D682A AD00E049 20D0118D B62AA20A [7.h*...I.....*..]
D/0002A0: BD611D9D 551DCAD0 F74CBC1D A9408DB6 [.a..U....L...@..]
D/0002B0: 2AA20CBD 6B1D9D55 1DCAD0F7 38B012AD [*...k..U....8...]
D/0002C0: B62AD004 A920D005 0A1005A9 4C20B225 [.*.......L..%]
D/0002D0: 18082051 28A9008D 5E2A8D52 2A286A8D [...Q(...^*.R*(j.]
D/0002E0: 512A3003 6C5E1D6C 5C1D0A10 198DB62A [Q*0.l^.l\......*]
D/0002F0: A20CBD77 1D9D551D CAD0F7A2 1DBD932A [...w..U........*]
D/000300: 9D752ACA 10F7ADB1 2A8D572A 20D427AD [.u*.....*.W*..'.]
D/000310: B32AF009 48209D26 68A00091 40205B27 [.*..H..&h...@.[']
D/000320: AD5F2AD0 20A22FBD 511E9DD0 03CA10F7 [._*.../.Q.......]
D/000330: AD531E8D F30349A5 8DF403AD 521E8DF2 [.S....I.....R...]
```

```
D/000340: 03A906D0 05AD622A F0068D5F 2A4C8021 [......b*..._*L.!]
D/000350: 604CBF1D 4C841D4C FD2A4CB5 37AD0F1D [`L..L..L.*L.7...]
D/000360: AC0E1D60 ADC22AAC C12A604C 5128EAEA [...`..*..*`LQ(..]
D/000370: 4C59FA4C 65FF4C58 FF4C65FF 4C65FF65 [LY.Le.LX.Le.Le.e]
D/000380: FF20D11E AD512AF0 1548AD5C 2A912868 [.....Q*..H.\*.(h]
D/000390: 30034C26 2620EA1D A424A960 9128ADB3 [0.L&&....$.`.(..]
D/0003A0: 2AF00320 8226A903 8D522A20 BA1F20BA [*....&...R*.....]
D/0003B0: 1E8D5C2A 8E5A2A4C B31F6C38 0020D11E [..\*.Z*L..l8....]
D/0003C0: AD522A0A AABD111D 48BD101D 48AD5C2A [.R*.....H...H.\*]
D/0003D0: 608D5C2A 8E5A2A8C 5B2ABAE8 E88E592A [`.\*.Z*.[*....Y*]
D/0003E0: A203BD53 2A9536CA 10F860AE B72AF003 [...S*.6...`..*..]
D/0003F0: 4C781FAE 512AF008 C9BFF075 C533F027 [Lx..Q*.....u.3.']
D/000400: A2028E52 2ACDB22A D019CA8E 522ACA8E [...R*..*....R*..]
D/000410: 5D2AAE5D 2A9D0002 E88E5D2A C98DD075 []*.]*.....]*...u]
D/000420: 4CCD1FC9 8DD07DA2 008E522A 4CA41FA2 [L.....}...R*L...]
D/000430: 008E522A C98DF007 ADB32AF0 67D05E48 [..R*......*.g.^H]
D/000440: 38ADB32A D003205E 266890EC AE5A2A4C [8..*...^&h...Z*L]
D/000450: 151FC98D D005A905 8D522A20 0E264C99 [.........R*..&L.]
D/000460: 1FCDB22A F085C98A F0F1A204 8E522AD0 [...*.........R*.]
D/000470: E1A9008D 522AF025 A9008DB7 2A205128 [....R*.%....*.Q(]
D/000480: 4CDC24AD 0002CDB2 2AF00AA9 8D8D0002 [L.$.....*.......]
D/000490: A2008E5A 2AA940D0 06A910D0 02A9202D [...Z*.@........-]
D/0004A0: 5E2AF00F 20BA1F20 C51F8D5C 2A8C5B2A [^*.........\*.[*]
D/0004B0: 8E5A2A20 5128AE59 2A9AAD5C 2AAC5B2A [.Z*.Q(.Y*..\*.[*]
D/0004C0: AE5A2A38 606C3600 A98D4CC5 1FA0FF8C [.Z*8`l6...L.....]
D/0004D0: 5F2AC88C 622AEE5F 2AA20008 BD0002CD [_*..b*._*.......]
D/0004E0: B22AD001 E88E5D2A 20A42129 7F598428 [.*....]*..!).Y.(]
D/0004F0: C80AF002 680890F0 28F020B9 8428D0D6 [....h...(....(..]
D/000500: AD0002CD B22AF003 4CA41FAD 0102C98D [.....*..L.......]
D/000510: D006205B 274C951F 4CC4260E 5F2AAC5F [...['L..L.&._*._]
D/000520: 2A205E26 900CA902 390929F0 05A90F4C [*.^&....9.)....L]
D/000530: D226C006 D0028433 A9203909 29F06120 [.&.....3..9.).a.]
D/000540: 95200820 A421F01E 0A900530 034C0020 [.....!.....0.L..]
D/000550: 6A4C5920 209321F0 0D99752A C8C03C90 [jLY...!...u*..<.]
D/000560: F3209321 D0FB28D0 0FAC5F2A A9103909 [...!..(..._*..9.]
D/000570: 29F00CA0 1E08D0CB AD932AC9 A0F013AD [).........*.....]
D/000580: 752AC9A0 D04BAC5F 2AA9C039 0929F002 [u*...K._*..9.)..]
D/000590: 103F4C00 20A03CA9 A099742A 88D0FA60 [.?L...<...t*...`]
D/0005A0: 8D752AA9 0C390929 F02720B9 21B01FA8 [.u*..9.).'..!...]
D/0005B0: D017E011 B013AC5F 2AA90839 0929F006 [......._*..9.)..]
D/0005C0: E008B0CE 900B8AD0 08A9024C D2264CC4 [...........L.&L.]
D/0005D0: 26A9008D 652A8D74 2A8D662A 8D6C2A8D [&...e*.t*.f*.l*.]
D/0005E0: 6D2A20DC 3FAD5D2A 20A421D0 1FC98DD0 [m*..?.]*..!.....]
D/0005F0: F7AE5F2A AD652A1D 0A295D0A 29D093AE [..._*.e*..)].)...]
D/000600: 632AF076 8D632A8E 5D2AD0DC A20ADD40 [c*.v.c*.]*.....@]
D/000610: 29F005CA D0F8F0B6 BD4A2930 470D652A [).........J)0G.e*]
D/000620: 8D652ACA 8E642A20 B921B0A2 AD642A0A [.e*..d*..!...d*.]
D/000630: 0AA8A545 D009A544 D9552990 8CA545D9 [...E...D.U)...E.]
D/000640: 5829900B D083A544 D9572990 02D0F5AD [X).....D.W).....]
D/000650: 632AD094 984AA8A5 4599672A A5449966 [c*...J..E.g*.D.f]
D/000660: 2A4CE820 48A9800D 652A8D65 2A68297F [*L..H...e*.e*h).]
D/000670: 0D742A8D 742AD0E9 F09C2080 214C831F [.t*.t*.....!L..]
D/000680: 205B2720 AE21AD5F 2AAABD1F 1D48BD1E [.['..!._*....H..]
D/000690: 1D4860AE 5D2ABD00 02C98DF0 06E88E5D [.H`.]*.........]]
D/0006A0: 2AC9AC60 209321F0 FAC9A0F0 F760A900 [*..`..!......`..]
D/0006B0: A01699BA 3588D0FA 60A90085 44854520 [....5...`...D.E.]
D/0006C0: A42108C9 A4F03C28 4CCE2120 A421D006 [.!....<(L.!..!..]
D/0006D0: A644A545 186038E9 B03021C9 0AB01D20 [.D.E.`8..0!.....]
D/0006E0: FE216544 AAA90065 45A820FE 2120FE21 [.!eD...eE...!..!]
D/0006F0: 8A654485 44986545 854590CF 38600644 [.eD.D.eE.E..8`.D]
D/000700: 26456028 20A421F0 C538E9B0 30EEC90A [&E`(..!..8..0...]
D/000710: 9008E907 30E6C910 B0E2A204 20FE21CA [....0.........!.]
D/000720: D0FA0544 85444C04 22A5444C 95FEA544 [...D.DL.".DL...D]
```

```
D/000730: 4C8BFEAD 5E2A0D74 2A8D5E2A 602C742A [L...^*.t*.^*`,t*]
D/000740: 500320C8 1FA9704D 742A2D5E 2A8D5E2A [P.....pMt*-^*.^*]
D/000750: 60A9008D B32AA544 48201623 688D572A [`....*.DH..#h.W*]
D/000760: 4CD427A9 0520AA22 206427A0 00989140 [L.'....".d'....@]
D/000770: 60A907D0 02A90820 AA224CEA 22A90CD0 [`........"L."...]
D/000780: F6AD081D 8DBD35AD 091D8DBE 35A9098D [......5.....5...]
D/000790: 632A20C8 224CEA22 20A32220 8C26D0FB [c*.."L."..".&..]
D/0007A0: 4C7136A9 004CD523 A9018D63 2AAD6C2A [Lq6..L.#...c*.l*]
D/0007B0: D00AAD6D 2AD005A9 018D6C2A AD6C2A8D [...m*.....l*.l*.]
D/0007C0: BD35AD6D 2A8DBE35 20EA22A5 45D0034C [.5.m*..5..".E..L]
D/0007D0: C8268541 A5448540 20432720 4E27201A [.&.A.D.@.C'.N'..]
D/0007E0: 27AD632A 8DBB354C A826AD75 2AC9A0F0 ['.c*..5L.&.u*...]
D/0007F0: 25206427 B03A20FC 224CEA22 20AF27D0 [%.d'.:..."L."..'.]
D/000800: 05A9008D B32AA000 98914020 4E27A902 [.....*....@.N'..]
D/000810: 8DBB354C A8262092 27D00520 9A27F010 [..5L.&..'....'..]
D/000820: 20AF27F0 F620AA27 F0F120FC 224C1623 [..'....'...."L.#]
D/000830: 60A9092D 652AC909 F0034C00 20A90420 [`..-e*....L.....]
D/000840: D523AD73 2AAC722A 20E023AD 6D2AAC6C [.#.s*.r*..#.m*.l]
D/000850: 2A20E023 AD732AAC 722A4CFF 2320A822 [*..#.s*.r*L.#.."]
D/000860: A97F2DC2 35C904F0 034CD026 A90420D5 [..-.5....L.&....]
D/000870: 23207A24 AAAD652A 2901D006 8E722A8C [#.z$..e*)....r*.]
D/000880: 732A207A 24AE722A AC732A4C 7124205D [s*.z$.r*.s*Lq$.]]
D/000890: 23205128 6C722AAD B62AF020 A5D61003 [#.Q(lr*..*......]
D/0008A0: 4CCC26A9 0220D523 38A5AFE5 67A8A5B0 [L.&....#8...g...]
D/0008B0: E56820E0 23A568A4 674CFF23 A90120D5 [.h..#.h.gL.#....]
D/0008C0: 2338A54C E5CAA8A5 4DE5CB20 E023A5CB [#8.L....M....#..]
D/0008D0: A4CA4CFF 238DC235 4820A822 684CC427 [..L.#..5H.."hL.']
D/0008E0: 8CC1358C C3358DC2 35A9048D BB35A901 [..5..5..5....5..]
D/0008F0: 8DBC3520 A826ADC2 358DC335 4CA8268C [..5..&..5..5L.&.]
D/000900: C3358DC4 35A9024C 863620A8 264CEA22 [.5..5..L.6..&L."]
D/000910: 4CD02620 162320A8 22A9232D C235F0F0 [L.&..#.."#-.5..]
D/000920: 8DC235AD B62AF028 A90220B1 24207A24 [..5..*.(....$.z$]
D/000930: 186567AA 986568C5 74B07085 B0856A86 [.eg..eh.t.p...j.]
D/000940: AF8669A6 67A46820 71242051 286C601D [..i.g.h.q$.Q(l`.]
D/000950: A90120B1 24207A24 38A54CED 602AAAA5 [....$.z$8.L.`*..]
D/000960: 4DED612A 9045A8C4 4B9040F0 3E84CB86 [M.a*.E..K.@.>...]
D/000970: CA8EC335 8CC4354C 0A24AD0A 1D8DC335 [...5..5L.$.....5]
D/000980: AD0B1D8D C435A900 8DC235A9 028DC135 [.....5....5....5]
D/000990: A9038DBB 35A9028D BC3520A8 26AD612A [....5....5..&.a*]
D/0009A0: 8DC235A8 AD602A8D C1356020 EA224CCC [..5..`*..5`.."L.]
D/0009B0: 26CDC235 F01AAE5F 2A8E622A 4AF0034C [&..5..._*.b*J..L]
D/0009C0: 9E25A21D BD752A9D 932ACA10 F74C7A25 [.%...u*..*...Lz%]
D/0009D0: 60ADB62A F0038DB7 2A201324 20C81F20 [`..*....*..$....]
D/0009E0: 51286C58 1DA54A85 CCA54B85 CD6C561D [Q(lX..J...K..lV.]
D/0009F0: 20162420 C81F2051 286C561D 2065D685 [..$....Q(lV..e..]
D/000A00: 3385D84C D2D72065 0E853385 D84CD40F [3..L...e..3..L..]
D/000A10: 202625A9 058D522A 4C831F20 2625A901 [.&%...R*L...&%..]
D/000A20: 8D512A4C 831F2064 27900620 A3224C34 [.Q*L...d'...."L4]
D/000A30: 25204E27 AD652A29 06F013A2 03BD6E2A [%.N'.e*).....n*]
D/000A40: 9DBD35CA 10F7A90A 8DBB3520 A82660A9 [..5.......5..&`.]
D/000A50: 402D652A F005AD66 2AD005A9 FE8D662A [@-e*...f*.....f*]
D/000A60: AD0D1D8D BC35A90B 20AA224C 9723A906 [.....5...."L.#..]
D/000A70: 20AA22AD BF358D66 2A60A94C 20B225F0 [.."..5.f*`.L..%.]
D/000A80: 2EA9008D B62AA01E 209720A2 09BDB72A [.....*.........*]
D/000A90: 9D742ACA D0F7A9C0 8D512A4C D124A920 [.t*......Q*L.$..]
D/000AA0: 20B225F0 05A9014C D226A900 8DB72A4C [..%....L.&....*L]
D/000AB0: 841DCD00 E0F00E8D 80C0CD00 E0F0068D [................]
D/000AC0: 81C0CD00 E06020A3 22AD4F2A 8DB42AAD [.....`.."".O*..*.]
D/000AD0: 502A8DB5 2AAD752A 8DB32AD0 0E206427 [P*..*.u*..*...d']
D/000AE0: 900620A3 224CEB25 204E27AD 652A2904 [....."L.%.N'.e*).]
D/000AF0: F01BAD6E 2AD008AE 6F2AF011 CE6F2ACE [...n*...o*...o*.]
D/000B00: 6E2A208C 26F038C9 8DD0F7F0 E560205E [n*..&.8......`.^]
D/000B10: 26B066AD 5C2A8DC3 35A9048D BB35A901 [&.f.\*..5....5..]
```

```
D/000B20: 8DBC354C A826205E 26B04EA9 068D522A [..5L.&.^&.N...R*]
D/000B30: 208C26D0 0F20FC22 A903CD52 2AF0CEA9 [..&...."...R*...]
D/000B40: 054CD226 C9E09002 297F8D5C 2AAE5A2A [.L.&....)..\*.Z*]
D/000B50: F009CABD 00020980 9D00024C B31F48AD [..........L..H.]
D/000B60: B62AF00E A676E8F0 0DA633E0 DDF00768 [.*...v....3....h]
D/000B70: 1860A5D9 30F96838 6020FC22 205B274C [.`..0.h8`.".['L]
D/000B80: B31F209D 26204E27 A903D0A1 A9038DBB [....&.N'........]
D/000B90: 35A9018D BC3520A8 26ADC335 60ADB52A [5....5..&..5`..*]
D/000BA0: 8541ADB4 2A854060 20062B90 16ADC535 [.A..*.@`..+....5]
D/000BB0: C905F003 4C5E364C 9236EA20 693AA200 [....L^6L.6..i:..]
D/000BC0: 8EC33560 A90BD00A A90CD006 A90ED002 [..5`............]
D/000BD0: A90D8D5C 2A20E63F ADB62AF0 04A5D830 [...\*..?..*....0]
D/000BE0: 0EA20020 0227AE5C 2A200227 20C81F20 [.....'.\*..'....]
D/000BF0: 5128205E 26AE5C2A A903B003 6C5A1D6C [Q(.^&.\*....lZ.l]
D/000C00: 5E1DBD3F 2AAA8E63 2ABD7129 48098020 [^..?*..c*.q)H...]
D/000C10: C51FAE63 2AE86810 ED60AD66 2A8DBF35 [...c*.h..`.f*..5]
D/000C20: AD682A8D C035AD6A 2A8DC135 AD061D8D [.h*..5.j*..5....]
D/000C30: C335AD07 1D8DC435 A5408D4F 2AA5418D [.5.....5.@.O*.A.]
D/000C40: 502A60A0 1DB9752A 91408810 F860A01E [P*`...u*.@...`..]
D/000C50: B14099A9 35C8C026 D0F660A0 008C512A [.@..5..&..`...Q*]
D/000C60: 8C522A60 A9008545 2092274C 7327209A [.R*`...E..'Ls'..]
D/000C70: 27F01D20 AA27D00A A5408544 A5418545 ['.... '...@.D.A.E]
D/000C80: D0ECA01D B140D975 2AD0E388 10F61860 [.....@.u*......`]
D/000C90: 3860AD00 1DAE011D D00AA025 B140F009 [8`.........%.@..]
D/000CA0: AA88B140 86418540 8A60A000 B14060AD [...@.A.@.`...@`.]
D/000CB0: B32AF00E ADB42AC5 40D008AD B52AC541 [.*....*.@....*.A]
D/000CC0: F001CA60 4DC235F0 0A297FF0 0620EA22 [...`M.5..)......"]
D/000CD0: 4CD02660 38AD001D 8540AD01 1D8541AD [L.&`8....@....A.]
D/000CE0: 572A8D63 2AA00098 9140A01E 38A540E9 [W*.c*....@..8.@.]
D/000CF0: 2D914048 A541E900 C89140AA CA6848C8 [-.@H.A....@..hH.]
D/000D00: 91408AC8 9140AACA 6848C891 40C88A91 [.@...@..hH..@...]
D/000D10: 40CE632A F017AA68 38E926C8 9140488A [@.c*...h8.&..@H.]
D/000D20: E900C891 40854168 85404CE5 2748A900 [....@.Ah.@L.'H..]
D/000D30: C89140C8 9140ADB6 2AF00B68 85748570 [..@..@..*..h.t.p]
D/000D40: 68857385 6F606885 4D85CB68 854C85CA [h.s.o`h.M..h.L..]
D/000D50: 60A539CD 031DF012 8D562AA5 388D552A [`.9......V*.8.U*]
D/000D60: AD021D85 38AD031D 8539A537 CD051DF0 [....8....9.7....]
D/000D70: 128D542A A5368D53 2AAD041D 8536AD05 [..T*.6.S*....6..]
D/000D80: 1D853760 494E49D4 4C4F41C4 534156C5 [..7`INI.LOA.SAV.]
D/000D90: 5255CE43 484149CE 44454C45 54C54C4F [RU.CHAI.DELET.LO]
D/000DA0: 43CB554E 4C4F43CB 434C4F53 C5524541 [C.UNLOC.CLOS.REA]
D/000DB0: C4455845 C3575249 54C5504F 53495449 [.EXE.WRIT.POSITI]
D/000DC0: 4FCE4F50 45CE4150 50454EC4 52454E41 [O.OPE.APPEN.RENA]
D/000DD0: 4DC54341 54414C4F C74D4FCE 4E4F4D4F [M.CATALO.MO.NOMO]
D/000DE0: CE5052A3 494EA34D 41584649 4C45D346 [.PR.IN.MAXFILE.F]
D/000DF0: D0494ED4 42534156 C5424C4F 41C44252 [.IN.BSAV.BLOA.BR]
D/000E00: 55CE5645 524946D9 002170A0 70A170A0 [U.VERIF..!p.p.p.]
D/000E10: 70207020 70207020 70600022 06207422 [p.p.p.p.p`."..t"]
D/000E20: 06220423 78227030 70407040 80408008 [."#x"p0p@p@.@..]
D/000E30: 00080004 00407040 00217920 71207120 [.....@p@.!y.q.q.]
D/000E40: 70D6C4D3 CCD2C2C1 C3C9CF40 20100804 [p..........@....]
D/000E50: 0201C0A0 900000FE 00010002 00010007 [................]
D/000E60: 000100FF 7F0000FF 7F0000FF 7F0000FF [................]
D/000E70: FF0D078D 4C414E47 55414745 204E4F54 [....LANGUAGE.NOT]
D/000E80: 20415641 494C4142 4CC55241 4E474520 [.AVAILABL.RANGE.]
D/000E90: 4552524F D2575249 54452050 524F5445 [ERRO.WRITE.PROTE]
D/000EA0: 435445C4 454E4420 4F462044 4154C146 [CTE.END.OF.DAT.F]
D/000EB0: 494C4520 4E4F5420 464F554E C4564F4C [ILE.NOT.FOUN.VOL]
D/000EC0: 554D4520 4D49534D 415443C8 492F4F20 [UME.MISMATC.I/O.]
D/000ED0: 4552524F D2444953 4B204655 4CCC4649 [ERRO.DISK.FUL.FI]
D/000EE0: 4C45204C 4F434B45 C453594E 54415820 [LE.LOCKE.SYNTAX.]
D/000EF0: 4552524F D24E4F20 42554646 45525320 [ERRO.NO.BUFFERS.]
D/000F00: 41564149 4C41424C C546494C 45205459 [AVAILABL.FILE.TY]
```

```
D/000F10: 5045204D 49534D41 5443C850 524F4752 [PE.MISMATC.PROGR]
D/000F20: 414D2054 4F4F204C 415247C5 4E4F5420 [AM.TOO.LARG.NOT.]
D/000F30: 44495245 43542043 4F4D4D41 4EC48D00 [DIRECT.COMMAN...]
D/000F40: 03191924 333E4C5B 646D7884 98AABB00 [...$3>L[dmx.....]
D/000F50: 00000000 00000000 03000000 00000000 [................]
D/000F60: 00000000 00000000 00000000 01000000 [................]
D/000F70: 00000000 00C8C5CC CCCFA0A0 A0A0A0A0 [................]
D/000F80: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000F90: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000FA0: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0 [................]
D/000FB0: A0038400 00000000 C1D0D0CC C5D3CFC6 [................]
D/000FC0: D4E837BB 33BB3400 407E3321 2B052C57 [..7.3.4.@~3!+.,W]
D/000FD0: 2C6F2C2A 2D972DEE 2CF52C39 2C112D8D [,o,*-.-.,.,9,.-.]
D/000FE0: 2E172D7E 337E3389 2C952C86 2C922C7E [..-~3~3.,.,.,.,~]
D/000FF0: 337E33BD 2CC92CBA 2CC62C7E 33E000F0 [3~3.,.,.,.,~3...]
D/001000: 02A2028E 5F2ABA8E 9B33206A 2EADBB35 [...._*...3.j...5]
D/001010: C90DB00B 0AAABDCA 2A48BDC9 2A48604C [........*H..*H`L]
D/001020: 63332028 2B4C7F33 20DC2BA9 018DE335 [c3.(+L.3..+....5]
D/001030: AEBE35AD BD35D005 E000D001 E88DE835 [..5..5.........5]
D/001040: 8EE93520 C931905E 8E9C33AE 5F2ABD09 [..5..1.^..3._*..]
D/001050: 29AE9C33 4AB00DAD 512AC9C0 D0034C5F [)..3J...Q*....L_]
D/001060: 334C7333 A9009DE8 34A9019D E7348E9C [3Ls3....4....4..]
D/001070: 33204432 AE9C339D C7348DD2 358DD435 [3.D2..3..4..5..5]
D/001080: ADF1359D C6348DD1 358DD335 ADC2359D [..5..4..5..5..5.]
D/001090: C8342037 30200C2F 20D63720 3A2FAE9C [.4.70../..7.:/..]
D/0010A0: 33A9068D C535BDC6 348DD135 BDC7348D [3....5..4..5..4.]
D/0010B0: D235BDC8 348DC235 8DF635BD E7348DEE [.5..4..5..5..4..]
D/0010C0: 35BDE834 8DEF358E D935A9FF 8DE0358D [5..4..5..5....5.]
D/0010D0: E135ADE2 338DDA35 184C5E2F A900AA9D [.5..3..5.L^/....]
D/0010E0: D135E8E0 2DD0F8AD BF3549FF 8DF935AD [.5..-....5I...5.]
D/0010F0: C0358DF8 35ADC135 0A0A0A0A AA8EF735 [.5..5..5.......5]
D/001100: A9118DFA 3560201D 2F20342F 20C332A9 [....5`../.4/..2.]
D/001110: 022DD535 F02120F7 2FA90018 20113038 [.-.5.!../.....08]
D/001120: CED835D0 F7AED935 ADEE359D E734ADEF [..5....5..5..4..]
D/001130: 359DE834 2037304C 7F332028 2BADF635 [5..4.70L.3.(+..5]
D/001140: 302BADBD 358542AD BE358543 AE9C3320 [0+..5.B..5.C..3.]
D/001150: 1C322037 304C7F33 ADBC35C9 05B00B0A [.2.70L.3..5.....]
D/001160: AABDE62A 48BDE52A 48604C67 334C7B33 [...*H..*H`Lg3L{3]
D/001170: ADF63530 F8ADBC35 C905B0EE 0AAABDF2 [..50...5........]
D/001180: 2A48BDF1 2A486020 003320A8 2C8DC335 [*H..*H`..3..,..5]
D/001190: 4C7F3320 003320B5 3120A82C 4820A231 [L.3..3..1..,H..1]
D/0011A0: A0006891 424C962C 20B630B0 0BB14248 [..h.BL.,..0...BH]
D/0011B0: 205B3120 94316860 4C6F3320 0033ADC3 [.[1..1h`Lo3..3..]
D/0011C0: 3520DA2C 4C7F3320 003320A2 31A000B1 [5..,L.3..3..1...]
D/0011D0: 4220DA2C 20B5314C CA2C4820 B6306891 [B..,..1L.,H..0h.]
D/0011E0: 42A9400D D5358DD5 35205B31 4C9431A9 [B.@..5..5.[1L.1.]
D/0011F0: 808D9E33 D005A900 8D9E3320 282BAE9C [...3......3.(+..]
D/001200: 33BDC834 297F0D9E 339DC834 2037304C [3..4)...3..4.70L]
D/001210: 7F332000 334C7F33 20282B20 B630B0EF [.3..3L.3.(+..0..]
D/001220: EEE435D0 F6EEE535 4C1B2D20 282BAE9C [..5....5L.-.(+..]
D/001230: 33BDC834 10034C7B 33AE9C33 BDC6348D [3..4..L{3..3..4.]
D/001240: D1359DE6 34A9FF9D C634BCC7 348CD235 [.5..4....4..4..5]
D/001250: 20373018 205E2FB0 2A200C2F A00C8C9C [.70..^/.*../....]
D/001260: 33B14230 0BF00948 C8B142A8 6820892D [3.B0...H..B.h..-]
D/001270: AC9C33C8 C8D0E7AD D335ACD4 3520892D [..3......5..5..-]
D/001280: 38B0D120 FB2F4C7F 333820DD 32A900A2 [8..../L.38..2...]
D/001290: 059DF035 CA10FA60 20DC2BA9 FF8DF935 [...5...`..+....5]
D/0012A0: 20F72FA9 168D9D33 202F2E20 2F2EA20B [../....3./../...]
D/0012B0: BDAF3320 EDFDCA10 F78645AD F6378544 [..3.......E..7.D]
D/0012C0: 20422E20 2F2E202F 2E182011 30B05DA2 [.B../../....0.].]
D/0012D0: 008E9C33 BDC634F0 53304AA0 A0BDC834 [...3..4.S0J....4]
D/0012E0: 1002A0AA 9820EDFD BDC83429 7FA0070A [.........4)....]
D/0012F0: 0AB00388 D0FAB9A7 3320EDFD A9A020ED [........3.......]
```

```
D/001300: FDBDE734 8544BDE8 34854520 422EA9A0 [...4.D..4.E.B...]
D/001310: 20EDFDE8 E8E8A01D BDC63420 EDFDE888 [..........4.....]
D/001320: 10F6202F 2E203032 90A7B09E 4C7F33A9 [.../..02....L.3.]
D/001330: 8D20EDFD CE9D33D0 08200CFD A9158D9D [......3.........]
D/001340: 3360A002 A90048A5 44D9A433 9012F9A4 [3`....H.D..3....]
D/001350: 338544A5 45E90085 45686900 484C472E [3.D.E...Ehi.HLG.]
D/001360: 6809B020 EDFD8810 DB602008 2FA0008C [h........`../...]
D/001370: C535B142 99D135C8 C02DD0F6 18602008 [.5.B..5..-...`..]
D/001380: 2FA000B9 D1359142 C8C02DD0 F66020DC [/....5.B..-..`..]
D/001390: 2BA90420 5830ADF9 3549FF8D C133A911 [+...X0..5I...3..]
D/0013A0: 8DEB33A9 018DEC33 A238A900 9DBB33E8 [..3....3.8....3.]
D/0013B0: D0FAA20C E08CF014 A003B9A0 339DF333 [............3..3]
D/0013C0: E88810F6 E044D0EC A248D0E8 20FB2FA2 [.....D...H..../.]
D/0013D0: 008A9DBB 34E8D0FA 204530A9 11ACF033 [....4....E0....3]
D/0013E0: 88888DEC 378DBC34 8CBD34C8 8CED37A9 [....7..4..4...7.]
D/0013F0: 02205830 ACBD3488 3005D0EC 98F0E620 [..X0..4.0.......]
D/001400: C237204A 374C7F33 A200F006 A202D002 [.7.J7L.3........]
D/001410: A204BDC7 358542BD C8358543 602CD535 [....5.B..5.C`,.5]
D/001420: 70016020 E42FA902 205230A9 BF2DD535 [p.`../...R0..-.5]
D/001430: 8DD53560 ADD53530 0160204B 2FA90220 [..5`..50.`.K/...]
D/001440: 5230A97F 2DD5358D D53560AD C9358DF0 [R0..-.5..5`..5..]
D/001450: 37ADCA35 8DF137AE D335ACD4 35600820 [7..5..7..5..5`..]
D/001460: 342F204B 2F200C2F 28B009AE D135ACD2 [4/.K/../(....5..]
D/001470: 354CB52F A001B142 F008AAC8 B142A84C [5L./...B.....B.L]
D/001480: B52FADBB 35C904F0 02386020 4432A002 [./..5....8`.D2..]
D/001490: 91424888 ADF13591 4248203A 2F20D637 [.BH...5.BH.:/..7]
D/0014A0: A005ADDE 359142C8 ADDF3591 4268AA68 [....5.B...5.Bh.h]
D/0014B0: A8A902D0 02A9018E D3358CD4 35205230 [.........5..5.R0]
D/0014C0: A005B142 8DDC3518 6DDA358D DE35C8B1 [...B..5.m.5..5..]
D/0014D0: 428DDD35 6DDB358D DF351860 20E42FA9 [B..5m.5..5.`../.]
D/0014E0: 014C5230 ACCB35AD CC358CF0 378DF137 [.LR0..5..5..7..7]
D/0014F0: AED635AC D73560A9 01D002A9 02ACC32A [..5..5`........*]
D/001500: 8CF037AC C42A8CF1 37AEFA35 A0004C52 [..7..*..7..5..LR]
D/001510: 30082045 3028B008 ACBD33AE BC33D00A [0..E0(....3..3..]
D/001520: AEBC34D0 023860AC BD348E97 338C9833 [..4..8`..4..3..3]
D/001530: A9012052 30186020 4530AE97 33AC9833 [...R0.`.E0..3..3]
D/001540: A9024C52 30ADC52A 8DF037AD C62A8DF1 [..LR0..*..7..*..]
D/001550: 37608EEC 378CED37 8DF437C9 02D0060D [7`..7..7..7.....]
D/001560: D5358DD5 35ADF935 49FF8DEB 37ADF735 [.5..5..5I...7..5]
D/001570: 8DE937AD F8358DEA 37ADE235 8DF237AD [..7..5..7..5..7.]
D/001580: E3358DF3 37A9018D E837ACC1 2AADC22A [.5..7....7..*..*]
D/001590: 20B537AD F6378DBF 35A9FF8D EB37B001 [..7..7..5....7..]
D/0015A0: 60ADF537 A007C920 F008A004 C910F002 [`..7............]
D/0015B0: A008984C 8533ADE4 35CDE035 D008ADE5 [...L.3..5..5....]
D/0015C0: 35CDE135 F066201D 2FADE535 CDDD3590 [5..5.f../..5..5.]
D/0015D0: 1CD008AD E435CDDC 359012AD E535CDDF [.....5..5....5..]
D/0015E0: 359010D0 08ADE435 CDDE3590 06205E2F [5......5..5...^/]
D/0015F0: 90D76038 ADE435ED DC350A69 0CA8200C [..`8..5..5.i....]
D/001600: 2FB142D0 0FADBB35 C904F002 38602034 [/.B....5....8`.4]
D/001610: 314C2031 8DD635C8 B1428DD7 3520DC2F [1L.1..5..B..5../]
D/001620: ADE4358D E035ADE5 358DE135 20102FAC [..5..5..5..5../.]
D/001630: E6351860 8C9D3320 4432AC9D 33C89142 [.5.`..3.D2..3..B]
D/001640: 8DD73588 ADF13591 428DD635 20102F20 [..5...5.B..5../.]
D/001650: D637A9C0 0DD5358D D53560AE EA358EBD [.7....5..5`..5..]
D/001660: 35AEEB35 8EBE35AE EC35ACED 358EBF35 [5..5..5..5..5..5]
D/001670: 8CC035E8 D001C8CC E935D011 ECE835D0 [..5......5....5.]
D/001680: 0CA200A0 00EEEA35 D003EEEB 358EEC35 [.......5....5..5]
D/001690: 8CED3560 EEE635D0 08EEE435 D003EEE5 [..5`..5....5....]
D/0016A0: 3560ACC3 35AEC435 84428643 EEC335D0 [5`..5..5.B.C..5.]
D/0016B0: 03EEC435 60ACC135 D008AEC2 35F007CE [...5`..5....5...]
D/0016C0: C235CEC1 35604C7F 3320F72F ADC33585 [.5..5`L.3../..5.]
D/0016D0: 42ADC435 8543A901 8D9D33A9 008DD835 [B..5.C....3....5]
D/0016E0: 18EED835 201130B0 51A2008E 9C33BDC6 [...5..0.Q....3..]
```

```
D/0016F0: 34F01F30 22A000E8 E8E8B142 DDC634D0 [4..0".....B..4.]
D/001700: 0AC8C01E D0F3AE9C 33186020 303290DB [........3.`.02..]
D/001710: B0CFAC9D 33D0C1AC 9D33D0EF A000E8E8 [....3....3......]
D/001720: E8B1429D C634C8C0 1ED0F5AE 9C333860 [..B..4.......38`]
D/001730: 18AD9C33 6923AAE0 F560A900 AC9D33D0 [...3i#...`....3.]
D/001740: 974C7733 ADF135F0 21CEF035 301718A2 [.Lw3..5.!..50...]
D/001750: 043EF135 CAD0FA90 F0EEEE35 D003EEEF [.>.5.......5....]
D/001760: 35ADF035 60A9008D F135A900 8D9E3320 [5..5`....5....3.]
D/001770: F72F18AD EB336DEC 33F009CD EF339014 [./...3m.3....3..]
D/001780: A9FFD00A AD9E33D0 37A9018D 9E338DEC [......3.7....3..]
D/001790: 33186911 8DEB338D F135A80A 0AA8A204 [3.i...3..5......]
D/0017A0: 18B9F633 9DF135F0 0638A900 99F63388 [...3..5..8....3.]
D/0017B0: CAD0EE90 BD20FB2F ADF0338D F035D089 [......./..3..5..]
D/0017C0: 4C7733AD F135D001 604820F7 2FACF035 [Lw3..5..`H../..5]
D/0017D0: 681820DD 32A9008D F1354CFB 2FA2FC7E [h...2....5L./..~]
D/0017E0: F634E8D0 FAC8CCF0 33D0F20A 0AA8F00F [.4......3.......]
D/0017F0: A204BDF1 3519F633 99F63388 CAD0F360 [....5..3..3....`]
D/001800: ADBD358D E6358DEA 35ADBE35 8DE4358D [..5..5..5..5..5.]
D/001810: EB35A900 8DE535A0 10AAADE6 354AB003 [.5....5.....5J..]
D/001820: 8A900E18 ADE5356D E8358DE5 358A6DE9 [......5m.5..5.m.]
D/001830: 356A6EE5 356EE435 6EE63588 D0DB18AD [5jn.5n.5n.5.....]
D/001840: BF358DEC 356DE635 8DE635AD C0358DED [.5..5m.5..5..5..]
D/001850: 356DE435 8DE43590 03EEE535 600000A9 [5m.5..5....5`...]
D/001860: 01D022A9 02D01EA9 03D01AA9 04D016A9 [.."............]
D/001870: 05D012A9 06D00E4C ED3FEAA9 0AD006AD [.......L.?......]
D/001880: C5351890 0138088D C535A900 8548207E [.5...8...5...H.~]
D/001890: 2E28AE9B 339A6000 00000000 00000000 [.(..3.`.........]
D/0018A0: 0000FFFF 010A64D4 C9C1C2D3 D2C1C2A0 [......d.........]
D/0018B0: C5CDD5CC CFD6A0CB D3C9C404 110F0400 [................]
D/0018C0: 00FEFCFD FFFCFFFC FCFEFFFC FDFFFEFF [................]
D/0018D0: FDFEFFFD FDFDFFFE FDFFFCFF FFFDFDFC [................]
D/0018E0: FEFE7AFE FDFFFDFE FEFFFC11 01000023 [..z............#]
D/0018F0: 100001FF FCFCFDFC FEFCFCFD FDFEFCFC [................]
D/001900: FFFCFCFD FCFDFDFC FFFEFCFC FCFCFEFE [................]
D/001910: FCFDFCFC FCFEFCFD FCFCFEFC FCFCFEFC [................]
D/001920: FCFFFCFC FCFCFDFD FCFFFFFC FEFDFEFD [................]
D/001930: FEFCFFFC FEFFFDFC FFFCFCFC FDFEFDFE [................]
D/001940: FCFCFCFF FEFCFCFD FCFDFFFC FCFCFCFE [................]
D/001950: FDFCFDFC FCFCFDFC FDFFFCFE FDFCFCFD [................]
D/001960: FCFCFCFC FCFCFFFF FEFCFFFC FDFFFEFF [................]
D/001970: FEFEFCFC FDFCFCFC FEFEFEFD FEFDFCFE [................]
D/001980: FDFDFCFC FEFEFCFC FDFCFCFE FCFCFDFC [................]
D/001990: FEFEFCFD FCFCFCFC FCFDFDFC FEFDFCFC [................]
D/0019A0: FEFCFCFC FCFCFEFC FDFDFCFF FDFCFEFE [................]
D/0019B0: FCFDFDFE FFFDFEFF FEFFFC02 FCFDFCFC [................]
D/0019C0: FCFEFEFE FDFEFFFE FCFCFCFC FCFEFCFD [................]
D/0019D0: FCFCFFFD FCFEFCFC FDFEFEFF FCFEFFFD [................]
D/0019E0: FFFEFCFC FEFEFDFF FEFFFFFD FCFEFFFC [................]
D/0019F0: FEFDFEFC FEFCFFFE FEFEFCFD FDFEFFFC [................]
D/001A00: FCFCFCFC FCFEFCFD FDFCFFFD FCFEFEFC [................]
D/001A10: FEFDFCFF FFFCFCFE FEFDFEFD FEFEFCFC [................]
D/001A20: FEFDFDFC FEFDFCFC FEFCFCFE FCFCFDFC [................]
D/001A30: FDFDFCFF FDFCFEFD FCFDFFFC FEFCFCFC [................]
D/001A40: FCFCFCFD FCFCFEFC FDFCFCFF FCFCFEFD [................]
D/001A50: FCFDFFFC FFFDFEFF FDFFFEFC FDFFFFFF [................]
D/001A60: FCFCFEFC FCFFFFFC FCFDFCFE FCFCFCFC [................]
D/001A70: FCFEFCFC FDFDFCFC FEFCFDFC FCFEFEFC [................]
D/001A80: FCFCFCFC FFFCFCFC FCFDFDFC FFFCFCFE [................]
D/001A90: FFFCFDFF FCFFFDFE FFFDFDFD FFFEFDFF [................]
D/001AA0: FCFFFDFD FDFCFEFC FCFFFEFC FCFEFCFC [................]
D/001AB0: FEFCFDFD FCFFFDFC FEFCFCFF FFFCFCFD [................]
D/001AC0: FCFEFEFE FCFFFCFF FCFFFFFE FCFEFCFD [................]
D/001AD0: FFFFFDFD FEFCFCFC FCFCFEFD FDFFFEFC [................]
```

```
D/001AE0: FEFCFCFC FFFCFCFE FEFCFEFC FCFCFFFC [................]
D/001AF0: FCFCFEFC FCFFFDFE FCFCFDFC FCFFFDFC [................]
D/001B00: 01A527C9 09D018A5 2B4A4A4A 4A09C085 [..'.....+JJJJ...]
D/001B10: 3FA95C85 3E18ADFE 086DFF08 8DFE08AE [?.\.>....m......]
D/001B20: FF083015 BD4D0885 3DCEFF08 ADFE0885 [..0..M..=.......]
D/001B30: 27CEFE08 A62B6C3E 00EEFE08 EEFE0820 ['....+l>........]
D/001B40: 89FE2093 FE202FFB A62B6CFD 08000D0B [...../..+l.....]
D/001B50: 09070503 010E0C0A 08060402 0F002064 [..............d]
D/001B60: 27B008A9 00A88D5D 369140AD C5354CD2 ['......]6.@..5L.]
D/001B70: 26AD5D36 F008EEBD 35D003EE BE35A900 [&.]6....5....5..]
D/001B80: 8D5D364C B3368DBC 3520A826 20EA224C [.]6L.6..5..&.."L]
D/001B90: 7D22A013 B142D014 C8C017D0 F7A019B1 [}"...B.........]
D/001BA0: 4299A435 C8C01DD0 F64CBB26 A2FF8E5D [B..5.....L.&...]]
D/001BB0: 36D0F6AD BD358DE6 358DEA35 ADBE358D [6....5..5..5..5.]
D/001BC0: E7358DEB 358DE435 BA8E9B33 4C7F3300 [.5..5..5...3L.3.]
D/001BD0: 00000000 00000000 00000000 00000000 [................]
D/001BE0: 00000000 00000000 00000000 00000000 [................]
D/001BF0: 00000000 00000000 00000000 00003609 [..............6.]
D/001C00: 8EE9378E F737A901 8DF8378D EA37ADE0 [..7..7....7..7..]
D/001C10: 378DE137 A9028DEC 37A9048D ED37ACE7 [7..7....7....7..]
D/001C20: 37888CF1 37A9018D F4378A4A 4A4A4AAA [7...7....7.JJJJ.]
D/001C30: A9009DF8 049D7804 209337A2 FF9A8EEB [......x...7.....]
D/001C40: 374CC83F 2089FE4C 031BADE7 3738EDF1 [7L.?...L....78..]
D/001C50: 378DE137 ADE7378D F137CEF1 37A9028D [7..7..7..7..7...]
D/001C60: EC37A904 8DED37A9 028DF437 209337AD [.7....7....7..7.]
D/001C70: E7378DFE 36186909 8DF137A9 0A8DE137 [.7..6.i...7....7]
D/001C80: 38E9018D FF368DED 37209337 60000000 [8....6..7..7`...]
D/001C90: 000000AD E537ACE4 3720B537 ACED3788 [.....7..7..7..7.]
D/001CA0: 1007A00F EAEACEEC 378CED37 CEF137CE [........7..7..7.]
D/001CB0: E137D0DF 60087820 003DB003 28186028 [.7..`.x..=..(.`(]
D/001CC0: 3860ADBC 358DF137 A9008DF0 37ADF935 [8`..5..7....7..5]
D/001CD0: 49FF8DEB 3760A900 A89142C8 D0FB6000 [I...7`....B...`.]
D/001CE0: 1B000A1B E8370036 01600100 0000FB37 [.....7.6.`.....7]
D/001CF0: 00000001 00000060 01000000 01EFD800 [.......`........]
D/001D00: A200A002 88B13E4A 3E003C4A 3E003C99 [......>J>.<J>.<.]
D/001D10: 003BE8E0 5690EDA2 0098D0E8 A255BD00 [.;..V........U..]
D/001D20: 3C293F9D 003CCA10 F5603886 278E7806 [<)?..<...`8.'.x.]
D/001D30: BD8DC0BD 8EC0307C AD003C85 26A9FF9D [......0|..<.&...]
D/001D40: 8FC01D8C C04868EA A0044868 20B93888 [.....Hh...Hh..8.]
D/001D50: D0F8A9D5 20B838A9 AA20B838 A9AD20B8 [......8....8....]
D/001D60: 3898A056 D003B900 3C59FF3B AABD293A [8..V....<Y.;..):]
D/001D70: A6279D8D C0BD8CC0 88D0EBA5 26EA5900 [.'..........&.Y.]
D/001D80: 3BAABD29 3AAE7806 9D8DC0BD 8CC0B900 [;..):.x.........]
D/001D90: 3BC8D0EA AABD293A A62720BB 38A9DE20 [;.....):.'..8...]
D/001DA0: B838A9AA 20B838A9 EB20B838 A9FF20B8 [.8....8....8....]
D/001DB0: 38BD8EC0 BD8CC060 1848689D 8DC01D8C [8......`.Hh.....]
D/001DC0: C060A000 A256CA30 FBB9003B 5E003C2A [.`...V.0...;^.<*]
D/001DD0: 5E003C2A 913EC8C4 26D0EB60 A02088F0 [^.<*.>..&..`....]
D/001DE0: 61BD8CC0 10FB49D5 D0F4EABD 8CC010FB [a.....I.........]
D/001DF0: C9AAD0F2 A056BD8C C010FBC9 ADD0E7A9 [.....V..........]
D/001E00: 00888426 BC8CC010 FB59003A A4269900 [...&.....Y.:.&..]
D/001E10: 3CD0EE84 26BC8CC0 10FB5900 3AA42699 [<...&.....Y.:.&.]
D/001E20: 003BC8D0 EEBC8CC0 10FBD900 3AD013BD [.;..........:...]
D/001E30: 8CC010FB C9DED00A EABD8CC0 10FBC9AA [................]
D/001E40: F05C3860 A0FC8426 C8D004E6 26F0F3BD [.\8`...&....&...]
D/001E50: 8CC010FB C9D5D0F0 EABD8CC0 10FBC9AA [................]
D/001E60: D0F2A003 BD8CC010 FBC996D0 E7A90085 [................]
D/001E70: 27BD8CC0 10FB2A85 26BD8CC0 10FB2526 ['.....*.&.....%&]
D/001E80: 992C0045 278810E7 A8D0B7BD 8CC010FB [.,.E'...........]
D/001E90: C9DED0AE EABD8CC0 10FBC9AA D0A41860 [..............`]
D/001EA0: 862B852A CD7804F0 53A90085 26AD7804 [.+.*.x..S...&.x.]
D/001EB0: 852738E5 2AF033B0 0749FFEE 78049005 [.'8.*.3..I..x...]
D/001EC0: 69FECE78 04C52690 02A526C9 0CB001A8 [i..x..&...&.....]
```

```
D/001ED0: 3820EE39 B9113A20 003AA527 1820F139 [8..9..:..:.'...9]
D/001EE0: B91D3A20 003AE626 D0C32000 3A18AD78 [..:..:.&....:..x]
D/001EF0: 0429032A 052BAABD 80C0A62B 60000000 [.).*.+.....+`...]
D/001F00: A211CAD0 FDE646D0 02E64738 E901D0F0 [......F...G8....]
D/001F10: 60013028 24201E1D 1C1C1C1C 1C702C26 [`.0($........p,&]
D/001F20: 221F1E1D 1C1C1C1C 1C96979A 9B9D9E9F ["...............]
D/001F30: A6A7ABAC ADAEAFB2 B3B4B5B6 B7B9BABB [................]
D/001F40: BCBDBEBF CBCDCECF D3D6D7D9 DADBDCDD [................]
D/001F50: DEDFE5E6 E7E9EAEB ECEDEEEF F2F3F4F5 [................]
D/001F60: F6F7F9FA FBFCFDFE FFAE5F2A E01CF005 [..........._*....]
D/001F70: A2008E5D 3660A9FF 8DFB048D 0CC08D0E [...]6`..........]
D/001F80: C04C2FFB 00000000 00000000 00000000 [.L/.............]
D/001F90: 00000000 00000001 98990203 9C040506 [................]
D/001FA0: A0A1A2A3 A4A50708 A8A9AA09 0A0B0C0D [................]
D/001FB0: B0B10E0F 10111213 B8141516 1718191A [................]
D/001FC0: C0C1C2C3 C4C5C6C7 C8C9CA1B CC1C1D1E [................]
D/001FD0: D0D1D21F D4D52021 D8222324 25262728 [.......!."#$%&'(]
D/001FE0: E0E1E2E3 E4292A2B E82C2D2E 2F303132 [.....)*+.,-./012]
D/001FF0: F0F13334 35363738 F8393A3B 3C3D3E3F [..345678.9:;<=>?]
D/002000: 00000000 00000000 00000000 00000000 [................]
D/002010: 00000000 00000000 00000000 00000000 [................]
D/002020: 00000000 00000000 00000000 00000000 [................]
D/002030: 00000000 00000000 00000000 00000000 [................]
D/002040: 00000000 00000000 00000000 00000000 [................]
D/002050: 00000000 00000000 00000000 00000000 [................]
D/002060: 00000000 00000000 00000000 00000000 [................]
D/002070: 00000000 00000000 00000000 00000000 [................]
D/002080: 00000000 00000000 00000000 00000000 [................]
D/002090: 00000000 00000000 00000000 00000000 [................]
D/0020A0: 00000000 00000000 00000000 00000000 [................]
D/0020B0: 00000000 00000000 00000000 00000000 [................]
D/0020C0: 00000000 00000000 00000000 00000000 [................]
D/0020D0: 00000000 00000000 00000000 00000000 [................]
D/0020E0: 00000000 00000000 00000000 00000000 [................]
D/0020F0: 00000000 00000000 00000000 00000000 [................]
D/002100: 00000000 00000000 00000000 00000000 [................]
D/002110: 00000000 00000000 00000000 00000000 [................]
D/002120: 00000000 00000000 00000000 00000000 [................]
D/002130: 00000000 00000000 00000000 00000000 [................]
D/002140: 00000000 00000000 00000000 00000000 [................]
D/002150: 00000000 000038BD 8DC0BD8E C0305EA9 [......8......0^.]
D/002160: FF9D8FC0 DD8CC048 6820C33C 20C33C9D [.......Hh..<..<.]
D/002170: 8DC0DD8C C0EA88D0 F0A9D520 D53CA9AA [.............<..]
D/002180: 20D53CA9 9620D53C A54120C4 3CA54420 [..<....<.A..<.D.]
D/002190: C43CA53F 20C43CA5 41454445 3F484A05 [.<.?..<.AEDE?HJ.]
D/0021A0: 3E9D8DC0 BD8CC068 09AA20D4 3CA9DE20 [>......h....<...]
D/0021B0: D53CA9AA 20D53CA9 EB20D53C 18BD8EC0 [.<....<....<....]
D/0021C0: BD8CC060 484A053E 9D8DC0DD 8CC068EA [...`HJ.>......h.]
D/0021D0: EAEA09AA EAEA4868 9D8DC0DD 8CC06000 [......Hh......`.]
D/0021E0: 00000000 00000000 00000000 00000000 [................]
D/0021F0: 00000000 00000000 00000000 00000000 [................]
D/002200: 84488549 A0028CF8 06A0048C F804A001 [.H.I............]
D/002210: B148AAA0 0FD148F0 1B8A48B1 48AA6848 [.H....H...H.H.hH]
D/002220: 9148BD8E C0A008BD 8CC0DD8C C0D0F688 [.H.............]
D/002230: D0F868AA BD8EC0BD 8CC0A008 BD8CC048 [..h............H]
D/002240: 6848688E F805DD8C C0D00388 D0EE08BD [hHh.............]
D/002250: 89C0A006 B1489936 00C8C00A D0F6A003 [.....H.6........]
D/002260: B13C8547 A002B148 A010D148 F0069148 [.<.G...H...H...H]
D/002270: 28A00008 6A9005BD 8AC0B003 BD8BC066 [(...j..........f]
D/002280: 352808D0 0BA00720 003A88D0 FAAEF805 [5(.......:......]
D/002290: A004B148 205A3E28 D011A447 100DA012 [...H.Z>(...G....]
D/0022A0: 88D0FDE6 46D0F7E6 47D0F3A0 0CB148F0 [....F...G.....H.]
D/0022B0: 5AC904F0 586A08B0 03200038 A0308C78 [Z...Xj.....8.0.x]
```

```
D/0022C0: 05AEF805 20443990 24CE7805 10F3AD78 [.....D9.$.x....x]
D/0022D0: 0448A960 20953ECE F806F028 A9048DF8 [.H.`..>....(....]
D/0022E0: 04A90020 5A3E6820 5A3E4CBC 3DA42ECC [....Z>h.Z>L.=...]
D/0022F0: 7804F01C AD780448 9820953E 68CEF804 [x....x.H...>h...]
D/002300: D0E5F0CA 68A94028 4C483EF0 394CAF3E [....h.@(LH>.9L.>]
D/002310: A003B148 48A52FA0 0E914868 F008C52F [...HH./...Hh.../]
D/002320: F004A920 D0E1A005 B148A8B9 B83FC52D [.........H...?.-]
D/002330: D0972890 1C20DC38 08B08E28 A2008626 [..(....8...(...&]
D/002340: 20C238AE F8051824 38A00D91 48BD88C0 [..8....$8...H...]
D/002350: 60202A38 90F0A910 B0EE48A0 01B13C6A [`.*8......H...<j]
D/002360: 6890080A 206B3E4E 78046085 2A208E3E [h....k>Nx.`.*..>]
D/002370: B9780424 353003B9 F8048D78 04A52A24 [.x.$50.....x..*$]
D/002380: 35300599 F8041003 9978044C A0398A4A [50.......x.L.9.J]
D/002390: 4A4A4AA8 6048A002 B1486A66 35208E3E [JJJ.`H...Hjf5..>]
D/0023A0: 680A2435 300599F8 04100399 780460A0 [h.$50.......x.`.]
D/0023B0: 03B14885 41A9AA85 3EA056A9 00854499 [..H.A...>.V...D.]
D/0023C0: FF3B88D0 FA99003B 88D0FAA9 5020953E [.;.....;....P..>]
D/0023D0: A9288545 A544205A 3E200D3F A908B024 [.(.E.D.Z>..?...$]
D/0023E0: A9308D78 0538CE78 05F01920 4439B0F5 [.0.x.8.x....D9..]
D/0023F0: A52DD0F1 20DC38B0 ECE644A5 44C92390 [.-....8...D.D.#.]
D/002400: D3189005 A00D9148 38BD88C0 60A90085 [.......H8...`...]
D/002410: 3FA080D0 02A44520 563CB06B 202A38B0 [?.....E.V<.k.*8.]
D/002420: 66E63FA5 3FC91090 ECA00F84 3FA9308D [f.?.?.......?.0.]
D/002430: 780599A8 3F8810FA A4452087 3F20873F [x...?....E..?..?]
D/002440: 20873F48 68EA88D0 F1204439 B023A52D [..?Hh.....D9.#.-]
D/002450: F015A910 C545A545 E9018545 C905B011 [.....E.E...E....]
D/002460: 38602044 39B00520 DC38901C CE7805D0 [8`.D9....8...x..]
D/002470: F1204439 B00BA52D C90FD005 20DC3890 [..D9...-......8.]
D/002480: 8CCE7805 D0EB3860 A42DB9A8 3F30DDA9 [..x...8`.-..?0..]
D/002490: FF99A83F C63F10CA A544D00A A545C910 [...?.?...D...E..]
D/0024A0: 90E5C645 C6451860 00000000 00000000 [...E.E.`........]
D/0024B0: 00000000 00000000 000D0B09 07050301 [................]
D/0024C0: 0E0C0A08 0604020F 2093FEAD 81C0AD81 [................]
D/0024D0: C0A9008D 00E02076 3A4C4437 8D632A8D [.......v:LD7.c*.]
D/0024E0: 702A8D71 2A60205B 278CB72A 60207E2E [p*.q*`.['..*`.~.]
D/0024F0: AE9B339A 201623BA 8E9B33A9 094C8533 [..3...#...3..L.3]

File ............. "DOS33C.OBJ"
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 DOS33C.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  DOS33C
; ############################################################################################

                SBTL        "DOS3.3C w/ APPEND fix, U/L case"
                LST         ON,VSYM
                MSB         OFF
ORIGIN          EQU         $1B00
DIAGMODE        EQU         0
DOS33B          EQU         1
ULC             EQU         0                           ;1=ASM with lower case patch
                INCLUDE     RELOCTR,,2
                INCLUDE     DOSINIT,,2
                INCLUDE     DOSHOOK,,2
                INCLUDE     CMDSCAN,,2
                INCLUDE     XOPNCLS,,2
                INCLUDE     XLODSAV,,2
                INCLUDE     XMISCMD,,2
                INCLUDE     DOSGOER,,2
                INCLUDE     BLDFTAB,,2
                INCLUDE     CMDTBLS,,2
                INCLUDE     FDOSENT,,2
                INCLUDE     FOPCLRW,,2
                INCLUDE     FDELCAT,,2
                INCLUDE     FMTRWIO,,2
                INCLUDE     FLOCNXB,,2
                INCLUDE     FLOCSEC,,2
                INCLUDE     FVCBUFS,,2
                INCLUDE     BOOTLDR,,2
                INCLUDE     COREQUS,,1
                INCLUDE     PRENIBL,,1
                INCLUDE     WRITRTN,,1
                INCLUDE     POSTNRD,,1
                INCLUDE     RDADSEK,,1
                INCLUDE     MSWAITR,,1
                INCLUDE     WRITADR,,1
                INCLUDE     RWTSONE,,1
                INCLUDE     RWTSTWO,,1
                INCLUDE     FORMATR,,1
                INCLUDE     DOSPTCH,,1

; ############################################################################################
; #   END OF FILE:  DOS33C
; #   LINES      :  36
; #   CHARACTERS :  1440
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ############################################################################################
```

```
================================================================================
DOCUMENT DOSGOER.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  DOSGOER
; ###########################################################################################

               PAGE
;
; DOSGO - GOTO DOS
;
DOSGO          EQU          *
               JSR          DOSENT                    ; GO TO DOS
               BCC          DG3                       ; BR IF NOT ERROR
;
               LDA          CCBSTA                    ;GET RETURN CODE
               CMP          #5
               BEQ          YESEOF
               JMP          CLOSFILE                  ;NO. CLOSE & COMPLAIN
YESEOF         JMP          EOFFIX                    ;MABYE FIX IT UP?
               NOP
***************************
*
* DOS 3.3  (REV B) PATCH
*
***************************
               DO           DOS33B
DOSGO2A        JSR          MOVEOF                    ; MOVE END OF FILE PATCH
               ELSE
               NOP
DOSGO2A        NOP
               NOP
               FIN
               LDX          #0                        ; SET OTHER EIF
               STX          CCBDAT                    ; DONE
DG3            RTS
;
               PAGE
;
; ERROR ROUTINE
;
ESYNTX         LDA          #CREFLK+1
               BNE          ERROR
ENFA           LDA          #CREFLK+2
               BNE          ERROR
MFERR          LDA          #CREFLK+4
               BNE          ERROR
ETYP           EQU          *
ERNU1          LDA          #CREFLK+3
;
ERROR          EQU          *
               STA          SVA                       ; SAVE MSG NUMBER
               JSR          CLRSTS1                   ;PATCH TO CLR RSTATE TOO (WAS JSR
CLRSTS)
               LDA          ASIBSW                    ; GET AS/IN BASIC SWITCH
               BEQ          ERNAS                     ; BR IF NOT APPLESOFT
               LDA          ASONERR                   ;GET AS ERR FLAG
               BMI          ERRTN                     ; BRT IF ON ERR IS GO
ERNAS          EQU          *
               LDX          #0
```

```
                JSR         EMPR                    ; GO OUTPUT
                LDX         SVA                     ; GET SAVE MSG
                JSR         EMPR                    ; GO OUTPUT MSG
                JSR         PRCRIF                  ;OUTPUT A CARRAGE RETURN AFTER
MESSAGE
ERRTN           JSR         MVCSW                   ; GO MOVE CHAR I/ SW
                JSR         TSTRUN
                LDX         SVA
                LDA         #03
                BCS         ERRTN1                  ;DON'T GOTO BREAK HANDLER IF NOT
RUNNING
                JMP         (BREAK)
ERRTN1          JMP         (CONT)                  ;REENTER CONT IF NOT RUN
;
EMPR            EQU         *
                LDA         EMDTB,X                 ; GET ITS DISPL
                TAX                                 ; INTO X
EMPR1           EQU         *
                STX         TEMP1A                  ; SAVE DISPL
                LDA         EMSG,X                  ; GET MSG CHAR
                PHA                                 ; SAVE CHAR
                ORA         #$80                    ; SET MSB ON
                JSR         ORTN1                   ; OUTPUT CHAR
                LDX         TEMP1A                  ; GET INDEX
                INX                                 ; INCREMENT IT
                PLA                                 ; RE-LOAD CHAR
                BPL         EMPR1                   ; BR IF MORE CHARS
                RTS                                 ; DONE
                PAGE
;
; OPNSUP - OPEN SET UP
;
OPNSUP          EQU         *
                LDA         CV                      ; VOLUME
                STA         CCBVOL
                LDA         CD                      ; DRIVE
                STA         CCBDRV
                LDA         CS                      ; SLOT
                STA         CCBSLT
                LDA         FN1ADR                  ; FILENAME 1 ADR
                STA         CCBFN1
                LDA         FN1ADR+1
                STA         CCBFN1+1
                LDA         ZPGWRK
                STA         CFTABA
                LDA         ZPGWRK+1
                STA         CFTABA+1
                RTS
;
; MVFN1 - MOVE FILE NAME 1 TO FILE PTR
;
MVFN1           EQU         *
                LDY         #29
MVFN1A          LDA         FNAME1,Y
                STA         (ZPGWRK),Y
                DEY
                BPL         MVFN1A
                RTS
;
; MVBUFP - MOVE BUFFER PTRS TO CCB
;
MVBUFP          EQU         *
                LDY         #30
```

```
MVBP1           LDA         (ZPGWRK),Y
                STA         CCBFCB-30,Y
                INY
                CPY         #38
                BNE         MVBP1
                RTS
;
; CLRSTS - CLEAR STATES
;
CLRSTS          EQU         *
                LDY         #0
                STY         ISTATE
                STY         OSTATE
                RTS
                PAGE
;
; FILSRC - SEARCH FOR FILE NAME1
;
FILSRC          EQU         *
                LDA         #0                      ; CLEAR SV AVAIL
                STA         CNUM+1
;
                JSR         TSINIT                  ; GO INIT SEARCH
                JMP         FLS1A
FLS1            JSR         TSNXT                   ; LOOK AT NEXT
                BEQ         FLS4                    ; BR IF NO NEXT
;
FLS1A           JSR         TSTOPN                  ; GO TEST OPEN
                BNE         FLS2                    ; BR IF OPEN
;
                LDA         ZPGWRK                  ; SAVE AVAIL ENTRY ADR
                STA         CNUM
                LDA         ZPGWRK+1
                STA         CNUM+1
                BNE         FLS1                    ; GO LOOK SOME MORE
;
FLS2            LDY         #29                     ; FILE HAD 30 CHARS
FLS3            LDA         (ZPGWRK),Y              ; GET CHAR
                CMP         FNAME1,Y                TEST CHAR
                BNE         FLS1                    ; BR NOT
                DEY
                BPL         FLS3                    ; LOOK AT 30 CHARS
                CLC                                 ; FOUND
                RTS                                 ; DONE
;
FLS4            SEC                                 ; NOT FOUND
                RTS                                 ; DONE
                PAGE
;
; TSINIT - INITIALIZE FOR FTAB SEARCH
; TSNXT - GET NEXT FTAB ENTRY
;
TSINIT          EQU         *
                LDA         FTAB                    ; GET 1ST PTR ADR
                LDX         FTAB+1
                BNE         TSST
TSNXT           EQU         *
                LDY         #37                     ; GET LINK
                LDA         (ZPGWRK),Y
                BEQ         TSR                     ; BR IF NO LINK
;
                TAX
                DEY
```

```
                LDA           (ZPGWRK),Y
TSST            EQU           *
                STX           ZPGWRK+1
                STA           ZPGWRK
                TXA                                   ; SET NE CC
TSR             RTS                                   ; RTN
;
; TSTOPN - TST FOR OPEN FILE
;
TSTOPN          EQU           *
                LDY           #0                      ; GET 1ST CHAR OF FN
                LDA           (ZPGWRK),Y
                RTS
;
; TSTEXC - TEST CURRENT FILE FOR EXECUTE
;
TSTEXC          EQU           *
                LDA           ESTATE                  ; IF ESTATE = 0
                BEQ           TXC1                    ; THEN NO EXECUTE FILE
                LDA           EFTABA                  ; TEST CURRENT
                CMP           ZPGWRK
                BNE           TXC2                    ; IS NOT
                LDA           EFTABA+1
                CMP           ZPGWRK+1
                BEQ           TXC2                    ; IS
TXC1            DEX                                   ; IS NOT
TXC2            RTS                                   ; DONE
                PAGE
;
; TSTFUC - TEST FILE USE CODE FOR PGM
;
TSTFUC          EQU           *
                EOR           CCBFUC
                BEQ           TFUCR
                AND           #$7F
                BEQ           TFUCR
                JSR           ECLOSE                  ; GO CLOSE THE SOB
                JMP           ERNU1
TFUCR           RTS

; ###############################################################################################
; #   END OF FILE:  DOSGOER
; #   LINES      :  215
; #   CHARACTERS :  8197
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT DOSHOOK.pretty
================================================================================

; ###############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  DOSHOOK
; ###############################################################################

               PAGE
;
;CHRIN - CHAR RCVD VIA IN SWITCH
;
CHRIN          EQU          *
               JSR          SVREGS
               LDA          ISTATE                    ; IF NOT DISKIN
               BEQ          CHIN1                     ; THEN BRANCH, ELSE
               PHA                                    ;SAVE ISTATE
               LDA          SVA
               STA          ($28),Y                   ;REPLACE CURSOR
               PLA                                    ;GET ISTATE AGAIN
               BMI          CHIN0                     ;BRANCH IF NOT 'READ' FROM DISK
               JMP          ICFD                      ; AND GET CHAR FROM DISK
CHIN0          JSR          INITC
               LDY          $24                       ;GET CURSOR HORIZ
               LDA          #$60                      ;RESTORE A FLASHING CURSOR
               STA          ($28),Y                   ; TO PROMPT USER
CHIN1          EQU          *
               LDA          ESTATE
               BEQ          CHIN2
               JSR          NXTEXC                    ;RETURNS TO HERE ONLY WHEN 'EXEC' IS
EXHAUSTED
CHIN2          EQU          *
               LDA          #3                        ; SET OUT CHAR
               STA          OSTATE                    ; STATE TO INPUT ECHO
               JSR          LDREGS
               JSR          GETIN
               STA          SVA                       ;SAVE CHAR & INDEX
               STX          SVX
               JMP          ORTN
;
GETIN          JMP          (INSW)
;
;CHROUT - CHAR RCVD VIA OUTPUT SWITCH
;
CHROUT         EQU          *
               JSR          SVREGS                    ; SAVE REGS
;
               LDA          OSTATE                    ; GET OUT SPARE
               ASL          A
               TAX
               LDA          OUTSVT+1,X                ; GET ROUTINE ADR
               PHA
               LDA          OUTSVT,X
               PHA
               LDA          SVA
               RTS                                    ; GO TO ROUTINE
;
;SVREGS - SAVE REGS WHILE PROCESSING CHARS
;
SVREGS         EQU          *
               STA          SVA                       ; SAVE ACU
```

```
SVRGSA          EQU         *
                STX         SVX                     ; SAVE X
                STY         SVY                     ; SAVE Y
                TSX                                 ;SAVE STACK POINTER
                INX
                INX                                 ;ADJUST IT TO ORIGINAL
                STX         SVSTK
                LDX         #3                      ; SET FOR FOUR BYTE MOVE
SVRB            LDA         SVOUTS,X                ; MOVE SAVED OUT AND IN SW
                STA         OUTSW,X                 ; TO APPLE OUT/IN SW
                DEX
                BPL         SVRB
                RTS                                 ; DONE
                PAGE
;
;COS0 - 1ST CHAR OF PRINTED OUTPUT LINE
;CHECK FOR CNTL-D
;
COS0            EQU         *
                LDX         RSTATE                  ;FIRST CHECK FOR 'AFTER APSFT RELOC'
                BEQ         COS00                   ;BRANCH IF NOT
                JMP         COS7
COS00           LDX         ISTATE                  ; IS IN STATE NOT ZERO
                BEQ         COS01
                CMP         #'?'+$80                ; THEN IS THIS ?
                BEQ         COS6                    ; THEN PRINT ONLY IF MONITOR
                CMP         PROMPT
                BEQ         COS2A
COS01           EQU         *
                LDX         #2
                STX         OSTATE
                CMP         CCHAR                   ; IF NOT CNTL-D
                BNE         COS2                    ; THEN GO TO STATE 2
                DEX
                STX         OSTATE                  ; ELSE STATE = 1
                DEX
                STX         LBUFD                   ; AND LBUFD=0
;
;COS1 - ACCUMULATE CMD FROM PRINTED OUTPUT
;
COS1            EQU         *
                LDX         LBUFD                   ; GET LINE BUFF DISPL
COS1A           STA         LBUFF,X                 ; PUT CHAR IN BUFF
                INX                                 ; INCR PTR
                STX         LBUFD                   ; SAVE PTR
                CMP         #$8D                    ; WAS THIS A CR
                BNE         CMDRTN                  ; IF NOT THEN PR CHAR
;
                JMP         SCNCMD                  ; GO SCAN COMMAND
;
;COS2 - PRINTED OUTPUT, NOT FIRST CHAR
;
COS2            EQU         *
                CMP         #$8D                    ; IS IT A CR
                BNE         PRRTN                   ; BR IF NOT
COS2A           LDX         #0                      ; SET FOR POSSIBLE C-D NEXT
                STX         OSTATE                  ; NEXT STATE
                JMP         PRRTN                   ; GO PRINT CHAR
                PAGE
;
;COS3 - KEY IN ECHO PRINT
;
COS3            EQU         *
```

```
                LDX         #0
                STX         OSTATE                  ; RESET OUT STATE
                CMP         #$8D                    ; IS IT CR
                BEQ         COS3A                   ; IF CR THEN CMD CHECK
COS3B           LDA         ESTATE                  ; ELSE: IF NOT EXECUTE
                BEQ         PRRTN                   ; THEN PRINT CHAR
                BNE         DRTNI                   ; ELSE: PRINT IF MON INPUT
COS3A           PHA                                 ;SAVE CARRAGE RETURN
                SEC                                 ;ANTICIPATE EXEC FILE INPUT.
                LDA         ESTATE                  ;CHECK EXEC FLAG
                BNE         COS3C                   ;BR IF WAS INPUT FROM EXEC.
                JSR         TSTRUN                  ;GO TEST FOR RUN MODE.
COS3C           PLA
                BCC         COS3B                   ;IGNORE INPUT IF RUNNING.
                LDX         SVX                     ; GET LINE INDEX
                JMP         COS1A
;
;COS4 - DISK OUTPUT MODE
;
COS4            EQU         *
                CMP         #$8D                    ; IS IT CR
                BNE         COS4A                   ; BR IF NOT CR
                LDA         #5                      ; SET STATE FOR CNTL-D
                STA         OSTATE                  ; EXAMINE
COS4A           JSR         OCTD                    ; GO OUTPUT CJHAR TO DISK
                JMP         DRTNO                   ; GO TO DATA RETURN (OUT)
;
;COS5 - DISK OUTPUT MODE - 1ST CHAR OF A LINE
;
COS5            EQU         *
                CMP         CCHAR                   ; IS IT CNTL D
                BEQ         COS0                    ; BR IF CNTL- D
                CMP         #$8A                    ; LINE FEED?
                BEQ         COS4A
                LDX         #4
                STX         OSTATE                  ; SET NEW OUT STATE
                BNE         COS4                    ; BR IF NOT CNTL D
;
;COS6 - DISK INPUT ECHO
;
COS6            LDA         #0
                STA         OSTATE                  ; RESET OUT STATE = 0
                BEQ         DRTNI                   ; GO TO DATA IN RETURN
;
; COS7 - SPECIAL FOR RECOVER FROM AS ROM/RAM RELOC.
;
COS7            LDA         #0                      ;RESET RELOC STATE
                STA         RSTATE
                JSR         MVCSW                   ;FOR COMPATABILITY ON REENTRY
                JMP         ERUN1
                PAGE
;
;PRRTN - PRINT CHAR RETURN
;
;
; CMDRTN - PRINT CHAR IF MONITOR CMBS MODE
; DRTNO - PRINT CHAR IF MONITOR DATA OUT
; DRTNI - PRINT CHAR IF MONITOR DATA IN
;
CERTN           EQU         *
                LDA         LBUFF                   ; CHECK FOR PRINTED COMMAND
                CMP         CCHAR
                BEQ         CMDRTN                  ; IF PC THEN NO RESET X REG
```

```
                LDA         #$8D                    ; CARRAGE RETURN
                STA         LBUFF                   ; TO OUT BUFFER
                LDX         #0                      ; RESET TO SOL
                STX         SVX
CMDRTN          LDA         #MC
                BNE         MODECK
DRTNO           LDA         #MO
                BNE         MODECK
DRTNI           LDA         #MI
;
MODECK          EQU         *
                AND         MONMOD                  ; AND WITH MODE
                BEQ         ORTN                    ; BR IF NOT PRINT
PRRTN           JSR         LDREGS
                JSR         ORTN1
                STA         SVA                     ;SAVE REGISTERS
                STY         SVY
                STX         SVX
;
ORTN            EQU         *
                JSR         MVCSW                   ; GO MOVE CHAR I/O SWITCH
                LDX         SVSTK                   ;RESTORE ORIGINAL STACK POINTER
(YEEECH!)
                TXS
LDREGS          EQU         *
                LDA         SVA                     ; ACU
                LDY         SVY                     ; Y
                LDX         SVX                     ; X
                SEC                                 ;(FOR 'ESC' SCREEN FUNCTIONS)
                RTS                                 ; BY PASS PRINT
;
ORTN1           JMP         (OUTSW)
;
; PRCRIF - PRINT CR IF MON CMDS
;
PRCRIF          EQU         *
                LDA         #$8D                    ; ELSE PRINT CR
                JMP         ORTN1

; ###########################################################################################
; #   END OF FILE:  DOSHOOK
; #   LINES      :  215
; #   CHARACTERS :  9534
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT DOSINIT.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  DOSINIT
; ############################################################################################

HEREL           EQU         >*
REMDR           EQU         256-HEREL
                ORG         *+REMDR
;
;RELOCATION TABLES
;
START           EQU         *
SAT1            EQU         *
FTAB            DW          *-45                    ; START OF FTABS
CINA            DW          CHRIN                   ; CHAR IN ADR
COUTA           DW          CHROUT                  ; CHAR OUT ADR
FN1ADR          DW          FNAME1
FN2ADR          DW          FNAME2
SVBLA           DW          SVBL
ASTART          DW          BEGIN                   ; CHANGED TO START BY RELOCATE
CCBADR          DW          CCB
;
OUTSVT          EQU         *                       ; CHAR OUTPUT STATE VECTOR TABLE
                DW          COS0-1
                DW          COS1-1
                DW          COS2-1
                DW          COS3-1
                DW          COS4-1
                DW          COS5-1
                DW          COS6-1
;COMMAND EXECUTION TABLE
CMDETB          EQU         *
                DW          EINIT-1
                DW          ELOAD-1
                DW          ESAVE-1
                DW          ERUN-1
                DW          ECHAIN-1
                DW          EDEL-1
                DW          ELOCK-1
                DW          EUNLK-1
                DW          ECLOSE-1
                DW          EREAD-1
                DW          EEXEC-1
                DW          EWRITE-1
                DW          EPOS-1
                DW          EOPEN-1
                DW          EAPND-1
                DW          EREN-1
                DW          ECAT-1
                DW          EMON-1
                DW          ENOMON-1
                DW          EPR-1
                DW          EIN-1
                DW          EMAXF-1
                DW          EAS-1
                DW          EINT-1
                DW          EBSV-1
                DW          EBLD-1
```

```
                DW          EBRUN-1
                DW          EVAR-1
EAT1            EQU         *
                PAGE
;
;NON-RELOCATING ADRS
;
IBASVT          EQU         *
CHAIN           DW          IBCHN
RUN             DW          IBRUN
BREAK           DW          IBBRK
GO              DW          IBGO
CONT            DW          IBCONT                  ; BASIC CONT ENTRY POINT
ASEQ            DW          0
IBVT            DW          IBCHN
                DW          IBRUN
                DW          IBBRK
                DW          IBGO
                DW          IBCONT
IBVTL           EQU         *-IBVT
;
AS1VT           DW          ASRUN1
                DW          ASRUN1
                DW          ASBRK1
                DW          IBGO
                DW          ASCNTU1
                DW          ASRSEQ1
AS1VTL          EQU         *-AS1VT
;
AS2VT           DW          ASRUN2
                DW          ASRUN2
                DW          ASBRK2
                DW          DBINIT
                DW          ASCNTU2
                DW          ASRSEQ2
AS2VTL          EQU         *-AS2VT
                PAGE
;
;DOS BASIC INTERPRETER - INITIAL ENTRY
;
SC1             EQU         *
DBINIT          EQU         *
                LDA         IBSLOT                  ; GET BOOT SLOT
                LSR         A
                LSR         A
                LSR         A
                LSR         A
                STA         CS                      ; SET AS CUURENT SLOT
                LDA         IBDRVN                  ; GET BOOT DRIVE NUMBER
                STA         CD                      ; SET AS CURRENT DRIVE
                LDA         AITSTL                  ; GET APPLESOFT/IB TEST
                EOR         #ITSTV                  ; IF AS THEN
                BNE         IAS1                    ; GO TO AS INIT
;;ELSE INIT FOR IB
                STA         ASIBSW                  ; SET SW FOR IB
                LDX         #IBVTL                  ; GET IB VT LENGTH
IIB1            LDA         IBVT-1,X                ; MOVE IB ADDR
                STA         IBASVT-1,X
                DEX
                BNE         IIB1
                JMP         INITAA
;
IAS1            EQU         *
```

```
                LDA         #$40                    ; INDICATE ROM APPLESOFT
                STA         ASIBSW
                LDX         #AS1VTL
IAS1A           LDA         AS1VT-1,X               ; MOVE ROM AS ADRS
                STA         IBASVT-1,X
                DEX
                BNE         IAS1A
;
INITAA          EQU         *
                SEC                                 ; INDICATE INIT
                BCS         INITA
DBRST           LDA         ASIBSW                  ;GET APPLESOFT/INITGER BASIC FLAG.
                BNE         INITA1                  ;BRANCH IF NOT INTIGER BASIC
                LDA         #ITSTV                  ;GET INTIGER TEST VALUE AND GO SET
                BNE         INITA2                  ; ROM SWITCH TO 'IB'. (BRANCH
ALWAYS)
INITA1          ASL         A                       ;TEST FOR ROM APPLESOFT
                BPL         INITA3                  ;BRANCH IF RAM VERSION
                LDA         #ATSTV                  ;GET APPLESOFT TEST VALUE AND GO SET
INITA2          JSR         SWTST                   ;GO SELECT PROPER ROM BASIC.
INITA3          CLC                                 ; INDICATE RESET
;
INITA           EQU         *
                PHP                                 ; SAVE INIT/RESET
                JSR         MVCSW                   ; GO MOVE CHAR SWITCH
                LDA         #0                      ; CLR MONITOR MODES
                STA         MONMOD
;
                STA         OSTATE                  ; CLEAR OUTSTATE AND EXECUTE STATE
                PLP                                 ; GET INIT/RESET
                ROR         A                       ; SHIFT CARRY TO MSB
                STA         ISTATE                  ; SAVE INSTATE
                BMI         INITB                   ; BR IF INIT
                JMP         (CONT)                  ; GO TO CONTINUE ENTRY
INITB           JMP         (GO)                    ; GO TO GO ENTRY
                PAGE
INITC           EQU         *
                ASL         A                       ; OF ISTATE NOT ON
                BPL         INITD                   ; THEN NOT RAM AS
                STA         ASIBSW                  ; SET RAM AS
                LDX         #AS2VTL
IAS2A           LDA         AS2VT-1,X               ; MOVE RAM AS ADRS
                STA         IBASVT-1,X
                DEX
                BNE         IAS2A
                LDX         #29
IAS2B           LDA         FNAME2,X
                STA         FNAME1,X
                DEX
                BPL         IAS2B
;
INITD           EQU         *
                LDA         DFNFTS                  ; GO BUILD FILE TABS
                STA         CNFTBS                  ; AND SET MEM BOUNDS
                JSR         BLDFTB
                LDA         ESTATE                  ; GET EXEC STATE
                BEQ         INITZ                   ; BR IF NOT EXECUTE
                PHA                                 ; SVE CHAR
                JSR         MVEFTA                  ; GO MOVE EX FILE TAB ADR TO ZP
                PLA                                 ; GET SAVED CHAR
                LDY         #0
                STA         (ZPGWRK),Y              ;
INITZ           EQU         *
```

```
                JSR         CLRSTS                  ; SET IN AND OUT STATES TO ZERO
                LDA         CMDNO                   ; IF NOT BOOT (DUPLICATED FROM LINES
4540, 4550)
                BNE         INITF                   ; THEN DONE
                LDX         #IFBL
INITE           LDA         DBVECT,X                ; MOVE RESTART VECTORS
                STA         $3D0,X
                DEX
                BPL         INITE
                LDA         DBVECT+2                ;SET RESET VECTORS FOR NEW MONITOR.
                STA         ZRSET+1                 ;NOTE: THESE ARE NOT NORMALLY USED
AND
                EOR         #$A5                    ; ARE ONLY SET ONCE ON BOOT.
                STA         PWCNST                  ;POWER UP CONSTANT=COMPLIMENT OF HI
RESET VECTOR.
                LDA         DBVECT+1                ;SET LOW VECTOR ADDRESS.
                STA         ZRSET                   ;NOW APPLE RESET WILL KEEP DOS IN
I/O LOOP.
                LDA         #6                      ;INDICATE RUN
                BNE         INITF1                  ;LOAD AND RUN THE 'HELLO' PROGRAM
;
INITF           EQU         *
                LDA         SVCMD
                BEQ         INITG
INITF1          STA         CMDNO
                JMP         CMDGO1
INITG           EQU         *
                RTS
;
IFB             EQU         *
DBVECT          JMP         DBRST
                JMP         DBINIT
                JMP         USERENT                 ;USER EXTERNAL ENTRY TO FILE MANAGER
                JMP         DISKIO
CCBLDR          EQU         *
                LDA         CCBADR+1
                LDY         CCBADR
                RTS
IOBLDR          EQU         *
                LDA         AIOB+1
                LDY         AIOB
                RTS
                JMP         MVCSW
                NOP
                NOP
                JMP         MONBRK                  ;SET BREAK VECTOR FOR NEW MONITOR
(JMP IS FOR PROP RELOC)
                JMP         MONRST                  ;AFTER RELOC TO $3F2 THIS BECOMES:
DW DBRST,XOR(A5) ADR HI
                JMP         IORTS                   ;SET AS '&' FUNC TO KNOWN RTS
                JMP         MONRST                  ;GOTO MONITOR RESET (CONTROL Y
FUNCTION)
                JMP         MONRST                  ;GOTO MONITOR RESET
                DW          MONRST                  ;IRQ GOTO MONITOR RESET
IFBL            EQU         *-IFB-1

; ###########################################################################
; #   END OF FILE:  DOSINIT
; #   LINES      :  226
; #   CHARACTERS :  10022
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################
```

```
================================================================================
DOCUMENT DOSPTCH.pretty
================================================================================

; ###############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  DOSPTCH
; ###############################################################################

                SBTL        "DOS PATCHES"
*******************************
*
* DOS 3.2 PATCHES BY DICK HUSTON
*
*******************************
* AFTER THE FACT PATCHES
* CLRBYTE CALLED FROM DOS2 LABEL SOPTS +7 LINES
* CLRSTS1 CALLED FROM DOS3 LABEL ERROR +2 LINES
* ERROR9X CALLED FROM DOS5 LABEL ERROR9
*
*
                REP         40
*
* DOS 3.3 REVISION B PATCH
*
                REP         40
*****************************
SDP1            EQU         *                       ;START OF DOS PATCHES
RCPATCH         EQU         *
                JSR         SETVID
                LDA         $C081
                LDA         $C081
                LDA         #0
                STA         $E000
                DO          DOS33B
                JSR         OFF80
                JMP         RCBACK
                ELSE
                JMP         RCBACK
                DS          3,0
                FIN
*****************************
CLRBYTE         STA         TEMP1A
                STA         CB                      ;SET TYPE PARAM DEFAUTL=0
                STA         CB+1
                RTS
                SKP         4
CLRSTS1         JSR         CLRSTS
                STY         RSTATE                  ;PREVENTS FOREVER 'FILE NOT FOUND'
                RTS                                 ; IN APPLESOFT
*****************************
ERROR9X         JSR         RTNFCB
                LDX         ENTSTK                  ;GET STACK
                TXS                                 ;MESSY MESSY
                JSR         CLALL                   ;GO CLOSE EVERYBODY
                TSX
                STX         ENTSTK                  ;RESTORE SAVE STK
                LDA         #9
                JMP         ERRORA                  ;AND BACK
*****************************
EDP1            EQU         *-1                     ;END OF DOS PATCHES FOR RELOCTR
ENDOFDOS        EQU         *
```

```
                DO          ENDOFDOS-$4000
                FAIL        2,'DOS                    LENGTH NOT CORRECT'
                FIN

; ###########################################################################################
; #   END OF FILE:  DOSPTCH
; #   LINES      :  56
; #   CHARACTERS :  2060
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT EASM.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  EASM
; ##############################################################################

NEW
.CLOSE
DR1
ASMDOS33C,DOS33C.OBJ
ASMDOS33C,DOS33C.OBJ
ASMDOS33C,DOS33C.OBJ



; ##############################################################################
; #   END OF FILE:  EASM
; #   LINES      :  9
; #   CHARACTERS :  72
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT FDELCAT.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FDELCAT
; ##############################################################################

                PAGE
;
*   FLOCK - LOCK A FILE
;
FLOCK           LDA         #$80                    ; REMEMBER LOCK
                STA         TEMP3
                BNE         LCKGO
;
*   FUNLCK - UNLOCK A FILE
;
FUNLCK          LDA         #00                     ; REMEMBER UNLOCK
                STA         TEMP3
;
LCKGO           EQU         *
;
                JSR         DOPEN                   ; GO OPEN FILE
                LDX         TEMP1
                LDA         VDFILE+2,X              ; GET FILE USE CODE
                AND         #$7F                    ; TURN OFF LOCK
                ORA         TEMP3
                STA         VDFILE+2,X
                JSR         WRVDIR
GOGOOD          JMP         GOODIO
;
*   FPOSTN - POSITION A FILE
FPOSTN          JSR         LOCSEC                  ; GO POSITION
                JMP         GOODIO                  ; DONE
;
;
*   FVAR - VARIFY A FILE
;
FVAR            EQU         *
                JSR         DOPEN                   ; OPEN FILE
VAR1            JSR         LOCNXB                  ; READ A SECTOR
                BCS         GOGOOD                  ; BR IF EOF
                INC         DCBCRS                  ; INCREMENT SECTOR
                BNE         VAR1
                INC         DCBCRS+1
                JMP         VAR1                    ; READ THIS ONE
                PAGE
;
*   FDEL - DELETE A FILE
;
FDEL            EQU         *
                JSR         DOPEN                   ; GO OPEN FILE
;
FD2             LDX         TEMP1                   ; SAVED INDEX
                LDA         VDFILE+2,X              ; IS FILE LOCKED
                BPL         FD3                     ; BR NOT LOCKED
                JMP         ERRR10
;
FD3             EQU         *
                LDX         TEMP1                   ; GET SAVED INDEX
```

```
                LDA         VDFILE,X                ; GET DIR TRACK
                STA         DCBFDT                  ; SET AS 1ST FD TRACK
                STA         VDFILE+32,X             ; SAVE IN LC OF FN
                LDA         #$FF                    ; DELETED FILE MARKER
                STA         VDFILE,X                ; CLEAR ENTRY
                LDY         VDFILE+1,X              ; GET DIR SECTOR
                STY         DCBFDS                  ; SET AS 1ST FD SEC
                JSR         WRVDIR                  ; GO WRITE VOLUME DIR
                CLC
FD4             JSR         RDFDIR                  ; GET 1ST FILE DIR SECTOR
                BCS         FD7                     ; BR IF NO MORE
                JSR         MVFCBD                  ; MOVE DIR TO ZPG
                LDY         #FDENT                  ; POINT Y TO 1ST SEC ENT
FD5             STY         TEMP1                   ; SAVE Y
                LDA         (ZPGFCB),Y              ; GET REACK
                BMI         FD6                     ; BR IF NONE
                BEQ         FD6                     ; BR IF END OF FILE
                PHA                                 ; SAVE TRK
                INY
                LDA         (ZPGFCB),Y              ; GET SECTOR
                TAY                                 ; TO Y
                PLA                                 ; GET TRK
                JSR         FDSUB                   ; GO FREE SECTOR
FD6             LDY         TEMP1                   ; GET DIR INDEX
                INY                                 ; INCR TO NEXT ENTRY
                INY
                BNE         FD5                     ; BR NOT DONE THIS DIR
                LDA         DCBCDT                  ; GET THIS DIR TRK
                LDY         DCBCDS                  ; AND SECTOR
                JSR         FDSUB                   ; AND GO FREE IT
                SEC                                 ; GO
                BCS         FD4                     ; READ NEXT DIR
FD7             EQU         *
                JSR         WRVTOC
                JMP         GOODIO
;
FDSUB           EQU         *
                SEC                                 ; SET FOR RE USE OF SEC
                JSR         FRESEC                  ; GO FREE SECTOR
                LDA         #0                      ; CLEAR DCB BIT MAP
                LDX         #5                      ;CLEAR ALL OF TRK BITMAP SO
*     ;>16 SECTORS/TRK WILL WORK
FDS1            STA         DCBALS,X
                DEX
                BPL         FDS1
                RTS
                PAGE
;
*   RDIR - PRINT DIRECTORY
;
RDIR            EQU         *
                JSR         DCBSUP
                LDA         #$FF
                STA         DCBVOL
                JSR         RDVTOC
                LDA         #22                     ; SET 21 LINES
                STA         TEMP2
                JSR         PRCR                    ; GO PRINT
                JSR         PRCR                    ; PRINT ANOTHER CHAR
                LDX         #VML                    ; VOLUME MSG LENGTH
RD0             LDA         VOLMES,X                ; GET MSG CHAR
                JSR         PRINT                   ; PRINT IT
                DEX                                 ; DECREMENT COUNT
```

```
                BPL         RD0                     ; BR IF MORE
;
                STX         CNUM+1
                LDA         IBSMOD                  ; MOVE VOL NO FOR
                STA         CNUM                    ; CONVERSION
                JSR         PRNUM                   ; GO PRINT VOL NO
;
                JSR         PRCR                    ; PRINT CR
                JSR         PRCR                    ; AND AGAIN
;
                CLC                                 ; FIRST RECORD
;
RD1             JSR         RDVDIR                  ; GO READ REC
                BCS         RD5
                LDX         #0                      ; SET INDEX=0
RD2             STX         TEMP1                   ; SAVE INDEX
                LDA         VDFILE,X                ; GET TRACK
                BEQ         RD5                     ; BR IF END OF DIR
                BMI         RD4                     ; BR IF DELETED
;
                LDY         #$A0                    ; BLANK
                LDA         VDFILE+2,X              ; GET TYPE
                BPL         RD2A                    ; BR IF NOT LOCKED
                LDY         #'*'+$80                ; AST
RD2A            TYA                                 ; ACU = AST OR BLANK
                JSR         PRINT                   ; PRINT ACU
;
                LDA         VDFILE+2,X              ; GET TYPE
                AND         #$7F                    ; MASK OUT MISC
                LDY         #7                      ; SET INDEX = 7
                ASL         A                       ;GET RID OF HI BIT
RD2B            ASL         A                       ; SHIFT OUT MSB
                BCS         RD2C                    ; BR IF TYPE BIT OUT
                DEY                                 ; DEC INDEX
                BNE         RD2B                    ; BR IF NOT ACC BITS
RD2C            EQU         *
                LDA         FTTAB,Y                 ; GET TYPE CODE
                JSR         PRINT                   ; PRINT IT
                LDA         #$A0                    ; BLANK
                JSR         PRINT                   ; PRINT
;
                LDA         VDFILE+33,X             ; MOVE FILE LENGTH
                STA         CNUM                    ; TO CNUM
                LDA         VDFILE+34,X
                STA         CNUM+1
                JSR         PRNUM                   ; GO PRINT NUMBER
                LDA         #$A0                    ; BLANK
                JSR         PRINT                   ; PRINT
;
                INX
                INX
                INX
                LDY         #29
RD3             LDA         VDFILE,X                ; GET CHAR
                JSR         PRINT                   ; PRINT CHAR
                INX
                DEY
                BPL         RD3
RD3A            EQU         *
                JSR         PRCR                    ; GO PRINT CR
RD4             JSR         VDINC                   ; INCR INDEX
                BCC         RD2                     ; BR IF MORE IN DIR
                BCS         RD1                     ; GO READ NEXT DIR SECT
```

```
;
RD5             JMP         GOODIO                  ; DONE
;
PRCR            EQU         *
                LDA         #$8D                    ; CR
                JSR         PRINT                   ; PRINTED
                DEC         TEMP2                   ; DEC LINE COUNTER
                BNE         PRCR1                   ; BR IF NOT ZERO
                JSR         GETKEY                  ; WAIT FOR INPUT
                LDA         #21                     ; RESET LINE COUNTER
                STA         TEMP2
PRCR1           RTS                                 ; DONE
                PAGE
PRNUM           EQU         *
                LDY         #2                      ; 3 DIGITS
PRN1            LDA         #0                      ; INIT DIGIT TO ZERO
                PHA                                 ; SAVE IT
;
PRN2            LDA         CNUM                    ; GET NUMBER
                CMP         CVTAB,Y                 ; IF NUM < CVTAB ENTRY
                BCC         PRN3                    ; THEN DONE THIS DIGIT
;
                SBC         CVTAB,Y                 ; SUBTRACT TABLE ENTRY
                STA         CNUM                    ; FROM NUM
                LDA         CNUM+1
                SBC         #0
                STA         CNUM+1
                PLA                                 ; INCREMENT DIGIT
                ADC         #0
                PHA
                JMP         PRN2                    ; TRY AGAIN
;
PRN3            EQU         *
                PLA                                 ; GET DIGIT
                ORA         #$B0                    ; ADD ASCII 0
                JSR         PRINT                   ; PRINT IT
                DEY                                 ; DECREMENT DIGIT COUNT
                BPL         PRN1                    ; BR IF MORE DIGIT
;
                RTS                                 ; DONE
                PAGE
;
*   CLCFCB - GET FCB VIA INDEX AND MOVE IT
;
CLCFCB          EQU         *
;
                JSR         MVFCBP                  ; MOVE FCB PTR TO ZPG
                LDY         #0
                STY         CCBSTA
CF3             LDA         (ZPGFCB),Y              ; MOVE FCB TO
                STA         FCB,Y                   ; FCB WORK AREA
                INY
                CPY         #FCBLEN
                BNE         CF3
;
                CLC                                 ; DONE
                RTS
;
*   RTNFCB - MOVE FCB FROM WORK AREA TO FCB
;
RTNFCB          EQU         *
                JSR         MVFCBP                  ; MOVE FCB ADR TO ZPG
;
```

```
                LDY         #0
RF1             LDA         FCB,Y
                STA         (ZPGFCB),Y
                INY
                CPY         #FCBLEN
                BNE         RF1
                RTS

; ###############################################################################################
; #   END OF FILE:  FDELCAT
; #   LINES      :  249
; #   CHARACTERS :  11371
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT FDOSENT.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FDOSENT
; ###########################################################################################

                PAGE
;
; MISC BUT REQD CELLS
;
CFTABA          DDB         0                       ; CURRENT FILE TABLE POINTER
ISTATE          DFB         0                       ; INPUT STATE
OSTATE          DFB         0                       ; OUTPUT STATE
SVOUTS          DDB         0                       ; SAVED OUT SWITCH
SVINS           DDB         0                       ; SAVED IN SWITCH
CNFTBS          DFB         0                       ; CURRENT NO FILE TABLES
DFNFTB          DFB         3                       ; DEFAULT NO FILE TABLES
SVSTK           DFB         0                       ; SAVED STACK PTR
SVX             DFB         0                       ; DSAVED X REG
SVY             DFB         0                       ; SAVED Y REG
SVA             DFB         0                       ; SAVED ACU
LBUFD           DFB         0                       ; LINE BUFF DISPL
MONMOD          DFB         0                       ; MONITOR MODE BITS
MC              EQU         $40                     ; MONITOR CMDS
*MI EQU $20  ; MONITOR INPUT
*MO EQU $10  ; MONITOR OUTPUT
CMDNO           DFB         $00                     ; COMMAND NO IS ZERO FOR BOOT UP
SVBL            DFB         0,0
SVCMD           DFB         0
TEMP1A          DFB         0
TEMP2A          DFB         0
INOPTS          DFB         0                       ; INPUT OPTIONS
CUROPT          EQU         *                       ; CURRENT OPTIONS
CV              DW          0                       ; VOLUME
CD              DW          0                       ; DRIVE
CS              DW          0                       ; SLOT
CL              DW          1                       ; RECORD LENGTH
CR              DW          0                       ; RECORD NUMBER
CB              DW          0                       ; RECORD BYTE
CA              DW          0                       ; ADDRESS
IMBITS          DFB         0
                MSB         ON
FNAME1          ASC         "HELLO                      " ; FILENAME 1"
                MSB         OFF
FNAME2          DS          30,$A0                  ;FILENAME 2
DFNFTS          DFB         3                       ; DEFAULT FILE TABLES = 3
CCHAR           DFB         $84                     ; CONTROL CHAR
ESTATE          DFB         0                       ; EXECUTE STATE
EFTABA          DFB         0,0                     ; EXECUTE FILE TABLE POINTER
ASIBSW          DFB         0                       ; APPLESOFT, IB SWITCH
RSTATE          DFB         0                       ;FOR APPLESOFT RUN PROGRAM AFTER
RELOC
FASB            DFB         $C1,$D0,$D0,$CC         ;'APPLESOFT' WITH BIT 7 HIGH
                DFB         $C5,$D3,$CF,$C6
                DFB         $D4
FASBL           EQU         *-FASB
                PAGE
;
; DOS ADR TABLES (RELOCATED)
```

```
;
SAT2            EQU         *
AIOB            DW          IOB                     ; 5-ADR IOB
AVTOC           DW          VTOC                    ; 6-ADR VTOC
AVOLDR          DW          VOLDIR                  ; 7-ADR VOLDIR
AEND            DW          ENDOFDOS                ;END OF DOS
;
CMDVT           DW          GOODIO-1                ; 0-NULL
                DW          FOPEN-1                 ; 1-OPEN FILE
                DW          FCLOSE-1                ; 2-CLOSE FILE
                DW          FREAD-1                 ; 3-READ DATA
                DW          FWRITE-1                ; 4-WRITE DATA
                DW          FDEL-1                  ; 5-DELETE FILE
                DW          RDIR-1                  ; 6-READ DIRECTORY
                DW          FLOCK-1                 ; 7-LOCK A FILE
                DW          FUNLCK-1                ; 8-UNLOCK A FILE
                DW          FRNME-1                 ; 9-RENAME
                DW          FPOSTN-1                ; 10-POSITION A FILE
                DW          FFMT-1                  ; FORMAT
                DW          FVAR-1                  ; VARIFY
                DW          GOODIO-1                ; 11-SPARE
;
RVT             EQU         *
                DW          GOODIO-1
                DW          RNXBYT-1                ; 1-RD NEXT BYTE
                DW          RNXBLK-1                ; 1-RD NEXT BLOCK
                DW          RSPBYT-1                ; 2-RD SPECIFIC BYTE
                DW          RSPBLK-1                ; 3 - RD SPECIFIC BLOCK
                DW          GOODIO-1                ; 4 - SPARE
;
WVT             EQU         *
                DW          GOODIO-1
                DW          WNXBYT-1                ; 1-WR NEXT BYTE
                DW          WNXBLK-1                ; WR NEXT BLOCK
                DW          WSPBYT-1                ; 2-WR SPECIFIC BYTE
                DW          WSPBLK-1                ; 3-WR SPECIFIC BLOCK
                DW          GOODIO-1                ; 4 - SPARE
EAT2            EQU         *
                PAGE
;
; USERENT - DOS EXTERNAL ENTRY POINT (USER ENTRY)
; ENTRY PARM:
;  A= HIGH ADDRESS OF CCB
;  Y= LOW ADDRESS OF CCB
;  X= 0 IF CREATE DESIRED
;  X> 0 IF CREATE NOT DESIRED
; EXIT PARM:
;  CARRY CLEAR = OPERATION OK
;  CARRY SET = ERROR
;
SC2             EQU         *
USERENT         CPX         #0                      ;IF X=0 THEN FILE ENTRY CREATED IF
NOT
                BEQ         USRCR                   ; FOUND. NOTE: FILE NOT FOUND ERROR
STILL IS RETURNED
                LDX         #2                      ;INDICATE NO CREATE ALLOWED
USRCR           STX         CMDNO                   ;SET UP FOR CREATE CAPIBILITY
DOSENT          EQU         *
                TSX
                STX         ENTSTK
                JSR         CLCFCB                  ; GO CALCULATE FCB
                LDA         CCBREQ                  ; GET REQUEST
                CMP         #CRQMAX                 ; TTEST REQ RANGE
```

```
                BCS             ERR2                    ; BR OUT OF RANGE
                ASL             A                       ; REQ CODE *2
                TAX
                LDA             CMDVT+1,X               ; PUSH ADR ONTO STACK
                PHA
                LDA             CMDVT,X
                PHA
DENRTS          RTS
ERR2            JMP             ERROR2

; ###########################################################################################
; #   END OF FILE:  FDOSENT
; #   LINES      :  122
; #   CHARACTERS :  6315
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT FLOCNXB.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FLOCNXB
; ############################################################################################

               PAGE
;
;RDVTOC - READ VTOC
;WRVTOC - WRITE VTOC
;
RDVTOC         EQU          *
               LDA          #IBCRTS                  ; READ
               BNE          VTIO
WRVTOC         EQU          *
               LDA          #IBCWTS                  ; WRITE
;
VTIO           LDY          AVTOC                    ; MOVE BUFF ADR
               STY          IBBUFP
               LDY          AVTOC+1
               STY          IBBUFP+1
;
               LDX          DCBVTN                   ; GET TRACK
               LDY          #0
               JMP          DCBIO                    ; GO DO I/O
               PAGE
;
;RDVDIR - READ VOLUME DIRECTOR
;
RDVDIR         EQU          *
               PHP                                   ; SAVE STATUS
               JSR          MVVDBA
;
               PLP                                   ; GET STATUS
               BCS          RVDA                     ; BR IF R0 NEXT
;
RVDC           LDY          VDIRSC                   ; GET 1ST SECTOR
               LDX          VDIRTK                   ; GET FIRST TRK
               BNE          RVDGO                    ; GO READ
;
RVDA           EQU          *
               LDX          VDLTRK                   ; GET LINK TRACK
               BNE          RDVC                     ; BR IF A LINK
               SEC                                   ; SET END OF DIR
               RTS
;
RDVC           LDY          VDLSEC                   ; GET SECTOR
RVDGO          EQU          *
               STX          CVDTRK                   ; SET CUR TRACK
               STY          CVDSEC                   ; SET CUR SECTOR
               LDA          #IBCRTS                  ; GET CMD
               JSR          DCBIO                    ; GO DO I/O
               CLC
               RTS
               PAGE
;
;WRVDIR - WRITE VOLUME DIRECTORY SECTOR
;
WRVDIR         EQU          *
```

```
                JSR         MVVDBA
;
                LDX         CVDTRK                  ; CURRENT TRACK
                LDY         CVDSEC                  ; CURRENT SECTOR
                LDA         #IBCWTS                 ; WRITE COMMAND
                JMP         DCBIO                   ; GO DO I/O
;
;MVVDBA - MOVE VOL DIR BUF ADR TO IOB
;
MVVDBA          EQU         *
                LDA         AVOLDR                  ; MOVE ADR
                STA         IBBUFP
                LDA         AVOLDR+1
                STA         IBBUFP+1
                RTS
                PAGE
;
;DCBIO - DO I/O FOR A DCB
;
DCBIO           EQU         *
                STX         IBTRK                   ; TRACK
                STY         IBSECT                  ; SECTOR
DCBIO2          EQU         *
                STA         IBCMD                   ; COMMAND
                CMP         #IBCWTS
                BNE         DCBIO1
                ORA         DCBWRF
                STA         DCBWRF
DCBIO1          EQU         *
                LDA         DCBVOL                  ; VOL
                EOR         #$FF                    ; UNINVERT VOL BITS
                STA         IBVOL
                LDA         DCBSLT                  ; SLOT
                STA         IBSLOT
                LDA         DCBDRV                  ; DRIVE
                STA         IBDRVN
                LDA         DCBSDL                  ; LENGTH
                STA         IBDLEN
                LDA         DCBSDL+1
                STA         IBDLEN+1
                LDA         #1                      ; IOB TYPE
                STA         IBTYPE
;
                LDY         AIOB                    ; IOB ADR
                LDA         AIOB+1
                JSR         DISKIO                  ; GO DO I/O
;
                LDA         IBSMOD
                STA         CCBVOL
                LDA         #$FF                    ; RESET VOL
                STA         IBVOL
                BCS         BADIO                   ; BR IF BAD
                RTS                                 ; RTN IF GOOD
;
BADIO           LDA         IBSTAT                  ; GET STATUS
                LDY         #CREVMM
                CMP         #IBVMME                 ; WAS IT VOLUME MISMATCH
                BEQ         BD2                     ; BR IF YES
                LDY         #CREPRO
                CMP         #IBWPER
                BEQ         BD2
                LDY         #CREIOE
BD2             TYA
```

```
                JMP         ERRORB                  ; GO RTN
                PAGE
;
;LOCNXB - LOCATE NEXT BYTE
;
LOCNXB          EQU         *
                LDA         DCBCRS                  ; IS THE CURRENT RELATIVE SECTOR
                CMP         DCBCMS                  ; EQUAL TO THE CURRENT MEM SECTOR
                BNE         LNB1                    ; BR IF NOT EQ
                LDA         DCBCRS+1
                CMP         DCBCMS+1
                BEQ         LNB8                    ; BR IF REQD SECTOR IN MEM
;
LNB1            EQU         *                       ; NEED A DIFFERENT SECTOR IN MEM
                JSR         WRSECT                  ; GO WRITE SECTOR (IF REQD)
;
LNB2            LDA         DCBCRS+1                ; IS CURRENT REL SECTORY
                CMP         DCBDFS+1                ; IN CURRENT DIRECTORY (LOW LIMIT)
                BCC         LNB4                    ; BR IF IN A PREVIOUS DIR
                BNE         LNB3                    ; BR IF MAYBE IN THIS ONE
                LDA         DCBCRS                  ; TEST LOW BYTES
                CMP         DCBDFS
                BCC         LNB4                    ; BR IF IN PREVIOUS DIR
;
LNB3            LDA         DCBCRS+1                ; IS CURRENT REL SECTOR
                CMP         DCBDNF+1                ; IN CURRENT DIRECTOR (HI LIMIT)
                BCC         LNB6                    ; BR IF IN THIS ONE
                BNE         LNB4                    ; BR IF IN A NEXT DIR
                LDA         DCBCRS
                CMP         DCBDNF
                BCC         LNB6                    ; BR IF IN THIS ONE
;REQD SECTOR IN A NEXT DIRECTORY
LNB4            JSR         RDFDIR                  ; GO READ NEXT FILE DIR
                BCC         LNB2                    ; BR NXT AVAIL
                RTS                                 ; RETURN IF EOF DIR
;
;
LNB6            EQU         *                       ; CALCULATE DISPL INTO DIR
                SEC
                LDA         DCBCRS                  ; REQD REL SECTOR MINUS
                SBC         DCBDFS
                ASL         A                       ; TIMES 2
                ADC         #FDENT                  ; PLUS DISPL TO 1ST
                TAY
                JSR         MVFCBD                  ; MOVE DIR ADR TO ZPG
                LDA         (ZPGFCB),Y              ; GET TRACK
                BNE         LNB7                    ; BR IF NOT ZERO
                LDA         CCBREQ
                CMP         #CRQWR                  ; WRITE!
                BEQ         LNB7A
                SEC
                RTS
LNB7A           JSR         GNWSEC                  ; GO GET A NEW SECTOR
                JMP         LNBCON
LNB7            STA         DCBTRK                  ; SET TRK INTO DCB
                INY
                LDA         (ZPGFCB),Y              ; GET SECTOR
                STA         DCBSEC                  ; PUT INTO DCB
                JSR         RDSECT                  ; GO READ SECTOR
LNBCON          LDA         DCBCRS                  ; MOVE CUR REL SECTOR
                STA         DCBCMS
                LDA         DCBCRS+1                ; TO CUR MEM SECTOR
                STA         DCBCMS+1
```

```
;
LNB8            EQU         *
                JSR         MVFCBS                  ; MOVE SECTOR BUFF ADR TO ZP
                LDY         DCBCSB                  ; GET SECT BYTE
                CLC                                 ; CARRY CLEAR = ALL OK
                RTS                                 ; DONE
                PAGE
;
;
GNWSEC          EQU         *                       ; NEED NEW SECTOR
                STY         TEMP2                   ; SAVE DIR INDEX
                JSR         GETSEC                  ; GET A SECTOR
                LDY         TEMP2
                INY
                STA         (ZPGFCB),Y              ; SET NEW SECTOR
                STA         DCBSEC
                DEY
                LDA         DCBATK
                STA         (ZPGFCB),Y              ; SET NEW TRACK
                STA         DCBTRK
;
                JSR         MVFCBS
                JSR         CLRSEC                  ; GO CLEAR SECTOR
;
;
                LDA         #$C0                    ; INDICATE BOTH
                ORA         DCBWRF                  ; DIR AND SECTOR
                STA         DCBWRF                  ; MUST BE WRITTEN
                RTS                                 ; DONE
                PAGE
;
;INCRRB - INCREMENT RELATIVE RECORD BYTE
;
INCRRB          EQU         *
                LDX         DCBCRR                  ; MOVE BYTE JUST READ OR WRITTEN
                STX         CCBRRN
                LDX         DCBCRR+1
                STX         CCBRRN+1
                LDX         DCBCRB                  ; X=REL BYTE (LOW)
                LDY         DCBCRB+1                ; Y=REL BYTE HI
                STX         CCBBYT
                STY         CCBBYT+1
                INX                                 ; INC REL BYTE (LOW)
                BNE         INCR1                   ; BR IF NO CARRY
                INY                                 ; INC REL BYTE (HI)
;
INCR1           CPY         DCBRCL+1                ; REL BYTE=REC LENGTH
                BNE         INCR2                   ; BR IF NOT
                CPX         DCBRCL                  ; TEST LOW BYTES
                BNE         INCR2
                LDX         #0
                LDY         #0                      ; RESET REL BYTE TO ZERO
                INC         DCBCRR                  ; AND INCR
                BNE         INCR2                   ; RELATIVE RECORD
                INC         DCBCRR+1
;
INCR2           STX         DCBCRB                  ; SAVE NEW RELATIVE BYTE
                STY         DCBCRB+1
;
                RTS
                PAGE
;
;INCSCB - INCREMENT SECTOR BYTE
```

```
;
INCSCB          EQU          *
                INC          DCBCSB                    ; INC SECTOR BYTE
                BNE          INCS2                     ; BR IF NOT FULL
                INC          DCBCRS                    ; AND INCR
                BNE          INCS2                     ; RELATIVE SECTOR
                INC          DCBCRS+1
;
;
INCS2           EQU          *
                RTS                                    ; DONE
                PAGE
;
;MIBDA - MOVE AND INCREMENT CCBDAT
;
MIBDA           EQU          *
                LDY          CCBBBA                    ; Y=ADR LOW
                LDX          CCBBBA+1                  ; X=ADR HI
                STY          ZPGFCB                    ; PUT ADR INTO ZPG
                STX          ZPGFCB+1
;
                INC          CCBBBA                    ; INC ADR LOW
                BNE          MIB1                      ; BR IF NOT ZERO
                INC          CCBBBA+1                  ; INC ADR HI
MIB1            RTS                                    ; DONE
;
;DTBLN - DECREMENT BLOCK LENGTH AND TEST ZERO
;
DTBLN           EQU          *
                LDY          CCBBLN                    ; GET LEN LOW
                BNE          DTB1                      ; BR IF NOT ZERO
                LDX          CCBBLN+1                  ; GET LEN HI
                BEQ          DTB2                      ; BR IF LEN=0
                DEC          CCBBLN+1                  ; DEC LEN (HIGH)
DTB1            DEC          CCBBLN                    ; DEC LEN (LOW)
                RTS                                    ; DONE
;
DTB2            JMP          GOODIO                    ; FINISHED BLOCK

; ###############################################################################################
; #   END OF FILE:  FLOCNXB
; #   LINES      :  280
; #   CHARACTERS :  12148
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT FLOCSEC.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FLOCSEC
; ##############################################################################

               PAGE
;
;FNDFIL - FIND FILE NAME IN VOLUUME DIR
;
FNDFIL         EQU         *
               JSR         RDVTOC                  ; GO GET VTOC
               LDA         CCBFN1                  ; MOVE FN PTR
               STA         ZPGFCB                  ; TO ZERO PAGE
               LDA         CCBFN1+1
               STA         ZPGFCB+1
               LDA         #1
FF1            STA         TEMP2
               LDA         #0
               STA         DCBVDR
               CLC
FF2            EQU         *
               INC         DCBVDR
               JSR         RDVDIR                  ; GO GET VDIR SECTOR
               BCS         FF4A
               LDX         #0                      ; SET FOR 1ST FILE
;
FF3            STX         TEMP1                   ; SAVE INDEX
               LDA         VDFILE,X                ; GET FILE TRK
               BEQ         FF6                     ; BR IF LAST ENTRY
               BMI         FF7                     ; BR DELETED ENTRY
               LDY         #0                      ; X=X+3
               INX
               INX
FF4            INX
               LDA         (ZPGFCB),Y              ; GET FN CHAR
               CMP         VDFILE,X                ; COMPARE TO ENTRY CHAR
               BNE         FF5                     ; BR IF NOT SAME
               INY
               CPY         #30                     ; ALL 30 CHARS
               BNE         FF4                     ; BR IF NOT
               LDX         TEMP1                   ; GET INDEX
               CLC                                 ; FILE FOUND
               RTS                                 ; RETURN
;
FF5            EQU         *
               JSR         VDINC
               BCC         FF3
               BCS         FF2
;
FF6            LDY         TEMP2                   ; LOOKING FOR DELETED
               BNE         FF1                     ; BR IF NOT (DO)
;
FF7            LDY         TEMP2                   ; LOOKING FOR EMPTY
               BNE         FF5                     ; BR IF NOT
;
MVFN           EQU         *
               LDY         #0                      ; HAVE NEW ENTTRY
               INX
```

```
                INX
FF8             INX
                LDA         (ZPGFCB),Y              ; MOVE FILE NAME
                STA         VDFILE,X
                INY
                CPY         #30
                BNE         FF8
;
                LDX         TEMP1                   ; GET INDEX
                SEC                                 ; SET NOT OLD
                RTS                                 ; DONE
VDINC           EQU         *
                CLC
                LDA         TEMP1
                ADC         #35
                TAX
                CPX         #VDFLEN
                RTS
FF4A            EQU         *
                LDA         #0
                LDY         TEMP2
                BNE         FF1
                JMP         ERROR9
                PAGE
;
;GETSEC - GET A SECTOR
;
GETSEC          EQU         *
                LDA         DCBATK                  ; GET ALLOCATED TRK
                BEQ         GSS1                    ; BR IF NONE
;
GS0             EQU         *
                DEC         DCBALS                  ; DECREMENT SECTOR NO
                BMI         CS2                     ; BR IF NO SECTORS REM
;
                CLC
                LDX         #4                      ; 4 BYTE SHIFT
GS1             ROL         DCBABM-1,X              ; SHIFT BYTE LEFT
                DEX
                BNE         GS1
                BCC         GS0                     ; BR IF NO SECTOR
;
                INC         DCBNSA
                BNE         GS1A
                INC         DCBNSA+1
GS1A            EQU         *
                LDA         DCBALS                  ; GET ALLOCATED SECTOR
                RTS                                 ; RETURN
;
CS2             LDA         #0                      ; CLEAR ALLOCATED
                STA         DCBATK                  ; TRK
;
GSS1            LDA         #0                      ; SET SEARCH STATE=0
                STA         TEMP3
                JSR         RDVTOC                  ; GET VTOC
;
GS2             EQU         *
                CLC
                LDA         VALCA1                  ; GET LAST ALLOCATTED TRK
                ADC         VALCA2                  ; AD (+1) OR (-1)
                BEQ         GS3                     ; BR IF DECK TO ZERO
                CMP         VNOTRK
                BCC         GS5                     ; BR IF NOT AT OUTER LIMIT
```

```
                LDA         #$FF                    ; SET (-1)
                BNE         GS4
GS3             LDA         TEMP3                   ; GET SEARCH STATE
                BNE         ERR9                    ; BR IF NOT ZERO
                LDA         #1                      ; SET (+1)
                STA         TEMP3                   ; SET SEARCH STATE = 1
GS4             STA         VALCA2                  ; SET NEW (+1) OR -1)
                CLC
                ADC         #17                     ; ADD VTOC TRK NO
GS5             STA         VALCA1                  ; SET NEW LAST ALLOCATED
                STA         DCBATK                  ; PUT IN DCB
;
                TAY                                 ; ALLOCATED TRACK
                ASL         A                       ; TIME 4
                ASL         A
                TAY
                LDX         #4
                CLC
GS6             LDA         VSECAL+3,Y              ; MOVE BIT MAP BYTE
                STA         DCBABM-1,X
                BEQ         GS7                     ; BR IF NO BITS ON
                SEC                                 ; SET HAVE A SECTOR
                LDA         #0                      ; CLEAR VTOC BYTE
                STA         VSECAL+3,Y
GS7             DEY
                DEX
                BNE         GS6                     ; BR IF MORE TO MOVE
                BCC         GS2
                JSR         WRVTOC                  ; GO WRITE VTOC
                LDA         VNOSEC                  ; GET NO SECTORS
                STA         DCBALS                  ; SET IN DCB SECTOR BYTE
                BNE         GS0                     ; GO ALLOCATED SECTOR
ERR9            JMP         ERROR9
                PAGE
;
;FRETRK - FREE TRACK OF SECTORS
;
FRETRK          EQU         *
                LDA         DCBATK                  ; GET ALLOCATED TRACK
                BNE         FT1                     ; BR IF NONE
                RTS                                 ; DONE
FT1             PHA
                JSR         RDVTOC                  ; GET VTOC
                LDY         DCBALS                  ; GET SECTOS
                PLA                                 ; GET TRACK
                CLC                                 ; SET FREE
                JSR         FRESEC                  ; GO FREE
                LDA         #0                      ; CLEAR ALLOCATED TRK
                STA         DCBATK
                JMP         WRVTOC                  ; WRITE VTOC
;
;FRESEC - FREE A SECTOR
;A=TRK, Y=SECTOR, C=ON/OFF
;
FRESEC          EQU         *
FS1             LDX         #252                    ; 4 BYTE SHIFT
FS2             ROR         DCBABM-252,X            ; SHIFT IN CARRY
                INX                                 ; NEXT BYTE
                BNE         FS2                     ; BR IF NOT DONE
                INY                                 ; INC SECTOR NO
                CPY         VNOSEC                  ; NORMAL
                BNE         FS1                     ; BR IF NOT
;
```

```
                ASL         A                       ; TRACK*4
                ASL         A
                TAY
                BEQ         FS4
                LDX         #4
FS3             LDA         DCBABM-1,X              ; GET BIT MAP BYTE
                ORA         VSECAL+3,Y              ; OR WITH VTOC BM
                STA         VSECAL+3,Y
                DEY
                DEX
                BNE         FS3
FS4             RTS                                 ; DONE
                PAGE
;
;LOCSEC - LOCATE SECTOR FOR RECORD I/O
;
;RELSEC = (REL REC * RECLEN + RELBYTE)/256
;SECBYT = REMAINDER
;
LOCSEC          EQU         *
                LDA         CCBRRN                  ; RELATIVE RECORD NUMBER
                STA         DCBCSB                  ; TO CSB FOR MULT
                STA         DCBCRR                  ; AND CRR FOR SAVE
                LDA         CCBRRN+1
                STA         DCBCRS
                STA         DCBCRR+1
                LDA         #0
                STA         DCBCRS+1                ; HIGH CRS=0
                LDY         #16                     ; 16 BIT MULT
;
LS1             TAX                                 ; SAVE MS BYTE
                LDA         DCBCSB
                LSR         A                       ; IF NO CARRY THEN NO PART PROD
                BCS         LS1A
                TXA
                BCC         LS2
LS1A            CLC
                LDA         DCBCRS+1                ; FPORM PARTIAL PROD
                ADC         DCBRCL
                STA         DCBCRS+1
                TXA
                ADC         DCBRCL+1
;
LS2             ROR         A                       ; MULT BY 2
                ROR         DCBCRS+1
                ROR         DCBCRS
                ROR         DCBCSB
                DEY                                 ; DEC BIT COUNT
                BNE         LS1                     ; BR IF MORE BITS
;
                DO          DOS33B
                CLC                                 ; FOR FILE LENGTH > $7FFF BYTES
                FIN
                LDA         CCBBYT                  ; ADD REL BYTE RESULT
                STA         DCBCRB                  ; (SAVE REL BYTE)
                ADC         DCBCSB
                STA         DCBCSB
                LDA         CCBBYT+1
                STA         DCBCRB+1                ; (SAVE REL BYTE)
                ADC         DCBCRS
                STA         DCBCRS
                DO          DOS33B
                BCC         DONTINC
```

```
                INC             DCBCRS+1
DONTINC         RTS
                DS              2,$00
                ELSE
                LDA             #0
                ADC             DCBCRS+1
                STA             DCBCRS+1
                RTS
                FIN
                PAGE
ERROR1          LDA             #CREFUN
                BNE             ERRORA
ERROR2          LDA             #CRERR
                BNE             ERRORA
ERROR3          LDA             #CREMRE
                BNE             ERRORA
ERROR4          LDA             #CREPRO
                BNE             ERRORA
ERROR5          LDA             #CREEOF
                BNE             ERRORA
ERROR6          LDA             #CREFNF
                BNE             ERRORA
ERROR9          JMP             ERROR9X                 ;MUST CLOSE ALL FILES (WAS LDA
#CRENSA)
                NOP
ERRR10          LDA             #CREFLK
                BNE             ERRORA
GOODIO          LDA             CCBSTA
                CLC                                     ; CARRY=CLR
                BCC             RETURN                  ; GO RETURN
ERRORA          EQU             *
ERRORB          SEC                                     ; CARRY=SET
RETURN          EQU             *
                PHP
                STA             CCBSTA                  ; SET STA
                LDA             #0                      ;(FIX FOR APPLE SYS MONITOR $48 USED
BY RWTS)
                STA             $48                     ;(THIS ADDED 11/1/78)
                JSR             RTNFCB                  ; GO RTN FCB
                PLP                                     ; GET STATUS
                LDX             ENTSTK                  ; GET ENT STACK
                TXS                                     ; RESTORE STACK
                RTS                                     ; DONE
EC2             EQU             *

; ###############################################################################
; #   END OF FILE:  FLOCSEC
; #   LINES      :  284
; #   CHARACTERS :  12221
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT FMTRWIO.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FMTRWIO
; ##############################################################################

               PAGE
;
*   FFMT - EXECUTE FORMAT REQUEST
;
FFMT           EQU         *
               JSR         DCBSUP                  ; SET UP DCB
               LDA         #IBFMT
               JSR         DCBIO2
               LDA         DCBVOL                  ; SET VOL NO
               EOR         #$FF
               STA         VVOLNO
               LDA         #17
               STA         VALCA1                  ; ALOCATE BYTE 1
               LDA         #1
               STA         VALCA2                  ; ADD BYTE 2
;
               LDX         #VSECAL-VTOC
               LDA         #0
NT1            STA         VTOC,X                  ; CLEAR SECTOR AREA
               INX
               BNE         NT1
;
               LDX         #3*4                    ; START AT TRACK 3
NT2            CPX         #35*4                   ; END AT TRACK 35
               BEQ         NT4
               LDY         #3                      ; 4 BYTES OF INFO
NT3            LDA         ALC10S,Y                ; 10 SECTORS ALLOCATE
               STA         VSECAL,X
               INX
               DEY
               BPL         NT3
               CPX         #17*4                   ; AT TRACK 17
               BNE         NT2                     ; BR IF NOT
               LDX         #18*4                   ; SKIP TO 18
               BNE         NT2
;
NT4            JSR         WRVTOC                  ; WRITE NEW VTOC
;
               LDX         #0
               TXA
NT5            STA         VOLDIR,X                ; CLEAR VOLDIR
               INX
               BNE         NT5
;
               JSR         MVVDBA                  ; MOVE BUF PTRS
;
               LDA         #17                     ; TRACK 17
               LDY         VNOSEC
               DEY
               DEY
               STA         IBTRK                   ; INTO IOB
NT6            STA         VDLTRK                  ; INTO LINK
NT7            STY         VDLSEC
```

```
                INY
                STY         IBSECT
                LDA         #IBCWTS
                JSR         DCBIO2
                LDY         VDLSEC
                DEY                                 ; DECREMENT SECTOR
                BMI         NT8                     ; BR LAST WRITTEN
                BNE         NT7                     ; BR NOT LAST
                TYA                                 ; LAST, SET LINK TRK=0
                BEQ         NT6
;
NT8             EQU         *
                JSR         DLDSUP                  ; GO SET UP FOR DOSLDR
                JSR         WBOOT                   ; GO WRITE THE BOOT
                JMP         GOODIO                  ; DONE
                PAGE
;
*   MVFCBX - MOVE FCB ADRS TO ZPGFCB
;
MVFCBP          LDX         #0                      ; MOVE FCB ADR
                BEQ         MVF1
MVFCBD          LDX         #2                      ; MOVE FCB DIR BUFF
                BNE         MVF1
MVFCBS          LDX         #4                      ; MOVE FCB SECTOR BUFF
;
MVF1            EQU         *
                LDA         CFCBAD,X                ; DO THE MOVE
                STA         ZPGFCB
                LDA         CFCBAD+1,X
                STA         ZPGFCB+1
                RTS
;
                PAGE
;
*   WRSECT - WRITE CURRENT SECTOR IF REQD
;
WRSECT          EQU         *
                BIT         DCBWRF                  ; GET WRITE REQD FLAG
                BVS         WRSGO                   ; BR IF WRITE SECTOR REQD
                RTS                                 ; RTS
;
WRSGO           EQU         *
                JSR         MVSBA                   ; GO MOVE SECT BUFF ADR
;
                LDA         #IBCWTS                 ; GET COMMAND
                JSR         DCBIO                   ; GO FILL IN IOB AND DO IO
;
                LDA         #$BF                    ; SET WRITE SECTOR REQD BIT OFF
                AND         DCBWRF
                STA         DCBWRF
                RTS                                 ; DONE
                PAGE
;
*   WRFDIR - WRITE FILE DIRECTRY IF REQD
;
WRFDIR          EQU         *
                LDA         DCBWRF                  ; GET WRITE REQD FLAG
                BMI         WRFDGO                  ; BR IF WRITE DIR REQD
                RTS                                 ; DONE IF NOT
;
WRFDGO          EQU         *
                JSR         MVFDBA
;
```

```
                LDA         #IBCWTS                 ; GET WRITE CMD
                JSR         DCBIO                   ; GO FILL IN IOB AND DO I/O
;
                LDA         #$7F                    ; TURN WRITE DIR REQD BIT OFF
                AND         DCBWRF
                STA         DCBWRF
                RTS                                 ; DONE
;
*   MVFDBA - MOVE FILE DIRECTORY BUFF ASDR TO IOD
;
MVFDBA          EQU         *
                LDA         CFCBDR                  ; MOVE ADR
                STA         IBBUFP
                LDA         CFCBDR+1
                STA         IBBUFP+1
                LDX         DCBCDT                  ; GET TRACK
                LDY         DCBCDS                  ; GET SECTOR
                RTS
                PAGE
;
*   RDFDIR - READ FILE DIRECTORY
;
RDFDIR          EQU         *
                PHP                                 ; SAVE STATUS
                JSR         WRFDIR                  ; GO WRITE CURRENT DIR IF REQD
                JSR         MVFDBA                  ; GO MOVE DBUFF ADR TO IOB
                JSR         MVFCBD                  ; MOVE DBUFF ADR TO ZPG
                PLP                                 ; GET SAVED STATUS
                BCS         RFDNXT                  ; BR IF RD NEXT
;
                LDX         DCBFDT                  ; TRACK
                LDY         DCBFDS                  ; SECTOR
                JMP         RFDIO1                  ; GO READ
;
RFDNXT          EQU         *
                LDY         #FDLTRK                 ; GET LINK TRACK
                LDA         (ZPGFCB),Y
                BEQ         RFDNL                   ; NR NO LINK
                TAX                                 ; PUT TRACK INTO X
                INY
                LDA         (ZPGFCB),Y              ; SET LINK SECTOR
                TAY                                 ; PUT SECTOR INTO Y
                JMP         RFDIO1                  ; GO DO I/O
;
RFDNL           EQU         *
                LDA         CCBREQ                  ; THIS A WRITE
                CMP         #CRQWR
                BEQ         RFDNL1                  ; BR IF WRITE
                SEC                                 ; SET EOF
                RTS                                 ; RETURN
;
RFDNL1          EQU         *
                JSR         GETSEC                  ; GET A SECTOR
                LDY         #FDLSEC
                STA         (ZPGFCB),Y              ; PUT IN LINK
                PHA                                 ; SAVE SECTOR
                DEY
                LDA         DCBATK                  ; GET TRACK
                STA         (ZPGFCB),Y              ; PUT IN LINK
                PHA                                 ; SAVE TRACK
                JSR         WRFDGO                  ; GO WRITE OLD DIR DEC
;
                JSR         CLRSEC                  ; CLEAN OUT DIR
```

```
                LDY         #FDFRS                  ; SET NEW DIR SEC 1ST REL
                LDA         DCBDNF                  ; FILE SECTOR
                STA         (ZPGFCB),Y
                INY
                LDA         DCBDNF+1
                STA         (ZPGFCB),Y
;
                PLA                                 ; GET SAVED TRACK
                TAX                                 ; INTO X
                PLA                                 ; GET SAVED SECTOR
                TAY                                 ; INTO Y
                LDA         #IBCWTS                 ; SET WRITE CMD
                BNE         RFDIO2                  ; GO DO I/O
;
RFDIO1          LDA         #IBCRTS                 ; SET READ CMD
RFDIO2          STX         DCBCDT                  ; SET CURR TRACK
                STY         DCBCDS                  ; SET CURR SECTOR
                JSR         DCBIO                   ; GO I/O
;
RDFDC           LDY         #FDFRS                  ; GET POINTER TO FIRST REL SECTOR
                LDA         (ZPGFCB),Y              ; GET FRS
                STA         DCBDFS                  ; SET INTO DCB
                CLC
                ADC         DCBDMS                  ; ADD MAX SECTORS
                STA         DCBDNF                  ; PUT INTO DCB
;
                INY                                 ; DO SAME FOR HI BYTE
                LDA         (ZPGFCB),Y
                STA         DCBDFS+1
                ADC         DCBDMS+1
                STA         DCBDNF+1
;
                CLC
                RTS                                 ; DONE
                PAGE
;
;RDSECT - READ A SECTOR
;
RDSECT          EQU         *
                JSR         MVSBA                   ; GO MOVE SECTOR BUFFER ADR
;
                LDA         #IBCRTS
                JMP         DCBIO                   ; GO DO I/O
;
;MVSBA - MOVE SECTOR BUFFER ADR FOR I/O
;
MVSBA           EQU         *
                LDY         CFCBSB                  ; GET SECTOR BUFF ADR
                LDA         CFCBSB+1
MSB1            STY         IBBUFP                  ; SET IOB SECTOR
                STA         IBBUFP+1                ; BUFF PTR
                LDX         DCBTRK                  ; GET TRACK
                LDY         DCBSEC                  ; GET SECTOR
                RTS                                 ; RTN

; ##############################################################################
; #   END OF FILE:  FMTRWIO
; #   LINES      :  233
; #   CHARACTERS :  9921
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT FOPCLRW.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FOPCLRW
; ###########################################################################################

               PAGE
;
; FOPEN - OPEN A FILE
;
FOPEN          EQU          *
               JSR          DOPEN
               JMP          GOODIO
;
DOPEN          EQU          *
;
               JSR          DCBSUP
;
;
               LDA          #1
               STA          DCBSDL+1
               LDX          CCBRLN+1                  ; MOVE RECORD LENGTH
               LDA          CCBRLN
               BNE          F02
               CPX          #0
               BNE          F02
               INX                                    ; SET RL=256
F02            STA          DCBRCL
               STX          DCBRCL+1
;
               JSR          FNDFIL                    ; GO FIND FILE
               BCC          F03                       ; BR IF FOUND
;
*  CREATE FILE IF COMMAND IS SAVE, OPEN, OR BSAVE
;
               STX          TEMP1                     ; SAVE DIR. INDEX.
               LDX          CMDNO                     ;TEST COMMAND FOR CREATE FLAG.
               LDA          CMDSTB,X                  ; (BIT 0 MUST=1)
               LDX          TEMP1                     ;RESTORE DIR INDEX
               LSR          A                         ;SHIFT CREFLG BIT TO CARRY
               BCS          F02B                      ;BRANCH ON VALID INSTR.
               LDA          ISTATE                    ;FIND OUT IF TRYING TO LOAD
APPLESOFT
               CMP          #$C0
               BNE          F02A                      ;NO GO
               JMP          ERROR1
F02A           JMP          ERROR6                    ;PRINT "FILE NOT FOUND" MESSAGE.
F02B           LDA          #0
               STA          VDFILE+34,X
               LDA          #1
               STA          VDFILE+33,X
               STX          TEMP1                     ; SAVE VDIR INDEX
               JSR          GETSEC                    ; GO ALLOCATE SECTOR
               LDX          TEMP1
               STA          VDFILE+1,X                ; PUT SECTOR INTO VDIR
               STA          DCBFDS                    ; PUT SECTOR AS 1ST FILE DIR
               STA          DCBCDS                    ; PUT SECTOR AS CURRENT FILE DIR
;
               LDA          DCBATK                    ; GET ALLOCATED TRACK
```

```
                STA         VDFILE,X                ; PUT INTO VDIR
                STA         DCBFDT                  ; AND AS 1ST FILE DIR
                STA         DCBCDT                  ; AND AS CURRENT FILE DIR
;
                LDA         CCBFUC                  ; SET USE CODE
                STA         VDFILE+2,X              ; INTO DIRECTORY
;
                JSR         WRVDIR                  ; GO WRITE VOL DIRECTORY
;
                JSR         MVFCBD                  ; MOVE FILE DIR ADR TO ZP
                JSR         CLRSEC                  ; GO CLEAR IT
                JSR         WRFDGO                  ; GO WRITE FILE DIRECTORY
; DONE CREATION
                LDX         TEMP1                   ; RE-GET INDEX
                LDA         #CREFNF
                STA         CCBSTA
;
F03             EQU         *
                LDA         VDFILE,X                ; MOVE FILE DIR TRACK
                STA         DCBFDT
                LDA         VDFILE+1,X              ; MOVE FILE DIR SECTOR
                STA         DCBFDS
                LDA         VDFILE+2,X              ; 7OVE FILE USE CODE
                STA         CCBFUC
                STA         DCBFUC
                LDA         VDFILE+33,X
                STA         DCBNSA
                LDA         VDFILE+34,X
                STA         DCBNSA+1
                STX         DCBVDI                  ;SAVE DIRECTORY INDEX
;
                LDA         #255                    ; INDICATE NO SECTOR
                STA         DCBCMS                  ; IN MEMORY
                STA         DCBCMS+1
                LDA         VTDMS                   ; MOVE MAX FD SECTS
                STA         DCBDMS                  ; TO DCB
                CLC
                JMP         RDFDIR                  ; READ 1ST DIRECTORY RECORD
;
;
;
;
DCBSUP          EQU         *
                LDA         #0
                TAX
F01             STA         FCBDCB,X                ; CLEAR DCB
                INX
                CPX         #DCBLEN
                BNE         F01
;
                LDA         CCBVOL                  ; MOVE VOL
                EOR         #$FF                    ; INVERT VOL BITS
                STA         DCBVOL
                LDA         CCBDRV                  ; MOVE DRIVE
                STA         DCBDRV
                LDA         CCBSLT                  ; GET USER SPEC SLOT
                ASL         A                       ; SLOT*16
                ASL         A
                ASL         A
                ASL         A
                TAX
F01A            EQU         *
                STX         DCBSLT
```

```
                LDA         #17
                STA         DCBVTN
                RTS
                PAGE
;
; FCLOSE - CLOSE A FILE
;
FCLOSE          EQU         *
                JSR         WRSECT                  ; WRITE OPEN SECTOR
                JSR         WRFDIR                  ; GO WRITE FILE DIRECTORY
                JSR         FRETRK                  ; FREE UNUSED SECTORS
                LDA         #IBCWTS
                AND         DCBWRF
                BEQ         FC2
;
                JSR         RDVTOC                  ; READ VTOC
                LDA         #0
                CLC
FC1             EQU         *
                JSR         RDVDIR                  ; READ VDIR
                SEC
                DEC         DCBVDR
                BNE         FC1                     ; BR IF NOT
                LDX         DCBVDI                  ; GET FILES INDEX
                LDA         DCBNSA                  ; MOVE NO SECTORS ALLOCATED
                STA         VDFILE+33,X
                LDA         DCBNSA+1
                STA         VDFILE+34,X
                JSR         WRVDIR                  ; WRITE VOL DIR REC
;
;
FC2             EQU         *
                JMP         GOODIO                  ; DONE
                PAGE
;
*   FRNME - RENAME A FILE
;
FRNME           EQU         *
                JSR         DOPEN                   ; GO OPEN FILE
                LDA         DCBFUC                  ; GET USE CODE
                BMI         ER10                    ; BR IF LOCKED
                LDA         CCBFN2                  ; MOVE NEW FN
                STA         ZPGFCB                  ; PTR TO ZPG
                LDA         CCBFN2+1
                STA         ZPGFCB+1
                LDX         TEMP1                   ; GET VDIR INDEX
                JSR         MVFN                    ; GO MOVE FILE NAME
                JSR         WRVDIR                  ; GO WRITE VDIR
                JMP         GOODIO                  ; DONE RENAME
                PAGE
;
*   FREAD - READ A FILE
;
FREAD           EQU         *
;
                LDA         CCBRQM                  ; GET REQ MOD
                CMP         #CRMMAX                 ; TEST LIMIT
                BCS         ERR3A                   ; BR BAD
;
                ASL         A                       ; CODE*2
                TAX
                LDA         RVT+1,X                 ; GET READ ROUTINE
                PHA                                 ; VECTOR ADR
```

```
                LDA         RVT,X
                PHA                                 ; AND
                RTS                                 ; GO TO IT
;
ERR3A           JMP         ERROR3
ER10            JMP         ERRR10
;
*   FWRITE - WRITE A FILE
;
FWRITE          EQU         *
                LDA         DCBFUC                  ; IS FILE LOCKED
                BMI         ER10                    ; BR IF LOCKED
                LDA         CCBRQM                  ; GET REQ MOD
                CMP         #CRMMAX                 ; IN RANGE
                BCS         ERR3A                   ; BR IF NOT IN RANGE
;
                ASL         A
                TAX
                LDA         WVT+1,X                 ; GET ROUTINE ADR
                PHA
                LDA         WVT,X
                PHA
                RTS                                 ; AND GO TO IT
                PAGE
;
*   RSPBYT - READ A SPECIFIC BYTE
;
RSPBYT          EQU         *
                JSR         LOCSEC                  ; GO GET REQD REL SECTOR
;
*   RNXBYT - READ NEXT BYTE
;
RNXBYT          JSR         GETBYT                  ; GO GET BYTE
                STA         CCBDAT                  ; PUT IN CCB
                JMP         GOODIO                  ; DONE
;
*   RSPBLK - READ A SPECIFIC BLOCK
;
RSPBLK          JSR         LOCSEC                  ; GO LOCATE REL SECTOR
;
*   RNXBLK - READ NEXT BLOCK
;
RNXBLK          EQU         *
                JSR         DTBLN                   ; GO DECR LEN (NOT RTN IF=0)
                JSR         GETBYT                  ; GO GET BYTE
                PHA
                JSR         MIBDA                   ; GO MOVE BLOCK ADR AND INCR
                LDY         #0
                PLA
                STA         (ZPGFCB),Y              ; SET DATA BYTE
                JMP         RNXBLK                  ; GO FOR NEXT BYTE
;
*   GETBYT - GET A DATA BYTE
;
GETBYT          EQU         *
                JSR         LOCNXB                  ; LOCATE NEXT BYTE
                BCS         EOFIN                   ; BR IF EOF
                LDA         (ZPGFCB),Y              ; GET DAT BYTE
                PHA                                 ; SAVE IT
                JSR         INCRRB                  ; INCR REC BYTE
                JSR         INCSCB                  ; INCR SEC BYTE
                PLA                                 ; GET SAVED BYTE
                RTS                                 ; RETURN
```

```
;
EOFIN           JMP         ERROR5                  ; GO TO EOF RTN
                PAGE
;
*   WSPBYT - WRITE SPECIFIC BYTE
;
WSPBYT          EQU         *
                JSR         LOCSEC                  ; GO LOCATE SECTOR
;
*   WNXBYT - WRITE NEXT BYTE
;
WNXBYT          EQU         *
                LDA         CCBDAT                  ; GET THE BYTE
                JSR         PUTBYT                  ; GO WRITE BYTE
                JMP         GOODIO                  ; DONE
;
*   WSPBLK - WRITE A SPECIFIC BLOCK
;
WSPBLK          EQU         *
                JSR         LOCSEC                  ; GO LOCATE SECTOR
;
*   WNXBLK - WRITE NEXT BLOCK
;
WNXBLK          EQU         *
                JSR         MIBDA                   ; GO MOVE ADR TO ZPG AND DEC
                LDY         #0
                LDA         (ZPGFCB),Y              ; GET DATA BYTE
                JSR         PUTBYT                  ; GO PUT IT
                JSR         DTBLN                   ; GO DEC BLK LEN (NOT RTN IF = 0)
                JMP         WNXBLK
;
*   PUTBYT - PUT OUT ONE BYTE
;
PUTBYT          EQU         *
                PHA                                 ; SAVE DATA BYTE
                JSR         LOCNXB                  ; GO LOCATE NEXT BYTE
;
PB0             PLA                                 ; GET SAVED BYTE
                STA         (ZPGFCB),Y              ; PUT THE BYTE
                LDA         #$40                    ; SET WRITE SECTOR REQD
                ORA         DCBWRF
                STA         DCBWRF
;
                JSR         INCRRB                  ; INCR REL REC BYTE
                JMP         INCSCB                  ; INCR SECTOR BYTE

; ###############################################################################
; #   END OF FILE:  FOPCLRW
; #   LINES      :  286
; #   CHARACTERS :  11685
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT FORMATR.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FORMATR
; ###########################################################################################

                SBTL         '16-SECTOR FORMATTER'
****************************
*                          *
*  FORMAT DISK AND RETURN  *
*                          *
****************************
*
* EQUATES FOR FORMATTER
*
NSYNC           EQU          $45                       ;NUM GAP SELF-SYNC NIBLS
*
                REP          40
DSKFORM         EQU          *
                DO           DIAGMODE                  ;ELIMINATE FMTR FROM DIAG ASSEMBLY
                LST          OFF
                ELSE
                LDY          #3
                LDA          (IOBPL),Y                 VOLUME NUMBER IN IOB.
                STA          NVOL                      FOR FORMATTER.
                LDA          #$AA                      SET Z-PAG LOC TO $AA FOR
                STA          AA                        TIME DEPENDENT REFERENCES.
                LDY          #$56
                LDA          #0
                STA          TRK                       TRACK NUMBER, 0 TO 34
CLRNBUF2        STA          NBUF2-1,Y                 CLEAR NBUFS TO WRITE
                DEY          ZERO                      SECTORS.
                BNE          CLRNBUF2
CLRNBUF1        STA          NBUF1,Y
                DEY
                BNE          CLRNBUF1
                LDA          #$50
                JSR          SETTRK                    FAKE LIKE ON TRACK 80.
                LDA          #$28
                STA          NSYNC                     BEGIN WITH 40 SELF-SYNC NIBLS.
FORMTRK         LDA          TRK
                JSR          MYSEEK                    GOTO NEXT TRACK.
                JSR          WTRACK16                  WRITE AND VERIFY TRACK.
FORMERR1        LDA          #8                        'UNABLE TO FORMAT' ERR CODE.
                BCS          FORMERR                   CONTINUE IF NO ERROR.
                LDA          #$30                      UP TO 48 SECTOR RETRIES
                STA          RETRYCNT                  TO FIND SECTOR 0.
FINDS0          SEC          ANTICIPATE                'UNABLE TO FORMAT'
                DEC          RETRYCNT                  DONE 48 RETRIES?
                BEQ          FORMERR                   IF SO, 'UNABLE TO FORMAT' ERR.
                JSR          RDADR16                   ;READ ADR FIELD.
                BCS          FINDS0                    RETRY IF ERR.
                LDA          SECT                      CHECK SECTOR THAT WAS READ.
                BNE          FINDS0                    CONTINUE SEARCHING IF NOT SECT 0.
                JSR          READ16                    ;NOW READ DATA FIELD.
                BCS          FINDS0                    CONTINUE SEARCH IF ERR.
*       (NOW POSITIONED PROPERLY FOR NEXT TRACK)
                INC          TRK                       INCREMENT TRACK NUMBER.
                LDA          TRK
```

```
                CMP         #$23                    CONTINUE IF LESS THAN 35.
                BCC         FORMTRK
                CLC         CLEAR                   CARRY TO INDICATE 'NO ERR'
                BCC         FORMDONE                ELSE TURN OFF MOTOR AND RETURN.
FORMERR         LDY         #$0D
                STA         (IOBPL),Y               RETURN ERROR CODE.
                SEC         SET                     CARRY TO INDICATE ERR.
FORMDONE        LDA         MOTOROFF,X              TURN MOTOR OFF.
                RTS         AND                     RETURN.
                PAGE
*****************************
*                           *
*   WRITE TRACK SUBROUTINE  *
*                           *
*****************************
WTRACK16        LDA         #0
                STA         NSECT                   SECTOR NUMBER, 0 TO 15.
                LDY         #128                    ;128 NIBS PRIOR SECTOR 0
                BNE         WSECT0                  ; TO INSURE NO BLANK SPOT BETW 15 &
0
WSECT           LDY         NSYNC                   CURRENT NUM OF GAP SELF-SYNC NIBLS.
WSECT0          EQU         *
                JSR         WADR16                  WRITE GAP AND ADR FIELD.
                BCS         WEXIT2                  ERR IF WRITE PROTECTED.
                JSR         WRITE16                 ;WRITE SECTOR FROM NBUF1, NBUF2.
                BCS         WEXIT2                  ERR IF WRITE PROTECTED.
                INC         NSECT                   NEXT OF 16 SECTORS.
                LDA         NSECT
                CMP         #$10
                BCC         WSECT                   CONTINUE IF NOT DONE.
                PAGE
*****************************
*                           *
*     VERIFY ROUTINE        *
*                           *
*  VERIFIES THAT THE FIRST  *
*  SECTOR ENCOUNTERED IS    *
*  SECTOR 0, AND THAT ALL   *
*  16 SECTORS ARE READABLE  *
*  WITH MINIMAL RETRIES.    *
*  (2 REVOLUTIONS MAXIMUM)  *
*                           *
*  IF FIRST SECTOR IS NOT   *
*  SECTOR 0 THEN THE        *
*  CURRENT NUMBER OF SELF-  *
*  SYNC NIBLS IS DECR'D BY  *
*  1 (IF ALREADY LESS THAN  *
*  16) OR BY 2. THEN SECTOR *
*  15 IS LOCATED SO AS TO   *
*  POSITION THE NEW TRACK   *
*  REWRITE.                 *
*                           *
*  IF UNABLE TO READ ANY    *
*  SECTOR THEN THE ENTIRE   *
*  TRACK IS REWRITTEN.      *
*                           *
*  AFTER VERIFYING TRACK 0, *
*  THE NUMBER OF SELF-SYNC  *
*  NIBLS, NSYNC, IS DECR'D  *
*  BY 2 (IF STILL 16 OR     *
*  GREATER).                *
*                           *
*****************************
```

```
            PAGE
VTRACK      LDY     #$F
            STY     NSECT           SET 16 BYTES OF
            LDA     #$30            SECTOR FOUND TABLE
            STA     RETRYCNT        TO $30 (MARK THEM).
CLRFOUND    STA     FOUND,Y
            DEY
            BPL     CLRFOUND
            LDY     NSYNC           DELAY 50 USEC FOR EVERY
S0DELAY     JSR     WEXIT2          (12)  SELF-SYNC NIBL
            JSR     WEXIT2          (12)  EXPECTED TO INSURE
            JSR     WEXIT2          (12)  PROPER GAP PRIOR SECTOR 0.
            PHA     (3)
            PLA     (4)
            NOP     (2)
            DEY     (2)
            BNE     S0DELAY         (3)
            JSR     RDADR16         ;READ NEXT ADDRESS FIELD.
            BCS     S15LOC          ERR, LOCATE SECT 15 AND REWRITE TRK.
            LDA     SECT            WAS IT SECTOR 0?
            BEQ     VDATA           YES, NOW VERIFY DATA FIELD.
            LDA     #$10
            CMP     NSYNC           DECR NSYNC BY 1 IF LESS THAN
            LDA     NSYNC           16, BY 2 IF NOT LESS.
            SBC     #1
            STA     NSYNC
            CMP     #5              IF LESS THAN 5, UNRECOVERABLE
            BCS     S15LOC          ERR, ELSE REWRITE AFTER DATA FLD 15.
VERR        SEC     DRIVE           EXTREMELY FAST OR
            RTS     OTHER           SEVERE ERROR.
VSECT       JSR     RDADR16         ;READ AN ADDRESS FIELD.
            BCS     VERR1           RETRY IF ERR.
VDATA       JSR     READ16          ;READ DATA FIELD.
            BCC     SECTOK          (GOOD)
VERR1       DEC     RETRYCNT        NEXT OF 48 SECTOR TRIES.
            BNE     VSECT           (KEEP TRYING)
S15LOC      JSR     RDADR16         ;READ ADDRESS FIELD.
            BCS     NOTS15          ERR, TRY UP TO 128 TIMES.
            LDA     SECT            SECTOR THAT WAS READ.
            CMP     #$F             SECTOR 15?
            BNE     NOTS15          NO, CONTINUE SEARCHING.
            JSR     READ16          ;READ DATA FIELD.
            BCC     WTRACK16        WRITE TRACK FROM HERE IF NO ERR.
NOTS15      DEC     RETRYCNT        $FF TO $7F, 128 TRIES.
            BNE     S15LOC          TRY FOR SECT 15 AGAIN.
            SEC     SET             CARRY TO INDICATE VERIFY ERR.
WEXIT2      RTS     AND             RETURN TO FORMATTER.
SECTOK      LDY     SECT            THIS IS SECTOR READ.
            LDA     FOUND,Y         ALREADY FOUND?
            BMI     VERR1           YES, IGNORE IT.
            LDA     #$FF
            STA     FOUND,Y         INDICATE THIS SECT NOW FOUND.
            DEC     NSECT           FOUND 16 SECTORS?
            BPL     VSECT           NO, LOOK FOR NEXT.
            LDA     TRK
            BNE     WEXIT1          IF TRACK 0 AND NSYNC > 16
            LDA     NSYNC           (NUM GAP SYNC NIBLS)
            CMP     #$10            THEN SUBTRACT 2 FROM NSYNC
            BCC     WEXIT2          TO AVOID RETRIES ON LATER TRKS.
            DEC     NSYNC
            DEC     NSYNC
WEXIT1      CLC     INDICATE        NO ERROR.
            RTS     RETURN.
```

```
*****************************
AEC2            EQU         *                       ;TELL RELOCTR WHERE RWTS ENDS
*****************************
FOUND           DFB         0,0,0,0                 'SECTOR FOUND' TABLE.
                DFB         0,0,0,0
                DFB         0,0,0,0
                DFB         0,0,0,0
                FIN
                LST         ON
                REP         40
* THIS TABLE IS USED TO TRANSLATE
*  LOGICAL (REQUESTED) SECTOR NUMBER
*  TO PHYSICAL SECTOR NUMBER. THE
*  DISKETTE IS FORMATTED WITH ALL
*  SECTORS IN MONOTONICALLY INCREASING
*  ORDER. THE TRANSLATION WILL ALLOW
*  TIME BETWEEN SECTORS FOR READS.
*
                REP         40
*
* NOTE: THE CURRENT IMPLEMENTATION OF DOS
*  USUALLY ACCESSES SECTORS IN DECREASING
*  ORDER ON A TRACK. THUS WE WILL
*  TRANSLATE IN REVERSE ORDER...
*
* THE INTERLEAVE IS THEN 9:1
*
* NOTE: WE MAP LOGICAL SECTOR 0
*  INTO PHYSICAL SECTOR 0 SO THAT
*  WRITING OF BOOT DURING 'INIT'
*  IS CORRECT FOR SECTOR ZERO.
*
INTRLEAV        EQU         *
                DFB         $00,$0D,$0B,$09
                DFB         $07,$05,$03,$01
                DFB         $0E,$0C,$0A,$08
                DFB         $06,$04,$02,$0F

; ##############################################################################################
; #   END OF FILE:  FORMATR
; #   LINES      :  215
; #   CHARACTERS :  10690
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################################
```

```
================================================================================
DOCUMENT FVCBUFS.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  FVCBUFS
; ##############################################################################

               PAGE
;MISC DOS WORK CELLS
;
CVDTRK         DFB          0                        ; CUR VOL DIR TRK
CVDSEC         DFB          0                        ; CUR VOL DIR SECTOR
CURCCB         DFB          0,0                      ; CURRENT CCB ADR
ENTSTK         DFB          0                        ; ENTRY STACK POINTER
TEMP1          DFB          0                        ; TEMP BYTE1
TEMP2          DFB          0                        ; TEMP BYTE 2
TEMP3          DFB          0                        ; TEMP BYTE 3
ENTSLT         DFB          0                        ; BOOT SLOT SAVED
ALC10S         DFB          0,0,$FF,$FF              ; ALLOCATATION TRACK BIT MAP
CVTAB          DFB          1,10,100                 ; CONVERSION TABLE
               MSB          ON
FTTAB          ASC          "TIAB" ; FILE TYPE CONVERSION TABLE"
               ASC          "SRAB"
VOLMES         ASC          " EMULOV KSID" ; "DISK VOLUME " BACKWARDS "
               MSB          OFF
VML            EQU          *-VOLMES-1
               PAGE
;VTOC RECORD AREA
VTOC           EQU          *
VDOST          DFB          4                        ; DOS TYPE
VDIRTK         DFB          17                       ; COLUME DIRECTORY SECTOR
VDIRSC         DFB          15                       ; VOLUME DIRECTORY SECTOR
VDOSRN         DFB          4                        ; DOS RELEASE NUMBER
               DFB          0                        ; SPARE
               DFB          0                        ; SPARE
VVOLNO         DFB          $FE                      ; VOLUME NUMBER
               DS           32                       ; SPARE
VTDMS          DFB          122                      ; MAX SECTORS IN A FILE DIR
VSPARE         DS           8                        ; SPARES
;
VALCA1         DFB          17                       ; ALOCATION ALGORITHM BYTE 1
VALCA2         DFB          1                        ; AA BYTE2
VALCA3         DFB          0                        ; AA BYTE3
VALCA4         DFB          0                        ; AA BYTE4
VNOTRK         DFB          35                       ; NO TRACKS ON VOL
VNOSEC         DFB          16                       ; NO SECTORS PER TRACK
VSECLN         DW           256                      ; NO. BYTES PER SECTOR
;
VSECAL         EQU          *                        ; SECTOR ALLOCATION AREA
;SECTORS ALLOCATED BY BIT MAP
;4 BYTES OF BITS PER TRACK
;LEFT MOST BIT REPRESENTS SECTOR N
;WHERE N=NO SECTORS PER TRACK
;
;
               PAGE
               ORG          VTOC+256
;
;VOLUME DIRECTORY AREA
;
```

```
VOLDIR          EQU          *
VDTCDE          DFB          2                          ; VOLUME DIRECTORY TYPE CODE
VDLTRK          DS           1                          ; VD LINK TRACK
VDLSEC          DS           1                          ; VD LINK SECTOR
VDNF            DS           1                          ; VD NUMBER FILES THIS SECTOR
VDSPAR          DS           7                          ; SPARES
;
VDFILE          EQU          *                          ; FILE ALLOCATION AREA (7 FILES)
;EACH FILE:
;FILE DIR TRK
;FILE DIR SECTOR
;FILE USE CODE
;FILE NAME (30)
;FILE SECTOR COUNT (2)
                ORG          VOLDIR+256
VDEND           EQU          *
VDLEN           EQU          *-VOLDIR
VDFLEN          EQU          *-VDFILE
;
                PAGE
;
;COMMAND CONTROL BLOCK (CCB)
;
CCB             EQU          *
CCBREQ          DS           1                          ; USER REQUEST BYTE
CRQNUL          EQU          0                          ; 0-NO REQUEST
CRQOPN          EQU          1                          ; 1-OPEN FILE
CRQCLS          EQU          2                          ; 2-CLOSE FILE
CRQRD           EQU          3                          ; 3-READ DATA
CRQWR           EQU          4                          ; WRITE DATA
CRQDEL          EQU          5                          ; 5-DELETE FILE
CRQDIR          EQU          6                          ; 6-READ DIRECTORY
CRQLCK          EQU          7                          ; 7-LOCK FILE
CRQUNL          EQU          8                          ; 8-UNLOCK FILE
CRQRNM          EQU          9                          ; 9-RENAME
CRQPOS          EQU          10                         ; 10-POSITION FILE
CRQFMT          EQU          11                         ; 11-FORMAT
CRQVAR          EQU          12                         ; 12 - VERIFY
CRQMAX          EQU          13
;
CCBBSA          EQU          *                          ; FORMAT - BOOT START ADR PAGE
CCBRQM          DS           1                          ; RREQUEST MODIFIER BYTE
CRMNUL          EQU          0                          ; NO MODIFIER
CRMNBT          EQU          1                          ; R/W - 1 - NEXT BYTE
CRMNBL          EQU          2                          ; R/W - 2 - NEXT BLOCK
CRMSBT          EQU          3                          ; R/W - 3 - SPECIFC BYTE
CRMSBL          EQU          4                          ; R/W - 4 - SPECIFIC BLOCK
CRMMAX          EQU          5
;
CCBRRN          EQU          *                          ; I/O - RELATIVE RECORD NUMBER
CCBFN2          EQU          *                          ; RENAME - FILE NAME 2 PTR
CCBRLN          DS           2                          ; OPEN - RECORD LENGTH
;
CCBBYT          EQU          *                          ; I/O - RELATIVE BYTE NO (2 BYTES)
CCBVOL          DS           1                          ; OPEN - VOL NO.
CCBDRV          DS           1                          ; OPEN - DRIVE
;
CCBBLN          EQU          *                          ; I/O - BLOCK LENGTH (2 BYTES)
CCBSLT          DS           1                          ; OPEN - SLOT NO
CCBFUC          DS           1                          ; OPEN - FILE USE CODE
;
CCBFN1          EQU          *                          ; OPEN, DELETE, LOCK, UNLOCK, RENAME
- FILENAME P
```

```
CCBBBA          EQU         *                       ; BLOCKK I/O - BLOCK BUFFER PTR
CCBDAT          DS          2                       ; BYTE I/O - DATA BYTE
;
CCBSTA          DS          1                       ; RESULT STATUS
CREFUN          EQU         1                       ; FCB UNALLOCATED
CRERR           EQU         2                       ; CCB REQ RANGE ERR
CREMRE          EQU         3                       ; REQ MOD RANGE ERR
CREPRO          EQU         4                       ; WRITE PROTECT
CREEOF          EQU         5                       ; END OF FILE ON READ
CREFNF          EQU         6                       ; FILE NOT FOUND
CREVMM          EQU         7                       ; VOL MIS MATCH
CREIOE          EQU         8                       ; I/O ERR
CRENSA          EQU         9                       ; NO SECTORS AVAILABLE
CREFLK          EQU         10                      ; FILE LOCKED
;
CCBSM           DS          1                       ; STATUS MODIFIER
CCBFCB          DS          2                       ; FCB PTR
CCBDBP          DS          2                       ; DIR BUF PTR
CCBSBP          DS          2                       ; SECTOR BUF PTR
CCBSPR          DS          4                       ; SPARE
CCBLEN          EQU         *-CCB                   ; CCB LENGTH
CFCBAD          EQU         CCBFCB
CFCBDR          EQU         CCBDBP
CFCBSB          EQU         CCBSBP
                PAGE
;
;FILE CONTROL BLOCK (FCB) DEFINITION
;DCB - FILE DATA CONTROL BLOCK
;
FCB             EQU         *
;
;DATA CONTROL BLOCK
;
FCBDCB          EQU         *
DCBFDT          DS          1                       ; 1ST FILE DIRECTORY TRACK
DCBFDS          DS          1                       ; 1ST FILE DIRECTORY SECTOR
DCBCDT          DS          1                       ; CURRENT FILE DIRECTORY TRACK
DCBCDS          DS          1                       ; CURRENT FILE DIRECTORY SECTOR
DCBWRF          DS          1                       ; WRITE REQD FLAG
;$80=WRITE FILE DIR
;$40=WRITE SECTOR DIR
DCBTRK          DS          1                       ; SECTOR TRACK ADR
DCBSEC          DS          1                       ; SECTOR ADR
DCBVDR          DS          1                       ; VOL DIR REC
DCBVDI          DS          1                       ; VOL DIR INDEX
DCBDMS          DS          2                       ; MAX NO DIRECTORY SECTORS
DCBDFS          DS          2                       ; CURRENT DIR 1ST REL SECTOR
DCBDNF          DS          2                       ; REL SECTOR OF NXT DIR
DCBCMS          DS          2                       ; SECTOR CURRENTLY IN MEMORY
DCBSDL          DS          2                       ; SECTOR DATA LENGTH
DCBCRS          DS          2                       ; CURRENT RELATIVE SECTOR
DCBCSB          DS          2                       ; CURRENT SECTOR BYTE
DCBRCL          DS          2                       ; RECORD LENGTH
DCBCRR          DS          2                       ; CURRENT RELATIVE REC
DCBCRB          DS          2                       ; CURRENT RELATIVE BYTE
DCBNSA          DS          2                       ; NO SECTORS ALLOCATED
;
DCBALS          DS          1                       ; ALLOCATION SECTOR BYTE
DCBATK          DS          1                       ; ALLOCATION TRACK
DCBABM          DS          4                       ; ALLOCATION TRACK SECTOR BIT MAP
;
DCBFUC          DS          1                       ; FILE USE CODE
DCBSLT          DS          1                       ; SLOT NUMBER
```

```
DCBDRV          DS          1                       ; DRIVE NUMBER
DCBVOL          DS          1                       ; VOLUME DRIVER
DCBVTN          DS          1                       ; VTOC TRACK NUMBER
;
DCBSPR          DS          3                       ; SPARES
;
DCBLEN          EQU         *-FCBDCB                ; DCB LENGTH
FCBLEN          EQU         *-FCB                   ; FCB LENGTH
;

; ###############################################################################
; #   END OF FILE:  FVCBUFS
; #   LINES      :  187
; #   CHARACTERS :  9792
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT HELLO.A.hex
================================================================================

File ............. "HELLO"
Fork ............. DATA
Size (bytes) ..... 193 (0KB) / $000000C1
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: 07080A00 97003308 1400A232 3A5324D0 [......3....2:S$.]
D/000010: 22415050 4C45205D 5B202044 4F532033 ["APPLE.][..DOS.3]
D/000020: 2E332020 534F5552 4345223A B0313030 [.3..SOURCE":.100]
D/000030: 30006808 1E00A234 3A5324D0 22284329 [0.h....4:S$."(C)]
D/000040: 20434F50 59524947 48542041 50504C45 [.COPYRIGHT.APPLE]
D/000050: 20434F4D 50555445 522C2049 4E432E22 [.COMPUTER,.INC."]
D/000060: 3AB03130 30300085 082800A2 363A5324 [:.1000...(..6:S$]
D/000070: D022564F 4C554D45 204F4E45 223AB031 [."VOLUME.ONE":.1]
D/000080: 30303000 95083200 8158D031 C1313030 [000...2..X.1.100]
D/000090: 303A8200 9E083C00 A2393ABF 00A408E7 [0:....<..9:.....]
D/0000A0: 038000BF 08E80396 283230C9 D328E328 [........(20..(.(]
D/0000B0: 532429CB 3229293A BA53243A B1000000 [S$).2)):.S$:....]
D/0000C0: 00                                  [.               ]

File ............. "HELLO"
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 HELLO.B.hex
================================================================================

File ............. "HELLO"
Fork ............. DATA
Size (bytes) ..... 193 (0KB) / $000000C1
Created .......... Wednesday, April 12, 2006 -- 9:20:30 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:30 AM

D/000000: 07080A00 97003308 1400A232 3A5324D0 [......3....2:S$.]
D/000010: 22415050 4C45205D 5B202044 4F532033 ["APPLE.][..DOS.3]
D/000020: 2E332020 534F5552 4345223A B0313030 [.3..SOURCE":.100]
D/000030: 30006808 1E00A234 3A5324D0 22284329 [0.h....4:S$."(C)]
D/000040: 20434F50 59524947 48542041 50504C45 [.COPYRIGHT.APPLE]
D/000050: 20434F4D 50555445 522C2049 4E432E22 [.COMPUTER,.INC."]
D/000060: 3AB03130 30300085 082800A2 363A5324 [:.1000...(..6:S$]
D/000070: D022564F 4C554D45 2054574F 223AB031 [."VOLUME.TWO":.1]
D/000080: 30303000 95083200 8158D031 C1313030 [000...2..X.1.100]
D/000090: 303A8200 9E083C00 A2393ABF 00A408E7 [0:....<..9:.....]
D/0000A0: 038000BF 08E80396 283230C9 D328E328 [........(20..(.(]
D/0000B0: 532429CB 3229293A BA53243A B1000000 [S$).2)):.S$:....]
D/0000C0: 00                                  [.               ]

File ............. "HELLO"
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 MAKE.MASTER.hex
================================================================================

File ............. "MAKE.MASTER"
Fork ............. DATA
Size (bytes) ..... 177 (0KB) / $000000B1
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: 18080200 BA3ABAE7 28342922 4D415846 [.....:..(4)"MAXF]
D/000010: 494C4553 33220020 080A0089 3A97002C [ILES3"......:..,]
D/000020: 081400A2 31323A96 31300044 081E00BA [....12:.10.D....]
D/000030: 22444F53 20332E33 4320544F 20444953 ["DOS.3.3C.TO.DIS]
D/000040: 4B22006F 082800BA 3A96333A BA224C45 [K".o.(..:.3:."LE]
D/000050: 41564520 54484953 20444953 4B20494E [AVE.THIS.DISK.IN]
D/000060: 20445249 56452055 4E54494C 22009208 [.DRIVE.UNTIL"...]
D/000070: 32009636 3ABA2249 4E535452 55435445 [2..6:."INSTRUCTE]
D/000080: 4420544F 2052454D 4F564520 49542E22 [D.TO.REMOVE.IT."]
D/000090: 00B0083C 00BA3ABA E7283429 22455845 [...<..:..(4)"EXE]
D/0000A0: 4320444F 5320544F 20444953 4B220000 [C.DOS.TO.DISK"..]
D/0000B0: 00                                  [.               ]

File ............. "MAKE.MASTER"
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.3.3E.hex
================================================================================

File ............. "MASTER.3.3E"
Fork ............. DATA
Size (bytes) ..... 1,885 (1KB) / $0000075D
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: 0D080500 4424D0E7 28342900 13080A00 [....D$..(4).....]
D/000010: 97004708 14009E3A BA224040 40404040 [..G....:."@@@@@@]
D/000020: 40404040 40404040 40404040 40404040 [@@@@@@@@@@@@@@@@]
D/000030: 40404040 40404040 40404040 40404040 [@@@@@@@@@@@@@@@@]
D/000040: 4040223A 9D007408 1E00A233 3A5324D0 [@@":..t....3:S$.]
D/000050: 22444F53 20332E33 45204D41 53544552 ["DOS.3.3E.MASTER]
D/000060: 494E4720 50524F47 52414D22 3AB03930 [ING.PROGRAM":.90]
D/000070: 30300092 082800A2 343A5324 D0225645 [00...(..4:S$."VE]
D/000080: 5253494F 4E20312E 30223AB0 39303030 [RSION.1.0":.9000]
D/000090: 00C50832 00A2363A BA222F2F 2F2F2F2F [...2..6:."//////]
D/0000A0: 2F2F2F2F 2F2F2F2F 2F2F2F2F 2F2F5C5C [////////////\\]
D/0000B0: 5C5C5C5C 5C5C5C5C 5C5C5C5C 5C5C5C5C [\\\\\\\\\\\\\\\\]
D/0000C0: 5C5C2200 E1083700 BA442422 424C4F41 [\\"...7..D$"BLOA]
D/0000D0: 44204D41 53544552 452E4F42 4A302200 [D.MASTERE.OBJ0".]
D/0000E0: 02093C00 A2393A84 224E414D 45204F46 [..<..9:."NAME.OF]
D/0000F0: 20444F53 2046494C 453F2022 3B4E4D24 [.DOS.FILE?.";NM$]
D/000100: 000E0946 00A5AB31 30303030 00210950 [...F...10000.!.P]
D/000110: 00BA4424 22424C4F 41442022 4E4D2400 [..D$"BLOAD."NM$.]
D/000120: 2C095A00 B9323232 2C300045 096400B0 [,.Z..222,0.E.d..]
D/000130: 31303030 3AB22047 45542049 4F422049 [1000:..GET.IOB.I]
D/000140: 4E464F00 5A096900 B0333030 303AB220 [NFO.Z.i..3000:..]
D/000150: 53415645 20494F42 0074096E 00B03230 [SAVE.IOB.t.n..20]
D/000160: 30303AB2 20504F4B 4520494F 4220494E [00:..POKE.IOB.IN]
D/000170: 464F0090 097000A2 373A8CC9 3935383A [FO...p..7:..958:]
D/000180: B220434C 45415220 544F2045 4F5300B9 [..CLEAR.TO.EOS..]
D/000190: 097100A2 31313A84 22505245 5353203C [.q..11:."PRESS.<]
D/0001A0: 52455455 524E3E20 544F2055 50444154 [RETURN>.TO.UPDAT]
D/0001B0: 453A2022 3B532400 D5097300 A2373A8C [E:.";S$...s..7:.]
D/0001C0: C9393538 3AB22043 4C454152 20544F20 [.958:..CLEAR.TO.]
D/0001D0: 454F5300 140A7500 A231313A 5324D022 [EOS...u..11:S$."]
D/0001E0: 55504441 54494E47 20444953 4B20494E [UPDATING.DISK.IN]
D/0001F0: 20534C4F 542022C8 E4285329 C8222C20 [.SLOT."..(S)",.]
D/000200: 44524956 452022C8 E4284429 3AB03930 [DRIVE."..(D):.90]
D/000210: 3030001F 0A7700B9 3232322C 30003D0A [00...w..222,0.=.]
D/000220: 78008C37 36383AB2 2043414C 4C205752 [x..768:..CALL.WR]
D/000230: 49544520 50524F47 52414D00 5D0A7900 [ITE.PROGRAM.].y.]
D/000240: ADE22832 323229D1 CF30C4B0 34303030 [..(222)..0..4000]
D/000250: 3AB22041 4E204552 524F5200 A70A8200 [:..AN.ERROR.....]
D/000260: A232303A 84225550 44415445 20414E4F [.20:."UPDATE.ANO]
D/000270: 54484552 3F20283C 52455455 524E3E3D [THER?.(<RETURN>=]
D/000280: 4E4F2920 223B5324 3AADC6E3 28532429 [NO)."";S$:...(S$)]
D/000290: C45324D0 224E223A A232303A 9633313A [.S$."N":.20:.31:]
D/0002A0: BA224E4F 2200BF0A 8C00AD53 24D02259 [."NO".....S$."Y]
D/0002B0: 22CE5324 D0227922 C4AB3131 3300D90A [".S$."y"..113...]
D/0002C0: 9600AD53 24D1CF22 4E22CD53 24D1CF22 [...S$.."N".S$.."]
D/0002D0: 6E22C4AB 31333000 F10AA000 B0333530 [n"..130......350]
D/0002E0: 303AB220 52455354 4F524520 494F4200 [0:..RESTORE.IOB.]
D/0002F0: F70AAA00 8000FD0A E7038000 130BE803 [................]
D/000300: B20A0A47 45542049 4F422049 4E464F0A [...GET.IOB.INFO.]
D/000310: 0A002F0B FC03A237 3A8CC939 35383AB2 [../....7:..958:.]
D/000320: 20434C45 41522054 4F20454F 53007A0B [.CLEAR.TO.EOS.z.]
D/000330: 0604A239 3A842253 4C4F5420 544F2042 [...9:."SLOT.TO.B]
```

```
D/000340: 45204D41 53544552 45443F20 283C5245 [E.MASTERED?.(<RE]
D/000350: 5455524E 3E3D3629 20223B53 243A53D0 [TURN>=6)."";S$:S.]
D/000360: E5285324 293AAD53 D030C453 D0363AA2 [.(S$):.S.0.S.6:.]
D/000370: 393A9633 353ABA53 00A70B10 04AD53D1 [9:.35:.S......S.]
D/000380: 31CE53CF 37C4A231 313ABAE7 28372922 [1.S.7..11:..(7)"]
D/000390: 494E5641 4C494420 4E554D42 4552223A [INVALID.NUMBER":]
D/0003A0: AB313033 3000F50B 1A04A231 313A8422 [.1030......11:."]
D/0003B0: 44524956 4520544F 20424520 4D415354 [DRIVE.TO.BE.MAST]
D/0003C0: 45524544 3F20283C 52455455 524E3E3D [ERED?.(<RETURN>=]
D/0003D0: 31292022 3B53243A 44D0E528 5324293A [1)."";S$:D..(S$):]
D/0003E0: AD44D030 C444D031 3AA23131 3A963336 [.D.0.D.1:.11:.36]
D/0003F0: 3ABA4400 220C2404 AD44D131 CE44CF32 [:.D.".$..D.1.D.2]
D/000400: C4A23133 3ABAE728 37292249 4E56414C [..13:..(7)"INVAL]
D/000410: 4944204E 554D4245 52223AAB 31303530 [ID.NUMBER":.1050]
D/000420: 00620C2E 04A23135 3A84224F 4B3F2028 [.b....15:."OK?.(]
D/000430: 3C524554 55524E3E 3D594553 2920223B [<RETURN>=YES)."";]
D/000440: 53243AAD C6E32853 2429C453 24D02259 [S$:...(S$).S$."Y]
D/000450: 223AA231 353A9632 303ABA22 59455322 [":.15:.20:."YES"]
D/000460: 00860C38 045324D0 E8285324 2C31293A [...8.S$..(S$,1):]
D/000470: AD5324D0 224E22CE 5324D022 6E22C4AB [.S$."N".S$."n"..]
D/000480: 31303030 00A10C42 04AD5324 D1CF2259 [1000...B..S$.."Y]
D/000490: 22CD5324 D1CF2279 22C4AB31 30373000 [".S$.."y"..1070.]
D/0004A0: A70C4C04 B100BE0C D007B20A 0A504F4B [..L..........POK]
D/0004B0: 4520494F 4220494E 464F0A0A 00CC0CDA [E.IOB.INFO......]
D/0004C0: 07494F42 D0343730 383000F0 0CE407B9 [.IOB.47080......]
D/0004D0: 494F42C8 312C53CA 31363AB2 20504F4B [IOB.1,S.16:..POK]
D/0004E0: 4520534C 4F54204E 554D4245 52200011 [E.SLOT.NUMBER...]
D/0004F0: 0DEE07B9 494F42C8 322C443A B220504F [....IOB.2,D:..PO]
D/000500: 4B452044 52495645 204E554D 42455200 [KE.DRIVE.NUMBER.]
D/000510: 330DF807 B9494F42 C8332C30 3AB22050 [3....IOB.3,0:..P]
D/000520: 4F4B4520 564F4C55 4D45204E 554D4245 [OKE.VOLUME.NUMBE]
D/000530: 5200580D 0208B949 4F42C831 322C323A [R.X....IOB.12,2:]
D/000540: B220504F 4B452022 57524954 45222043 [..POKE."WRITE".C]
D/000550: 4F4D4D41 4E44005E 0D0C08B1 00700DB8 [OMMAND.^.....p..]
D/000560: 0BB20A0A 53415645 20494F42 0A0A007E [....SAVE.IOB...~]
D/000570: 0DC20B49 4F42D034 37303830 00990DCC [...IOB.47080....]
D/000580: 0B4F53D0 E228494F 42C83129 3AB2204F [.OS..(IOB.1):..O]
D/000590: 4C442053 4C4F5400 B50DD60B 4F44D0E2 [LD.SLOT.....OD..]
D/0005A0: 28494F42 C832293A B2204F4C 44204452 [(IOB.2):..OLD.DR]
D/0005B0: 49564500 BB0DE00B B100D00D AC0DB20A [IVE.............]
D/0005C0: 0A524553 544F5245 20494F42 0A0A00F5 [.RESTORE.IOB....]
D/0005D0: 0DB60DB9 494F42C8 312C4F53 3AB22050 [....IOB.1,OS:..P]
D/0005E0: 4F4B4520 4F4C4420 534C4F54 204E554D [OKE.OLD.SLOT.NUM]
D/0005F0: 42455200 1B0EC00D B9494F42 C8322C4F [BER......IOB.2,O]
D/000600: 443AB220 504F4B45 204F4C44 20445249 [D:..POKE.OLD.DRI]
D/000610: 5645204E 554D4245 5200210E CA0DB100 [VE.NUMBER.!.....]
D/000620: 360EA00F B20A0A57 52495445 20455252 [6......WRITE.ERR]
D/000630: 4F520A0A 00470EAA 0FA2373A 8CC93935 [OR...G....7:..95]
D/000640: 383AA231 3100540E B40F53D0 E2283232 [8:.11.T...S..(22]
D/000650: 322900AC 0EBE0FAD 53D03137 C45324D0 [2).......S.17.S$.]
D/000660: 22575249 54452050 524F5445 43542045 ["WRITE.PROTECT.E]
D/000670: 52524F52 223AB039 3030303A A231333A [RROR":.9000:.13:]
D/000680: 5324D022 504C4541 53452052 454D4F56 [S$."PLEASE.REMOV]
D/000690: 45205752 49544520 50524F54 45435420 [E.WRITE.PROTECT.]
D/0006A0: 54414222 3AB03930 303000F1 0EC80FAD [TAB":.9000......]
D/0006B0: 53D03430 C45324D0 22492F4F 20455252 [S.40.S$."I/O.ERR]
D/0006C0: 4F52223A B0393030 303AA231 333A5324 [OR":.9000:.13:S$]
D/0006D0: D022504C 45415345 20434845 434B2059 [."PLEASE.CHECK.Y]
D/0006E0: 4F555220 4449534B 223AB039 30303000 [OUR.DISK":.9000.]
D/0006F0: 1B0FD20F A231383A 84225052 45535320 [.....18:."PRESS.]
D/000700: 3C524554 55524E3E 20544F20 434F4E54 [<RETURN>.TO.CONT]
D/000710: 494E5545 20223B53 2400210F DC0FB100 [INUE.";S$.!.....]
D/000720: 330F2823 B20A0A20 43454E54 4552494E [3.(#....CENTERIN]
```

```
D/000730: 47004E0F 32239628 3230C9D3 28E32853 [G.N.2#.(20..(.(S]
D/000740: 2429CB32 29293ABA 53243AB1 005B0F10 [$).2)):.S$:..[..]
D/000750: 27BA2245 5252223A 80000000 0A       ['."ERR":.....   ]

File ............. "MASTER.3.3E"
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 MASTERE.OBJ0.hex
================================================================================

File ............. "MASTERE.OBJ0"
Fork ............. DATA
Size (bytes) ..... 122 (0KB) / $0000007A
Created .......... Wednesday, April 12, 2006 -- 9:20:26 AM
Modified ......... Wednesday, April 12, 2006 -- 9:20:26 AM

D/000000: A00098AA 8EF0B78E ECB78EED B7BD5403 [..............T.]
D/000010: 8DF1B78A 203603C0 0FD00818 69018DEC [.....6......i...]
D/000020: B7A0FFC8 8CEDB748 E8BD5403 F0078DF1 [.......H..T.....]
D/000030: B7684C14 03608E53 03A9B7A0 E820D903 [.hL..`.S........]
D/000040: ADF5B7D0 0485DE68 68ADECB7 ACEDB7AE [.......hh.......]
D/000050: 53036000 36373839 3A3B3C3D 3E3F1B1C [S.`.6789:;<=>?..]
D/000060: 1D1E1F20 21222324 25262728 292A2B2C [....!"#$%&'()*+,]
D/000070: 2D2E2F30 31323334 3500              [-./012345.      ]

File ............. "MASTERE.OBJ0"
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 MSWAITR.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  MSWAITR
; ############################################################################################

                SBTL        '16-SECTOR MSWAIT'
**************************
*   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    *
**************************
                DS          3,0                     ;AVOID PAGE BOUNDARY CROSSING...
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
AEC1            EQU         *                       ;TELL RELOCATOR WHERE CORE ENDS
**************************
*  PHASE ON-, OFF-TIME   *
*   TABLES IN 100-USEC   *
*   INTERVALS. (SEEK)    *
**************************
ONTABLE         DFB         1,$30,$28
                DFB         $24,$20,$1E
                DFB         $1D,$1C,$1C
                DFB         $1C,$1C,$1C
OFFTABLE        DFB         $70,$2C,$26
                DFB         $22,$1F,$1E
                DFB         $1D,$1C,$1C
```

```
                DFB         $1C,$1C,$1C
                SBTL        '16-SECTOR NYBBLE TABLES'
**************************
*                        *
*     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.  *
*                        *
*  THIS TABLE MAY *NOT*  *
*  CROSS A PAGE BOUNDARY!*
*                        *
**************************
NIBL            DFB         $96,$97,$9A
                DFB         $9B,$9D,$9E
                DFB         $9F,$A6,$A7
                DFB         $AB,$AC,$AD
                DFB         $AE,$AF,$B2
                DFB         $B3,$B4,$B5
                DFB         $B6,$B7,$B9
                DFB         $BA,$BB,$BC
                DFB         $BD,$BE,$BF
                DFB         $CB,$CD,$CE
                DFB         $CF,$D3,$D6
                DFB         $D7,$D9,$DA
                DFB         $DB,$DC,$DD
                DFB         $DE,$DF,$E5
                DFB         $E6,$E7,$E9
                DFB         $EA,$EB,$EC
                DFB         $ED,$EE,$EF
                DFB         $F2,$F3,$F4
                DFB         $F5,$F6,$F7
                DFB         $F9,$FA,$FB
                DFB         $FC,$FD,$FE
                DFB         $FF
                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.*
*                        *
* THIS TABLE *MUST* BE   *
* ALIGNED AT THE END OF  *
* A PAGE IN MEMORY!!!    *
**************************
XP              EQU         <*                      ;CURRENT PAGE ADDRESS
DNIBL           EQU         256*XP                  ;DNIBL TABLE PAGE
```

```
                PAGE
*********************************************
*
* GHOST APPEND BUG PATCH BY
* BILL GRIMM
*
*********************************************
PSC1            EQU         *                       ;Tell relocater where to start
MOVEOF          EQU         *
                LDX         CMDNO                   ; GET CMD NUMBER
                CPX         #$1C                    ; APPEND COMMAND?
                BEQ         GOON                    ; YES, RETURN TO CALLING ROUTINE
                LDX         #$00                    ; NO, THEN CLEAR X
                STX         EOFFLAG                 ; CLEAR EOF FLAG
GOON            RTS
                SKP         4
*********************************************
*
* TURN Apple //e 80 COLUMN CARD
* OFF & INIT APPLE
*
*********************************************
OFF80           EQU         *
                LDA         #$FF
                STA         $4FB                    ; CLEARS FUNNY 80 COL STUFF
                STA         $C00C                   ; TURNS 80 COL OFF
                STA         $C00E                   ; TURN OFF ALT CHAR SET
                JMP         $FB2F                   ; MONITOR INIT ROUTINE
                PAGE
PEC1            EQU         *                       ;Tell relocater where to stop
PD1             EQU         >*
PD2             EQU         $96-PD1
                DS          PD2,0                   ;Must pad to $XX96
                DFB         $00,$01,$98
                DFB         $99,$02,$03
                DFB         $9C,$04,$05
                DFB         $06,$A0,$A1
                DFB         $A2,$A3,$A4
                DFB         $A5,$07,$08
                DFB         $A8,$A9,$AA
                DFB         $09,$0A,$0B
                DFB         $0C,$0D,$B0
                DFB         $B1,$0E,$0F
                DFB         $10,$11,$12
                DFB         $13,$B8,$14
                DFB         $15,$16,$17
                DFB         $18,$19,$1A
                DFB         $C0,$C1,$C2
                DFB         $C3,$C4,$C5
                DFB         $C6,$C7,$C8
                DFB         $C9,$CA,$1B
                DFB         $CC,$1C,$1D
                DFB         $1E,$D0,$D1
                DFB         $D2,$1F,$D4
                DFB         $D5,$20,$21
                DFB         $D8,$22,$23
                DFB         $24,$25,$26
                DFB         $27,$28,$E0
                DFB         $E1,$E2,$E3
                DFB         $E4,$29,$2A
                DFB         $2B,$E8,$2C
                DFB         $2D,$2E,$2F
                DFB         $30,$31,$32
```

```
                DFB           $F0,$F1,$33
                DFB           $34,$35,$36
                DFB           $37,$38,$F8
                DFB           $39,$3A,$3B
                DFB           $3C,$3D,$3E
                DFB           $3F
                PAGE
***************************
*                         *
*    NYBBLE BUFFERS       *
*                         *
* NBUF1 (256 BYTES) MUST  *
*  BE ALIGNED ON A PAGE   *
*  BOUNDARY.              *
*                         *
* NBUF2 (86 BYTES) MUST   *
*  BE ALIGNED ON A PAGE   *
*  BOUNDARY.              *
*                         *
***************************
*
NBUF1           DS            256,0                     ;NBUF1
NBUF2           DS            86,0                      ;NBUF2

; ###########################################################################################
; #   END OF FILE:  MSWAITR
; #   LINES      :  202
; #   CHARACTERS :  7321
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT POSTNRD.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  POSTNRD
; ###########################################################################################

                SBTL         '16-SECTOR POSTNIBLIZE'
***************************
*                         *
*    POSTNIBLIZE SUBR     *
*    16-SECTOR FORMAT     *
*                         *
***************************
*                         *
*  CONVERTS 6-BIT NIBLS   *
*  OF FORM 00ABCDEF IN    *
*  NBUF1 AND NBUF2 INTO   *
*  256 BYTES OF USER      *
*  DATA IN BUF.           *
*                         *
*   ---- ON ENTRY ----    *
*                         *
*  X-REG: HOLDS SLOTNUM   *
*           TIMES $10.    *
*                         *
*  BUF IS 2-BYTE POINTER  *
*    TO 256 BYTES OF USER *
*    DATA TO BE CONVERTED *
*    TO 6-BIT NIBLS IN    *
*    NBUF1 AND NBUF2      *
*    PRIOR TO WRITE.      *
*                         *
*  T0 CONTAINS BYTE COUNT *
*    CODE 0 = 256 BYTES   *
*    CODE 1 =   1 BYTE    *
*    CODE 2 =   2 BYTER   *
*                         *
*    CODE 255=255 BYTES   *
*                         *
*   ---- ON EXIT -----    *
*                         *
*  A-REG UNCERTAIN.       *
*  Y-REG SAME AS T0.      *
*  X-REG UNCERTAIN.       *
*  CARRY SET.             *
*                         *
*  6-BIT NIBLS OF FORM    *
*    00ABCDEF IN NBUF1    *
*    AND NBUF2.           *
*      (342 NIBLS)        *
***************************
POSTNB16        LDY          #0                      USER DATA BUF IDX.
POST1           LDX          #$56                    INIT NBUF2 INDEX.
POST2           DEX          NBUF                    IDX $55 TO $0.
                BMI          POST1                   WRAPAROUND IF NEG.
                LDA          NBUF1,Y
                LSR          NBUF2,X                 SHIFT 2 BITS FROM
                ROL          A                       CURRENT NBUF2 NIBL
                LSR          NBUF2,X                 INTO CURRENT NBUF1
```

```
                ROL         A                       NIBL.
                STA         (BUF),Y                 BYTE OF USER DATA.
                INY         NEXT                    USER BYTE.
                CPY         T0                      DONE IF EQUAL T0.
                BNE         POST2
                RTS         RETURN.
                SBTL        '16-SECTOR READ'
*************************
*                       *
*     READ SUBROUTINE   *
*   (16-SECTOR FORMAT)  *
*                       *
*************************
*                       *
*    READS 6-BIT NIBLS  *
*     (00ABCDEF) INTO   *
*     NBUF1 AND NBUF2   *
*    CONVERTING 7-BIT   *
*     NIBLS TO 6-BIT    *
*    VIA 'DNIBL' TABLE  *
*                       *
*  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.      *
*                       *
*     NBUF1 AND NBUF2   *
*       HOLD 6-BIT NIBLS*
*       (00ABCDEF)      *
*                       *
*     USES TEMP 'IDX'.  *
*                       *
*   ---- CAUTION -----  *
*                       *
*        OBSERVE        *
*    'NO PAGE CROSS'    *
*      WARNINGS ON      *
*    SOME BRANCHES!!    *
*                       *
*   ---- ASSUMES ----   *
*                       *
*   1 USEC CYCLE TIME   *
*                       *
*************************
READ16          LDY         #$20                    'MUST FIND' COUNT.
RSYNC           DEY         IF                      CAN'T FIND MARKS
```

```
                BEQ         RDERR                   ;THEN EXIT WITH CARRY SET.
READ1           LDA         Q6L,X                   READ NIBL.
                BPL         READ1                   *** NO PAGE CROSS! ***
RSYNC1          EOR         #$D5                    DATA MARK 1?
                BNE         RSYNC                   LOOP IF NOT.
                NOP         DELAY                   BETWEEN NIBLS.
READ2           LDA         Q6L,X
                BPL         READ2                   *** NO PAGE CROSS! ***
                CMP         #$AA                    DATA MARK 2?
                BNE         RSYNC1                  (IF NOT, IS IT DM1?)
                LDY         #$56                    INIT NBUF2 INDEX.
*              (ADDED NIBL DELAY)
READ3           LDA         Q6L,X
                BPL         READ3                   *** NO PAGE CROSS! ***
                CMP         #$AD                    DATA MARK 3?
                BNE         RSYNC1                  (IF NOT, IS IT DM1?)
*         (CARRY SET IF DM3!)
                LDA         #$00                    INIT CHECKSUM.
RDATA1          DEY
                STY         IDX
READ4           LDY         Q6L,X
                BPL         READ4                   *** NO PAGE CROSS! ***
                EOR         DNIBL,Y                 XOR 6-BIT NIBL.
                LDY         IDX
                STA         NBUF2,Y                 STORE IN NBUF2 PAGE.
                BNE         RDATA1                  TAKEN IF Y-REG NONZERO.
RDATA2          STY         IDX
READ5           LDY         Q6L,X
                BPL         READ5                   *** NO PAGE CROSS! ***
                EOR         DNIBL,Y                 XOR 6-BIT NIBL.
                LDY         IDX
                STA         NBUF1,Y                 STORE IN NBUF1 PAGE.
                INY
                BNE         RDATA2
READ6           LDY         Q6L,X                   READ 7-BIT CSUM NIBL.
                BPL         READ6                   *** NO PAGE CROSS! ***
                CMP         DNIBL,Y                 IF LAST NBUF1 NIBL NOT
                BNE         RDERR                   EQUAL CHKSUM NIBL THEN ERR.
READ7           LDA         Q6L,X
                BPL         READ7                   *** NO PAGE CROSS! ***
                CMP         #$DE                    FIRST BIT SLIP MARK?
                BNE         RDERR                   (ERR IF NOT)
                NOP         DELAY                   BETWEEN NIBLS.
READ8           LDA         Q6L,X
                BPL         READ8                   *** NO PAGE CROSS! ***
                CMP         #$AA                    SECOND BIT SLIP MARK?
                BEQ         RDEXIT                  (DONE IF IT IS)
RDERR           SEC         INDICATE                'ERROR EXIT'.
                RTS         RETURN                  FROM READ16 OR RDADR16.

; ##########################################################################################
; #   END OF FILE:  POSTNRD
; #   LINES      :  165
; #   CHARACTERS :  6677
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##########################################################################################
```

```
================================================================================
DOCUMENT PRENIBL.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  PRENIBL
; ############################################################################################

                SBTL         '16-SECTOR PRENIBLIZE'
****************************
*                          *
*    PRENIBLIZE SUBR       *
*   (16-SECTOR FORMAT)     *
*                          *
****************************
*                          *
*  CONVERTS 256 BYTES OF   *
*  USER DATA IN (BUF),0    *
*  TO (BUF),255 INTO 342   *
*  6-BIT NIBLS (00ABCDEF)  *
*  IN NBUF1 AND NBUF2.     *
*                          *
*    ---- ON ENTRY ----    *
*                          *
*  BUF IS 2-BYTE POINTER   *
*    TO 256 BYTES OF USER  *
*    DATA.                 *
*                          *
*    ---- ON EXIT -----    *
*                          *
*  A-REG UNCERTAIN.        *
*  X-REG HOLDS $FF.        *
*  Y-REG HOLDS $FF.        *
*  CARRY SET.              *
*                          *
*  NBUF1 AND NBUF2 CONTAIN *
*    6-BIT NIBLS OF FORM   *
*    00ABCDEF.             *
*                          *
****************************
PRENIB16        LDX          #$0                       ;START NBUF2 INDEX. CHANGED BY WOZ
                LDY          #2                        ;START USER BUF INDEX. CHANGED BY
WOZ.
PRENIB1         DEY          NEXT                      ;USER BYTE.
                LDA          (BUF),Y
                LSR          A                         ;SHIFT TWO BITS OF
                ROL          NBUF2,X                   ;CURRENT USER BYTE
                LSR          A                         ;INTO CURRENT NBUF2
                ROL          NBUF2,X                   ;BYTE.
                STA          NBUF1,Y                   ;(6 BITS LEFT).
                INX                                    ;FROM 0 TO $55.
                CPX          #$56
                BCC          PRENIB1                   ;BR IF NO WRAPAROUND.
                LDX          #0                        ;RESET NBUF2 INDEX.
                TYA                                    ;USER BUF INDEX.
                BNE          PRENIB1                   ;(DONE IF ZERO)
                LDX          #$55                      ;NBUF2 IDX $55 TO 0.
PRENIB2         LDA          NBUF2,X
                AND          #$3F                      ;STRIP EACH BYTE
                STA          NBUF2,X                   ;OF NBUF2 TO 6 BITS.
                DEX
```

```
                BPL          PRENIB2                    ;LOOP UNTIL X NEG.
                RTS                                     ;RETURN.

; ###########################################################################################
; #   END OF FILE:  PRENIBL
; #   LINES      :  54
; #   CHARACTERS :  2360
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT RDADSEK.pretty
================================================================================

; ###############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  RDADSEK
; ###############################################################################

                SBTL         '16-SECTOR READ ADDRESS'
*****************************
*                           *
*    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?)
*        (LEAVES CARRY SET!)
                LDA         #$0                     INIT CHECKSUM.
RDAFLD          STA         CSUM
RDA4            LDA         Q6L,X                   READ 'ODD BIT' NIBL.
                BPL         RDA4                    *** NO PAGE CROSS! ***
                ROL         A                       ;ALIGN ODD BITS, '1' INTO LSB.
                STA         LAST                    (SAVE THEM)
RDA5            LDA         Q6L,X                   READ 'EVEN BIT' NIBL.
                BPL         RDA5                    *** NO PAGE CROSS! ***
                AND         LAST                    MERGE ODD AND EVEN BITS.
                STA         CSSTV,Y                 STORE DATA BYTE.
                EOR         CSUM                    XOR CHECKSUM.
                DEY
                BPL         RDAFLD                  LOOP ON 4 DATA BYTES.
                TAY         IF                      FINAL CHECKSUM
                BNE         RDERR                   NONZERO, THEN ERROR.
RDA6            LDA         Q6L,X                   FIRST BIT-SLIP NIBL.
                BPL         RDA6                    *** NO PAGE CROSS! ***
                CMP         #$DE
                BNE         RDERR                   ERROR IF NONMATCH.
                NOP         DELAY                   BETWEEN NIBLS.
RDA7            LDA         Q6L,X                   SECOND BIT-SLIP NIBL.
                BPL         RDA7                    *** NO PAGE CROSS! ***
                CMP         #$AA
                BNE         RDERR                   ERROR IF NONMATCH.
RDEXIT          CLC         CLEAR                   CARRY ON
                RTS         NORMAL                  READ EXITS.
                SBTL        '16-SECTOR SEEK'
*****************************
*                           *
*  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            STX         SLOTTEMP                ;SAVE X-REG
                STA         TRKN                    ;SAVE TARGET TRACK
                CMP         CURTRK                  ON DESIRED TRACK?
                BEQ         SEEKRTS                 ;YES, 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         #$FE                    CALC TRKS TO GO.
                DEC         CURTRK                  DECR CURRENT TRACK (OUT).
MINTST          CMP         TRKCNT
                BCC         MAXTST                  AND 'TRKS MOVED'.
                LDA         TRKCNT
MAXTST          CMP         #$C
                BCS         STEP2                   ;IF TRKCNT>$B LEAVE Y ALONE (Y=$B).
STEP            TAY                                 ;ELSE SET ACCELERATION INDEX IN Y
STEP2           EQU         *
                SEC                                 ;CARRY SET=PHASE ON
                JSR         SETPHASE                ;PHASE ON
                LDA         ONTABLE,Y               FOR 'ONTIME'.
                JSR         MSWAIT                  (100 USEC INTERVALS)
*
```

```
                LDA         PRIOR
                CLC                                 ;CARRY CLEAR=PHASE OFF
                JSR         CLRPHASE                ;PHASE OFF
                LDA         OFFTABLE,Y              THEN WAIT 'OFFTIME'.
                JSR         MSWAIT                  (100 USEC INTERVALS)
                INC         TRKCNT                  'TRACKS MOVED' COUNT.
                BNE         SEEK2                   (ALWAYS TAKEN)
*
SEEKEND         EQU         *                       ;END OF SEEKING
                JSR         MSWAIT                  ;A=0: WAIT 25 MS SETTLE
                CLC                                 ; AND TURN OFF PHASE
*
* TURN HEAD STEPPER PHASE ON/OFF
*
SETPHASE        EQU         *
                LDA         CURTRK                  ;GET CURRENT PHASE
CLRPHASE        EQU         *
                AND         #3                      ;MASK FOR 1 OF 4 PHASES
                ROL         A                       ;DOUBLE FOR PHASE INDEX
                ORA         SLOTTEMP
                TAX
                LDA         PHASEOFF,X              ;FLIP THE PHASE
                LDX         SLOTTEMP                ;RESTORE X-REG
SEEKRTS         RTS                                 ;AND RETURN

; ###########################################################################################
; #   END OF FILE:  RDADSEK
; #   LINES      :  203
; #   CHARACTERS :  8943
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT RELOCTR.pretty
================================================================================

; ###############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  RELOCTR
; ###############################################################################################

                PAGE
*****************************************************
*                                                   *
* (C) COPYRIGHT 1978,1980,1982 APPLE COMPUTER, INC. *
*                                                   *
*****************************************************
                SKP          2
*********************************
*                               *
*  ADAPTED FOR MACRO EDASM BY   *
*        JOHN ARKLEY            *
*         DEC  1980             *
*                               *
*********************************
                SKP          2
*********************************
*                               *
*  DOS 3.3 REVISION B PATCHES   *
*   INSTALLED BY MARK HOUDE     *
*          JUL 1982             *
*                               *
*********************************
                SKP          2
*********************************
*                               *
*  DOS 3.3 REV B PATCHES VER 2  *
*   INSTALLED BY FERN BACHMAN   *
*          SEP 1982             *
*                               *
*********************************
                SKP          2
*********************************
*                               *
*    DOS33C PATCHES (APPEND &   *
*     UPPER/LOWER CASE CHECK)   *
*                               *
*            BY                 *
*        GUIL  BANKS            *
*         1983             *
*                               *
*********************************
;
;EQUATES REQD TO FIND THINGS IN APPLE II
;
SETVID          EQU          $FE93
SETKBD          EQU          $FE89
PROMPT          EQU          $33                    ; PROMPT CHAR
OUTSW           EQU          $36                    ; OUTPUT VECTOR SWITCH
INSW            EQU          $38                    ; INPUT VECTOR SWITCH
ZPGWRK          EQU          $40                    ; ZERO PAGE WORK CELL
CNUM            EQU          $44                    ; CONVERTED NUMERIC
LBUFF           EQU          $200                   ; LINE BUFFER
MULT            EQU          $FB63                  ; MULT ROUTINE
```

```
INPRT           EQU         $FE8B                   ; SET IN PORT
OUTPRT          EQU         $FE95                   ; SET OUT PORT
IBCHN           EQU         $E836                   ; BASIC RUN
IBLMEM          EQU         $4A                     ; BASIC LOW MEM
IBHMEM          EQU         $4C                     ; INTEGER BASIC HIMEM
IBSOP           EQU         $CA                     ; INTEGER BASIC START OF CGM
IBBRK           EQU         $E3E3                   ; BASIC BREAK
IBGO            EQU         $E000                   ; BASIC ENTRY POINT
IBCONT          EQU         $E003                   ; BASIC CONTINUE ENTRY POINT
IBSOV           EQU         $CC                     ; BASIC START OF VARIABLES
ASSOP           EQU         $67                     ; AS START OF PROGRAM
ASEOP           EQU         $AF                     ; AS END OF PROGRAM
ASEOP2          EQU         $69                     ; AS END-OF PGM 2
ASHM1           EQU         $73                     ; AS HIGH MEM 1
ASHM2           EQU         $6F                     ; AS HIGH MEM 2
ASRNX           EQU         $D6                     ;AS RUN-ONLY FLAG
ASONERR         EQU         $D8                     ;AS ON-ERR GOTO FLAG
ASLMEM          EQU         ASSOP                   ; AS LOW MEM
ASBRK1          EQU         $D865                   ; AS ROM BREAK
ASBRK2          EQU         $1067                   ; AS RAM BREAK
ASCNTU1         EQU         $D43C
ASCNTU2         EQU         $C3C
ASRSEQ1         EQU         $D4F2
ASRSEQ2         EQU         $CF2
AITSTL          EQU         $E000                   ; AS 1 IB TEST LOC
ATSTV           EQU         $4C                     ; AS TEST VALUE
ITSTV           EQU         $20                     ; IB TEST VALUE
BOOTSL          EQU         $2E                     ; BOOT FROM SLOT
ZPGFCB          EQU         $42                     ; ZERO PAGE WORK CELL
MONRST          EQU         $FF65                   ;MONITOR RESET ENTRY
MONBRK          EQU         $FA59                   ;MONITOR BREAK FUNCTION
IORTS           EQU         $FF58                   ;KNOWN RTS IN MONITOR ROM
HOME            EQU         $FC58
PRINT           EQU         $FDED
GETKEY          EQU         $FD0C
INSDS2          EQU         $F88E
LENGTH          EQU         $2F
ZRSET           EQU         $3F2                    ;NEW MONITOR ROM RESET VECTOR
PWCNST          EQU         $3F4                    ;NEW MONITOR ROM POWER UP CONSTANT
                REP         40
*
*
                ORG         ORIGIN
*
*
                REP         40
                PAGE
BEGIN           JMP         DBINIT
;
DOSREL          EQU         *
;
;GET RELOCATION PARMS
;
DR0             EQU         *
LOC1            EQU         $26
                LDA         #$BF                    ; START AT BF00
                STA         ZPGWRK+1                ; TO LOOK FOR
                LDX         #0                      ; HIGH RAM
                STX         ZPGWRK
DR0A            LDY         #0                      ; APPLE TEST
DR1B            EQU         *
                LDA         (ZPGWRK,X)
                STA         LOC1
```

```
DR1             TYA
                EOR         LOC1
                STA         LOC1
                TYA
                EOR         (ZPGWRK,X)
                STA         (ZPGWRK,X)
                CMP         LOC1
                BNE         DR1A
                INY
                BNE         DR1
                BEQ         DR2                     ; BR IF TOOK
DR1A            EQU         *
                DEC         ZPGWRK+1                ; NOT RAM
                BNE         DR0A                    ; TRY NEXT PAGE
;
DR2             EQU         *
;
                LDA         ZPGWRK+1                ;BEGIN PATCH TO INSURE   *****
                AND         #$DF                    ; PROPER HIGH MEMORY CHECK.
                STA         ZPGFCB+1                ;(DOS MASTER 3.1 CONTAINS
                STX         ZPGFCB                  ; THIS ROUTINE STARTING AT LOCATION
                LDA         (ZPGFCB,X)              ; $3540)
                PHA
                STA         LOC1                    ;SAVE TEST VALUE
DR2A            TYA                                 ;(FIRST TIME Y=0)
                EOR         LOC1                    ;TEST EACH (ALLEDGED) MEMORY BYTE
                STA         LOC1                    ; 256 TIMES TO DETERMINE IF
                TYA                                 ; IT IS REALLY GOOD MEMORY AND
                EOR         (ZPGWRK,X)              ; MIRRORED 8K LOWER IN RAM.
                STA         (ZPGFCB,X)
                CMP         LOC1                    ;DID IT PASS THIS TIME?
                BNE         DR2B                    ; BYTE NOT MIRRORED, THEN GOOD
MEMORY
                INY                                 ;MAYBE IT WAS COINCIDENCE
                BNE         DR2A                    ;BRANCH UNLESS IT'S MATCHED 256
TIMES
                LDY         ZPGFCB+1                ;HIMEM IS 8K LOWER THAN WAS
                PLA                                 ;ORIGINALLY THOUGHT!
                JMP         DR2C
DR2B            PLA                                 ;ORIGINAL HIMEM PROVED GOOD
                STA         (ZPGFCB,X)              ;RESTORE BYTE ORIGINALLY MESSED
WITH.
                LDY         ZPGWRK+1                ;END OF PATCH   *****
DR2C            INY                                 ; NEW END OF DOS
                STY         NEPAGE
                SEC
                TYA
                SBC         DOSLNG                  ; MINUS DOS LENGTH
                STA         NSPAGE                  ; IS NEW START OF DOS
                SEC
                SBC         RSPAGE                  ; MINUS OLD DOS START
                BEQ         BEGIN                   ; (BREIF NO DELTA)
                STA         DELTA                   ; IS DELTA
                LDA         RSPAGE                  ; RESET START PAGE TO NORMAL
                STA         ASTART+1
;
                LDA         #<DBINIT                ; RESET PI RTN TO NORMAL
                STA         DI3+2
                LDA         #>DBINIT
                STA         DI3+1
;
;RELOCATE ADR TABLES
;
```

```
                LDX         #0
                STX         ZPGWRK
DR3             EQU         *
                LDA         ADRTAB+1,X
                TAY
                LDA         ADRTAB+2,X
                STA         ZPGWRK+1
                JMP         DR5
;
DR4             EQU         *
                CLC
                LDA         (ZPGWRK),Y
                ADC         DELTA
                STA         (ZPGWRK),Y
                INY
                BNE         DR5
                INC         ZPGWRK+1
DR5             INY
                BNE         DR6
                INC         ZPGWRK+1
;
DR6             EQU         *
                LDA         ZPGWRK+1
                CMP         ADRTAB+4,X
                BCC         DR4
                TYA
                CMP         ADRTAB+3,X
                BCC         DR4
;
                TXA
                CLC
                ADC         #4
                TAX
                CPX         ADRTAB
                BCC         DR3
                PAGE
;
;RELOCATE CODE
;
                LDX         #0
DR7             STX         TEMP1
;
                LDA         CDETAB+1,X              ; GET A START OF CODE ADR
                STA         ZPGWRK                  ; PUT IN ZPG
                LDA         CDETAB+2,X
                STA         ZPGWRK+1
;
DR8             LDX         #0
                LDA         (ZPGWRK,X)              ; GET OP CODE
                JSR         INSDS2                  ; GO FIND OUT HOW LONG
;
                LDY         LENGTH                  ; GET HOW LONG
                CPY         #2                      ; IF IT AIN'T
                BNE         DR9                     ; 3 THEN DON'T RELOC
                LDA         (ZPGWRK),Y              ; GET PAGE FROM INST
                CMP         RSPAGE                  ; IF PAGE < REL START
                BCC         DR9                     ; THEN IGNOR
                CMP         REPAGE                  ; IF PAGE >= REL END
                BCS         DR9                     ; THEN IGNORE
                ADC         DELTA                   ; ELSE ADD DELTA
                STA         (ZPGWRK),Y              ; TO RELOCATE
;
DR9             SEC
```

```
                LDA         LENGTH                  ; ADD LENGTH
                ADC         ZPGWRK                  ; TO PC
                STA         ZPGWRK
                LDA         #0
                ADC         ZPGWRK+1
                STA         ZPGWRK+1
;
                LDX         TEMP1                   ; CHECK FOR END
                CMP         CDETAB+4,X              ; OF CODE SEGMENT
                BCC         DR8                     ; BR NOT END
                LDA         ZPGWRK
                CMP         CDETAB+3,X
                BCC         DR8                     ; BR NOT END
;
                TXA
                CLC
                ADC         #4                      ; INCREMENT TABLE INDEX
                TAX
                CPX         CDETAB                  ; DONE
                BCC         DR7                     ; BR IF NOT
;
                PAGE
;
;MOVE TO RELOCATED CODE
;
                LDA         #<ENDOFDOS-$80
                STA         ZPGWRK+1                ; ZPGWRK=FROM
                LDY         NEPAGE
                DEY
                STY         ZPGFCB+1                ; ZPGFCB = TOO
                LDA         #0
                STA         ZPGWRK
                STA         ZPGFCB
                TAY
;
DR10            LDA         (ZPGWRK),Y              ; BYTE FROM
                STA         (ZPGFCB),Y              ; BYTE TO
                INY                                 ; INCREMENT
                BNE         DR10                    ; BR NOT FULL PAGE
                DEC         DPGCNT                  ; DECREMENT PAGE CNT
                BEQ         DR11                    ; BR IF DONE
                DEC         ZPGWRK+1                ; INC FROM PAGE
                DEC         ZPGFCB+1                ; INC TOO PAGE
                BNE         DR10                    ; MOVE PAGE
;
DR11            JMP         DBVECT+3                ; DONE
                PAGE
ADRTAB          DFB         9*4
                DW          SAT1
                DW          EAT1
                DW          RUN
                DW          RUN+2
                DW          IBVT+2
                DW          IBVT+4
                DW          AS1VT
                DW          AS1VT+4
                DW          AS2VT
                DW          AS2VT+4
                DW          AS2VT+6
                DW          AS2VT+8
                DW          SAT2
                DW          EAT2
                DW          BAIOB
```

```
                DW          ADOSLD+2
                DW          IBDCTP
                DW          IBDCTP+2
                DFB         0,0,0,0
                DFB         0,0,0,0
                DFB         0,0,0,0
CDETAB          EQU         *
                DFB         8*4
                DW          SC1
                DW          EC1
                DW          SC2
                DW          EC2
                DW          SC3
                DW          EC3
                DW          SWADR1
                DW          EWADR1
                DW          ASC1
                DW          AEC1
                DW          PSC1
                DW          PEC1
                DW          ASC2
                DW          AEC2
                DW          SDP1
                DW          EDP1
RSPAGE          DFB         <START
REPAGE          DFB         <ENDOFDOS
;
NSPAGE          DFB         0
NEPAGE          DFB         0
;
DOSLNG          DFB         <ENDOFDOS-START
;
DELTA           DFB         0
DPGCNT          DFB         <ENDOFDOS-START
                PAGE

; ###############################################################################################
; #   END OF FILE:  RELOCTR
; #   LINES      :  337
; #   CHARACTERS :  14207
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT RWTSONE.pretty
================================================================================

; ############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  RWTSONE
; ############################################################################################

                SBTL          '16-SECTOR RWTS'
*****************************
*         DISK II           *
* READ/WRITE TRACK-SECTOR   *
*                           *
*   COPYRIGHT 1978 BY       *
*  APPLE COMPUTER, INC.     *
*                           *
*  ALL RIGHTS RESERVED      *
*                           *
*      R. WIGGINTON         *
*                           *
*  MODIFIED: 07/13/79       *
*     R. AURICCHIO          *
*  ADDED WAIT-SEEK WHEN     *
*   POWERING-UP MOTOR.      *
*                           *
*  MODIFIED: 10/25/79       *
*     R. AURICCHIO          *
*  ADDED DIAGMODE DISPLAYS  *
*   FOR ACTIVITY ANALYSIS.  *
*****************************
ASC2            EQU           *                         ;TELL RELOCTR WHERE RWTS BEGINS
*********************************
MOTOROFF        EQU           $C088
MOTORON         EQU           $C089
DRV1EN          EQU           $C08A
DRV2EN          EQU           $C08B
*****************************
* STATE MACHINE CONTROLS
* Q6   Q7   FUNCTION
* --   --   --------
* LO   LO   READ
* HI   LO   SENSE WRITE PROTECT
* LO   HI   WRITE
* HI   HI   WRITE LOAD
*****************************
DRV1TRK         EQU           $478
DRV2TRK         EQU           $4F8
IOBPL           EQU           $48
IOBPH           EQU           $49
SLOT            EQU           $5F8                      ;HOLDS SLOT NUM USED
PTRSDEST        EQU           $3C
DEVCTBL         EQU           PTRSDEST
DRIVNO          EQU           $35
MONTIME         EQU           $46
SECT            EQU           CSSTV+1
TRACK           EQU           CSSTV+2
VOLUME          EQU           CSSTV+3
MAXSEEKS        EQU           4                         ;MAX FOR SEEKCNT
SEEKCNT         EQU           $4F8                      ;# RESEEKS BEFORE RECALIBRATE
RETRYCNT        EQU           $578
RECALCNT        EQU           $6F8                      ;# RECALIBRATES -1
```

```
               PAGE
               LST         OFF
               DO          DIAGMODE
*
* DIAGMODE EQUATES...
*
SP             EQU         $A0                    ;SPACE (INDICATOR OFF)
TD             EQU         4                      ;DISPL BETWEEN CHARS
TC1            EQU         $1E                    ;^ - RWTS ACTIVE
TL1            EQU         $07D0
TC2            EQU         $0D                    ;M - MOTOR STARTUP
TL2            EQU         TL1+TD
TC3            EQU         $13                    ;S - SEEK IN PROGRESS
TL3            EQU         TL2+TD
TC4            EQU         $01                    ;A - READING ADDRESS
TL4            EQU         TL3+TD
TC5            EQU         $0C                    ;L - NOT DESIRED SECTOR (LATENCY)
TL5            EQU         TL4+1
TC6            EQU         $05                    ;E - ADDRESS ERROR
TL6            EQU         TL5+1
TC7            EQU         $10                    ;P - PRENIBBILIZING
TL7            EQU         TL4+TD
TC8            EQU         $17                    ;W - WRITING
TL8            EQU         TL7+TD
TC9            EQU         $12                    ;R - READING
TL9            EQU         TL8+TD
TC10           EQU         $05                    ;E - READ ERROR
TL10           EQU         TL9+TD
TC11           EQU         $0F                    ;O - POSTNIBBLIZING
TL11           EQU         TL10+TD
*
SCOUNT         EQU         $2FE                   ;SECTORS-ACCESSED COUNT
LCOUNT         EQU         $2FF                   ;LATENCY COUNT
TL12           EQU         TL11+TD                ;LATENCY POSITION
               PAGE
               FIN
               LST         ON
*************************
*                       *
*     READ/WRITE A      *
*   TRACK AND SECTOR    *
*                       *
*************************
*                       *
*   ENTER WITH A & Y    *
* REGISTERS POINTING TO *
* THE I/O CONTROL BLOCK *
* (THE 'IOB'). INSIDE   *
* THE IOB:              *
*                       *
* IBTYPE: IOB TYPE CODE *
*       (SHOULD BE A 01)*
*                       *
* IBSLOT: CONTROLLER SLOT*
*       NUMBER FOR THIS *
*       ACCESS.         *
*                       *
* IBDRVN: DRIVE NUMBER  *
*       FOR THIS ACCESS *
*                       *
* IBVOL: EXPECTED VOLUME *
*      NUMBER. NOTE THAT *
*      VOLUME 00 MATCHES *
```

```
*      ANY VOLUME NUMBER *
*                        *
* IBTRK: TRACK TO USE    *
*      THIS ACCESS       *
*                        *
* IBSECT: SECTOR NUMBER  *
*       TO USE THIS TIME *
*                        *
* IBDCTP: POINTER TO THE *
*       DEVICE CHARACTER-*
*       ISTICS TABLE.    *
*                        *
* IBBUFP: POINTER TO THE *
*       PLACE THE DATA IS*
*       OR SHOULD BE.    *
*                        *
* IBDLEN: AMOUNT OF DATA *
*       IN BYTES TO BE   *
*       PROCESSED.       *
*                        *
* IBCMD: COMMAND CODE:   *
*      0-> NULL COMMAND  *
*      1-> READ SECTOR   *
*      2-> WRITE SECTOR  *
*      4-> FORMAT DISK   *
*                        *
* IBSTAT: ERROR CODE:    *
*      0-> NO ERROR      *
*    $10-> WRITE PROTECT *
*    $20-> VOLUME ERROR  *
*    $40-> DRIVE ERROR   *
*    $80-> READ ERROR    *
*                        *
* IBSMOD: LOCATION TO    *
*       RETURN THE VOLUME*
*       NUMBER ACTUALLY  *
*       FOUND.           *
*                        *
* IOBPSN: PREVIOUS SLOT  *
*      NUMBER USED LAST  *
*      ACCESS.           *
*                        *
* IOBPDN: PREVIOUS DRIVE *
*      NUMBER USED LAST  *
*      ACCESS.           *
*                        *
**************************
*                        *
* DEVICE CHARACTERISTICS *
* TABLE DESCRIPTION:     *
*                        *
* DEVICE TYPE CODE       *
*  (ZERO FOR DISK II)    *
*                        *
* NUMBER OF PHASES PER   *
* TRACK (TWO FOR DISK II)*
*                        *
* MOTOR ON TIME IN 100   *
* MICROSECOND INTERVALS  *
* COMPLEMENTED. ($D8EF   *
* FOR DISK II)           *
*                        *
**************************
```

```
                PAGE
RWTS            STY         IOBPL                   ;UPON ENTRY, A&Y POINT AT THE
                STA         IOBPH                   ;I/O CONTROL BLOCK (IOB)
                LST         OFF
                DO          DIAGMODE
                LDY         #TC1                    ;SAY WE'RE ACTIVE
                STY         TL1
                FIN
                LST         ON
                LDY         #2                      ;SET RECALIBRATE
                STY         RECALCNT                ; COUNT
                LDY         #MAXSEEKS               ;SET RESEEK
                STY         SEEKCNT                 ; COUNT
                LDY         #1                      ;GET SLOT # FOR THIS OPERATION
                LDA         (IOBPL),Y
                TAX
                LDY         #$0F                    ;DID HE CHANGE SLOTS?
                CMP         (IOBPL),Y
                BEQ         SAMESLOT                ;IF HE DIDN'T, GOOD FOR HIM!
*
* NOW ARE USING A DIFFERENT SLOT.
* NOW WAIT FOR THIS DRIVE TO TURN OFF
* TO SENSE MOTOR NOT SPINNING, DATA FROM DISK MUST
* BE THE SAME FOR AT LEAST 96 MICROSECONDS
                TXA                                 ;SAVE NEW SLOT #
                PHA
                LDA         (IOBPL),Y               ;GET 'OLD SLOT NUMBER'
                TAX
                PLA
                PHA                                 ;PUT BACK ON STACK
                STA         (IOBPL),Y               ;SAVE 'NEW SLOT NUMBER'
                LDA         Q7L,X                   ;GO INTO READ MODE
STILLON         LDY         #$08                    ;TO BE SURE, DATA MUST REMAIN
                LDA         Q6L,X                   ;STABLE FOR 96 MICROSECONDS
NOTSURE         CMP         Q6L,X                   ;DATA STILL CHANGING?
                BNE         STILLON                 ;IF SO, STILL SPINNING
                DEY
                BNE         NOTSURE                 ;STABLE LONG ENOUGH? IF NOT, LOOP
*
* PREVIOUS SLOT'S DRIVE NOW OFF...
*
                PLA                                 ;RESTORE NEW SLOT #
                TAX
*
* NOW CHECK IF THE MOTOR IS ON, THEN START IT
*
SAMESLOT        LDA         Q7L,X                   ;MAKE SURE IN READ MODE
                LDA         Q6L,X
                LDY         #8                      ;WE MAY HAFTA CHECK SEVERAL TIMES TO
BE SURE
CHKIFON         EQU         *
                LDA         Q6L,X                   ;GET THE DATA
                PHA                                 ;DELAY FOR DISK DATA TO CHANGE
                PLA
                PHA
                PLA
                STX         SLOT
                CMP         Q6L,X                   ;CHECK RUNNING HERE
                BNE         ITISON                  ;=>IT'S ON...
                DEY                                 ;MAYBE WE DIDN'T CATCH IT
                BNE         CHKIFON                 ; SO WE'LL TRY AGAIN
*
ITISON          EQU         *
```

```
                PHP                             ;SAVE TEST RESULTS
                LDA         MOTORON,X           ;TURN ON MOTOR REGARDLESS
                LDY         #6                  ;MOVE OUT ALL POINTERS INTO ZPAGE
PTRMOV          LDA         (IOBPL),Y
                STA         PTRSDEST-6,Y
                INY
                CPY         #$0A                ;MOVED ALL POINTERS?
                BNE         PTRMOV
                LDY         #3                  ;SET UP THE
                LDA         (DEVCTBL),Y         ; MOTOR-ON TIME
                STA         MONTIME+1
                LDY         #2                  ;NOW GET PARAMS
                LDA         (IOBPL),Y           ;DETERMINE DRIVE ONE OR TWO
                LDY         #$10                ;SAME DRIVE USED BEFORE?
                CMP         (IOBPL),Y
                BEQ         OK                  ;IF SO, DON'T NECESSARILY WAIT FOR
MOTOR
                STA         (IOBPL),Y           ;NOW USING THIS DRIVE
                PLP                             ;TELL HIM MOTOR WAS OFF
                LDY         #$00                ;SET ZERO FLAG
                PHP
OK              ROR         A                   ;BY GOING INTO THE CARRY
                BCC         SD1                 ;SELECT DRIVE 2 !
                LDA         DRV1EN,X            ;ASSUME DRIVE 1 TO HIT
                BCS         DRVSEL              ;IF WRONG, ENABLE DRIVE 2 INSTEAD
SD1             LDA         DRV2EN,X
DRVSEL          EQU         *
                ROR         DRIVNO              ;SAVE SELECTED DRIVE
*
* DRIVE SELECTED. IF MOTORING-UP,
*  WAIT BEFORE SEEKING...
*
                PLP                             ;WAS THE MOTOR
                PHP                             ; PREVIOUSLY OFF?
                BNE         NOWAIT              ;=>NO, FORGET WAITING.
                LDY         #7                  ;YES, DELAY 150 MS
SEEKW           JSR         MSWAIT
                DEY
                BNE         SEEKW
                LDX         SLOT                ;RESTORE SLOT NUMBER
*
NOWAIT          EQU         *
*
* SEEK TO DESIRED TRACK...
*
                LDY         #4                  ;SET TO IOBTRK
                LDA         (IOBPL),Y           ;GET DESIRED TRACK
                JSR         MYSEEK              ;SEEK!
*
* SEE IF MOTOR WAS ALREADY SPINNING.
*
                PLP                             ;WAS MOTOR ON?
                BNE         TRYTRK              ;IF SO, DON'T DELAY, GET IT TODAY!
*
*  WAIT FOR MOTOR SPEED TO COME UP.
*
                LST         OFF
                DO          DIAGMODE
                LDY         #TC2                ;SAY 'MOTOR COMING ON'
                STY         TL2
                LDY         #0                  ;CLEAR SECTOR AND
                STY         SCOUNT              ; LATENCY COUNTERS
                STY         LCOUNT
```

```
                LDY         #SP                     ;SHUT OFF THE
                STY         TL12                    ; LATENCY INDICATOR
                FIN
                LST         ON
                LDY         MONTIME+1               ;IF MOTORTIME IS POSITIVE,
                BPL         MOTORUP                 ; THEN SEEK WASTED ENUFF TIME FOR US
MOTOF           LDY         #$12                    ;DELAY 100 USEC PER COUNT
CONWAIT         DEY
                BNE         CONWAIT
                INC         MONTIME
                BNE         MOTOF
                INC         MONTIME+1
                BNE         MOTOF                   ;COUNT UP TO $0000
MOTORUP         EQU         *
                LST         OFF
                DO          DIAGMODE
                LDY         #SP                     ;SAY 'MOTOR RUNNING'
                STY         TL2
                FIN
                LST         ON

; ###########################################################################################
; #   END OF FILE:  RWTSONE
; #   LINES      :  323
; #   CHARACTERS :  12866
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT RWTSTWO.pretty
================================================================================

; ###############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  RWTSTWO
; ###############################################################################

*
* DISK IS NOW UP TO SPEED: READ IT!
* NOW CHECK: IF IT IS NOT THE FORMAT DISK COMMAND,
*  LOCATE THE CORRECT SECTOR FOR THIS OPERATION.
*
TRYTRK          EQU         *
                LDY         #$0C
                LDA         (IOBPL),Y               ;GET COMMAND CODE #
                BEQ         GALLDONE                ;IF NULL COMMAND, GO HOME TO BED.
                CMP         #$04                    ;FORMAT THE DISK?
                BEQ         FORMDSK                 ;ALLRIGHT,ALLRIGHT, I WILL...
                ROR         A                       ;SET CARRY=1 FOR READ, 0 FOR WRITE
                PHP                                 ;AND SAVE THAT
                BCS         TRYTRK2                 ;MUST PRENIBBLIZE FOR WRITE.
                LST         OFF
                DO          DIAGMODE
                LDY         #TC7                    ;SAY 'PRENIBBLIZING'
                STY         TL7
                FIN
                LST         ON
                JSR         PRENIB16
                LST         OFF
                DO          DIAGMODE
                LDY         #SP                     ;PRENIB FINISHED
                STY         TL7
                FIN
                LST         ON
TRYTRK2         LDY         #$30                    ;ONLY 48 RETRIES OF ANY KIND.
                STY         RETRYCNT
TRYADR          LDX         SLOT                    ;GET SLOT NUM INTO X-REG
                LST         OFF
                DO          DIAGMODE
                LDA         #TC4                    ;SAY 'READING ADDRESS'
                STA         TL4
                LDA         #SP                     ;SAY 'NO ADDRESS ERROR' (YET)
                STA         TL6
                FIN
                LST         ON
                JSR         RDADR16                 ;READ NEXT ADDRESS FIELD
                LST         OFF
                DO          DIAGMODE
                LDA         #SP                     ;ADDRESS-READ DONE
                STA         TL4
                FIN
                LST         ON
                BCC         RDRIGHT                 ;IF READ IT RIGHT, HURRAH!
                LST         OFF
                DO          DIAGMODE
                LDA         #TC6                    ;SAY 'ADDRESS ERROR'
                STA         TL6
                FIN
                LST         ON
TRYADR2         DEC         RETRYCNT                ;ANOTHER MISTAEK!!
```

```
                BPL         TRYADR                  ; WELL, LET IT GO THIS TIME.,
*
* RRRRRECALIBRATE !!!!
*
RECAL           EQU         *
                LDA         CURTRK
                PHA                                 ;SAVE TRACK WE REALLY WANT
                LDA         #$60                    ;RECALIBRATE ALL OVER AGAIN!
                JSR         SETTRK                  ;PRETEND TO BE ON TRACK 96
                DEC         RECALCNT                ;ONCE TOO MANY??
                BEQ         DRVERR                  ;TRIED TO RECALIBRATE TOO MANY
TIMES, ERROR!
                LDA         #MAXSEEKS               ;RESET THE
                STA         SEEKCNT                 ; SEEK COUNTER
                LDA         #$00
                JSR         MYSEEK                  ;MOVE TO TRACK 00
                PLA
RESEEK          JSR         MYSEEK                  ;GO TO CORRECT TRACK THIS TIME!
                JMP         TRYTRK2                 ;LOOP BACK, TRY AGAIN ON THIS TRACK
*
* 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
* NO, DRIVE WAS ON A DIFFERENT TRACK. TRY
* RESEEKING/RECALIBRATING FROM THIS TRACK
                LDA         CURTRK                  ;PRESERVE DESTINATION TRACK
                PHA
                TYA
                JSR         SETTRK
                PLA
                DEC         SEEKCNT                 ;SHOULD WE RESEEK?
                BNE         RESEEK                  ;=>YES, RESEEK
                BEQ         RECAL                   ;=>NO, RECALIBRATE!
***
DRVERR          PLA                                 ;REMOVE CURTRK.
                LDA         #IBDERR                 ;BAD DRIVE ERROR
JMPTO1          PLP
                JMP         HNDLERR
GALLDONE        BEQ         ALLDONE
FORMDSK         JMP         DSKFORM                 ;=>GO TO IT!
*
* DRIVE IS ON RIGHT TRACK, CHECK VOLUME MISMATCH
*
RTTRK           LDY         #3                      ;IS THE RIGHT DISK IN?
                LDA         (IOBPL),Y               ;GET DESIRED VOLUM
                PHA                                 ;PRESERVE DESIRED VOLUME#
                LDA         VOLUME                  ;GET ACTUAL VOLUME HERE
                LDY         #$0E                    ;TELL OPSYS WHAT VOLUME WAS THERE
                STA         (IOBPL),Y
                PLA                                 ;GET DESIRED VOLUME BACK
                BEQ         CORRECTVOL              ;DESIRED VOLUME 00 MATCHES ALL.
                CMP         VOLUME
                BEQ         CORRECTVOL              ;YUP, IT WAS RIGHT
                LDA         #IBVMME                 ;HE SWITCHED DISCS!
                BNE         JMPTO1                  ;ALWAYS TAKEN
CORRECTVOL      EQU         *
                LDY         #5                      ; TO ALLOW FOR INTERLEAVE
                LDA         (IOBPL),Y               ;GET REQUESTED (LOGICAL) SECTOR
                TAY                                 ;MOVE TO INDEX REG
                LDA         INTRLEAV,Y              ;COMPUTE PHYSICAL SECTOR
```

```
                CMP         SECT                    ;DID WE GET THE SECTOR?
                LST         OFF
                DO          DIAGMODE
                BEQ         GOTSECT                 ;=>WE FOUND IT!
                LDA         #TC5                    ;SAY 'LATENCY'
                STA         TL5
                LDA         SCOUNT                  ;ARE WE WAITING FOR FIRST SECTOR?
                BEQ         NOLAT                   ;=>YES. LATENCY UNPREDICTABLE ANYWAY
                INC         LCOUNT                  ;NO, COUNT SECTORS MISSED
NOLAT           EQU         *
                JMP         TRYADR2                 ;NOW..GET CORRECT SECTOR..
                ELSE
                LST         ON
                BNE         TRYADR2                 ;NO, KEEP TRYING.
                FIN
*
* HOORAY! WE GOT THE RIGHT SECTOR!
*
GOTSECT         EQU         *
                LST         OFF
                DO          DIAGMODE
                LDA         #SP                     ;SAY 'NO LATENCY'
                STA         TL5
                INC         SCOUNT                  ;BUMP 'SECTORS-ACCESSED' COUNT
                FIN
                LST         ON
                PLP
                BCC         WRIT                    ;CARRY WAS SET FOR READ OPERATION,
                LST         OFF
                DO          DIAGMODE
                LDA         #TC9                    ;SAY 'READING'
                STA         TL9
                LDA         #SP                     ;SAY 'NO READ ERROR' (YET)
                STA         TL10
                FIN
                LST         ON
                JSR         READ16                  ;CLEARED FOR WRITE
                LST         OFF
                DO          DIAGMODE
                LDA         #SP                     ;READ FINISHED
                STA         TL9
                FIN
                LST         ON
                PHP                                 ;SAVE STATUS OF READ OPERATION
                LST         OFF
                DO          DIAGMODE
                BCC         GOODREAD                ;NO ERROR
                LDA         #TC10                   ;SAY 'READ ERROR'
                STA         TL10
                JMP         TRYADR2                 ;RETRY ON ERROR
*
GOODREAD        EQU         *
                ELSE
                LST         ON
                BCS         TRYADR2                 ;CARRY SET UPON RETURN IF BAD READ
                FIN
                PLP                                 ;CAREFUL OF STACK
                LDX         #0                      ;SET TO POSTNIBLIZE
                STX         T0                      ; ALL 256 BYTES OF THE SECTOR
                LST         OFF
                DO          DIAGMODE
                LDA         #TC11                   ;SAY 'POSTINIBBLIZING'
                STA         TL11
```

```
                FIN
                LST         ON
                JSR         POSTNB16                ;DECODE INTO REAL WORLD DATA
                LST         OFF
                DO          DIAGMODE
                LDA         #SP                     ;POSTNIB COMPLETED
                STA         TL11
                FIN
                LST         ON
                LDX         SLOT                    ;RESTORE SLOTNUM INTO X
ALLDONE         CLC
                DFB         $24                     ;SKIP OVER NEXT BYTE WITH BIT OPCODE
HNDLERR         SEC                                 ;INDICATE AN ERROR
                LDY         #$0D                    ;GIVE HIM ERROR#
                STA         (IOBPL),Y
                LDA         MOTOROFF,X              ;TURN IT OFF...
                LST         OFF
                DO          DIAGMODE
* AVERAGE LATENCY = LCOUNT/SCOUNT
                LDA         LCOUNT                  ;GET TOTAL LATENCY
                LDY         #0                      ;CLEAR QUOTIENT
DIVIDE          EQU         *
                CMP         SCOUNT                  ;DONE?
                BCC         PRTLAT                  ;=>YES.PRINT IT
                SBC         SCOUNT                  ;REMOVE SCOUNT
                INY                                 ;INCREMENT QUOTIENT
                BNE         DIVIDE
*
PRTLAT          TYA
                AND         #$0F                    ;MAX LATENCY=15
                ORA         #$B0                    ;MAKE ASCII
                CMP         #$BA                    ;IS IT A-F?
                BCC         PRTL2                   ;=>NO
                ADC         #6                      ;ADD 7 (INCLUDES CARRY)
PRTL2           STA         TL12                    ;STUFF LATENCY COUNT
                LDA         #SP                     ;SAY 'RWTS NOT ACTIVE'
                STA         TL1
                FIN
                LST         ON
                RTS
WRIT            EQU         *
                LST         OFF
                DO          DIAGMODE
                LDA         #TC8                    ;SAY 'WRITING'
                STA         TL8
                FIN
                LST         ON
                JSR         WRITE16                 ;WRITE NYBBLES NOW
                LST         OFF
                DO          DIAGMODE
                LDA         #SP                     ;WRITE FINISHED
                STA         TL8
                FIN
                LST         ON
                BCC         ALLDONE                 ;IF NO ERRORS.
                LDA         #IBWPER                 ;DISK IS WRITE PROTECTED!!
                BCS         HNDLERR                 ;ALWAYS TAKEN
*
* THIS IS THE 'SEEK' ROUTINE
*  SEEKS TRACK 'N' IN SLOT #X/$10
* IF DRIVNO IS NEGATIVE, ON DRIVE 1
* IF DRIVNO IS POSITIVE, ON DRIVE 2
*
```

```
MYSEEK          PHA                                     ;AND PRESERVE A-REGISTER
                LST             OFF
                DO              DIAGMODE
                LDA             #TC3                    ;SAY 'SEEKING'
                STA             TL3
                FIN
                LST             ON
                LDY             #$01                    ;IS THIS A TWO-PHASE DISC?
                LDA             (DEVCTBL),Y
                ROR             A                       ;GET # OF PHASES INTO CARRY
                PLA
                BCC             MYSEEK2                 ;IF ONE PHASE PER TRACK
                ASL             A
                JSR             MYSEEK2
                LSR             CURTRK                  ;DIVIDE BACK DOWN
                LST             OFF
                DO              DIAGMODE
                LDA             #SP
                STA             TL3                     ;SEEK DONE
                FIN
                LST             ON
                RTS
MYSEEK2         STA             TRKN                    ;SAVE DESTINATION TRACK(*2)
                JSR             XTOY                    ;SET Y=SLOT#
                LDA             DRV1TRK,Y
                BIT             DRIVNO
                BMI             WASD0                   ;IS MINUS, ON DRIVE ZERO
                LDA             DRV2TRK,Y
WASD0           STA             CURTRK                  ;THIS IS WHERE I AM
                LDA             TRKN                    ;AND WHERE I'M GOING TO
                BIT             DRIVNO                  ;NOW UPDATE SLOT DEPENDENT
                BMI             ISDRV1                  ;LOCATIONS WITH TRACK
                STA             DRV2TRK,Y               ;INFORMATION
                BPL             GOSEEK                  ;ALWAYS TAKEN
ISDRV1          STA             DRV1TRK,Y
GOSEEK          JMP             SEEK                    ;GO THERE!
XTOY            TXA
                LSR             A
                LSR             A
                LSR             A
                LSR             A
                TAY
                RTS
*
* THIS SUBROUTINE SETS THE SLOT DEPENDENT TRACK
* LOCATION.
*
SETTRK          PHA                                     ;PRESERVE DESTINATION TRACK
                LDY             #$02
                LDA             (IOBPL),Y
                ROR             A                       ;GET DRIVE # INTO CARRY
                ROR             DRIVNO                  ;INTO (DRIVNO)
                JSR             XTOY                    ;SET UP Y-REG
                PLA
                ASL             A                       ;ASSUME TRACK IS HELD *2
SETTRK2         BIT             DRIVNO
                BMI             ONDRV1                  ;IF ON DRIVE 1(1), DRIVNO MINUS
                STA             DRV2TRK,Y
                BPL             SETRTS
ONDRV1          STA             DRV1TRK,Y
SETRTS          RTS

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

```
; #   END OF FILE:  RWTSTWO
; #   LINES      :  302
; #   CHARACTERS :  14425
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT TEMPY.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  TEMPY
; ##############################################################################

*********************************************
*
* GHOST APPEND BUG PATCH BY
* BILL GRIMM
*
*********************************************
MOVEOF          EQU         *
                LDX         CMDNO                   ; GET CMD NUMBER
                CPX         #$1C                    ; APPEND COMMAND?
                BEQ         GOON                    ; YES, RETURN TO CALLING ROUTINE
                LDX         #$00                    ; NO, THEN CLEAR X
                STX         EOFFLAG                 ; CLEAR EOF FLAG
GOON            RTS
*********************************************
*
* TURN Apple //e 80 COLUMN CARD
* OFF & INIT APPLE
*
*********************************************
OFF80           EQU         *
                LDA         #$FF
                STA         $4FB                    ; CLEARS FUNNY 80 COL STUFF
                STA         $C00C                   ; TURNS 80 COL OFF
                STA         $C00E                   ; TURN OFF ALT CHAR SET
                JSR         $FB2F                   ; MONITOR INIT ROUTINE
                JMP         HOME                    ; CLEAR 80 COL GARBAGE
*********************************************
*
* FIXIT2 was developed to fix the wrap around
* problem APPEND has when trying to APPEND to
* a sequential file which is >255 sectors in
* length.
*          Fix by BANKS/BACHMAN
*           September 28, 1982
*
*********************************************
                SKP         1
FIXIT2          EQU         *
                LDA         CCBRLN                  ;Store current rec len low byte
                STA         DCBCSB                  ;in current sector byte
                STA         DCBCRR                  ;and in current relative rec
                TSX                                 ;Save status in ENTSTK
                STA         ENTSTK                  ;for proper exit from GOODIO
                JMP         GOODIO                  ;

; ##############################################################################
; #   END OF FILE:  TEMPY
; #   LINES      :  44
; #   CHARACTERS :  1976
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT TRASH.pretty
================================================================================

; ##############################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  TRASH
; ##############################################################################

               PAGE
;
;FNDFIL - FIND FILE NAME IN VOLUUME DIR
;
FNDFIL         EQU         *
               JSR         RDVTOC                  ; GO GET VTOC
               LDA         CCBFN1                  ; MOVE FN PTR
               STA         ZPGFCB                  ; TO ZERO PAGE
               LDA         CCBFN1+1
               STA         ZPGFCB+1
               LDA         #1
FF1            STA         TEMP2
               LDA         #0
               STA         DCBVDR
               CLC
FF2            EQU         *
               INC         DCBVDR
               JSR         RDVDIR                  ; GO GET VDIR SECTOR
               BCS         FF4A
               LDX         #0                      ; SET FOR 1ST FILE
;
FF3            STX         TEMP1                   ; SAVE INDEX
               LDA         VDFILE,X                ; GET FILE TRK
               BEQ         FF6                     ; BR IF LAST ENTRY
               BMI         FF7                     ; BR DELETED ENTRY
               LDY         #0                      ; X=X+3
               INX
               INX
FF4            INX
               LDA         (ZPGFCB),Y              ; GET FN CHAR
               CMP         VDFILE,X                ; COMPARE TO ENTRY CHAR
               BNE         FF5                     ; BR IF NOT SAME
               INY
               CPY         #30                     ; ALL 30 CHARS
               BNE         FF4                     ; BR IF NOT
               LDX         TEMP1                   ; GET INDEX
               CLC                                 ; FILE FOUND
               RTS                                 ; RETURN
;
FF5            EQU         *
               JSR         VDINC
               BCC         FF3
               BCS         FF2
;
FF6            LDY         TEMP2                   ; LOOKING FOR DELETED
               BNE         FF1                     ; BR IF NOT (DO)
;
FF7            LDY         TEMP2                   ; LOOKING FOR EMPTY
               BNE         FF5                     ; BR IF NOT
;
MVFN           EQU         *
               LDY         #0                      ; HAVE NEW ENTTRY
               INX
```

```
                INX
FF8             INX
                LDA         (ZPGFCB),Y              ; MOVE FILE NAME
                STA         VDFILE,X
                INY
                CPY         #30
                BNE         FF8
;
                LDX         TEMP1                   ; GET INDEX
                SEC                                 ; SET NOT OLD
                RTS                                 ; DONE
VDINC           EQU         *
                CLC
                LDA         TEMP1
                ADC         #35
                TAX
                CPX         #VDFLEN
                RTS
FF4A            EQU         *
                LDA         #0
                LDY         TEMP2
                BNE         FF1
                JMP         ERROR9
                PAGE
;
;GETSEC - GET A SECTOR
;
GETSEC          EQU         *
                LDA         DCBATK                  ; GET ALLOCATED TRK
                BEQ         GSS1                    ; BR IF NONE
;
GS0             EQU         *
                DEC         DCBALS                  ; DECREMENT SECTOR NO
                BMI         CS2                     ; BR IF NO SECTORS REM
;
                CLC
                LDX         #4                      ; 4 BYTE SHIFT
GS1             ROL         DCBABM-1,X              ; SHIFT BYTE LEFT
                DEX
                BNE         GS1
                BCC         GS0                     ; BR IF NO SECTOR
;
                INC         DCBNSA
                BNE         GS1A
                INC         DCBNSA+1
GS1A            EQU         *
                LDA         DCBALS                  ; GET ALLOCATED SECTOR
                RTS                                 ; RETURN
;
CS2             LDA         #0                      ; CLEAR ALLOCATED
                STA         DCBATK                  ; TRK
;
GSS1            LDA         #0                      ; SET SEARCH STATE=0
                STA         TEMP3
                JSR         RDVTOC                  ; GET VTOC
;
GS2             EQU         *
                CLC
                LDA         VALCA1                  ; GET LAST ALLOCATTED TRK
                ADC         VALCA2                  ; AD (+1) OR (-1)
                BEQ         GS3                     ; BR IF DECK TO ZERO
                CMP         VNOTRK
                BCC         GS5                     ; BR IF NOT AT OUTER LIMIT
```

```
                LDA         #$FF                    ; SET (-1)
                BNE         GS4
GS3             LDA         TEMP3                   ; GET SEARCH STATE
                BNE         ERR9                    ; BR IF NOT ZERO
                LDA         #1                      ; SET (+1)
                STA         TEMP3                   ; SET SEARCH STATE = 1
GS4             STA         VALCA2                  ; SET NEW (+1) OR -1)
                CLC
                ADC         #17                     ; ADD VTOC TRK NO
GS5             STA         VALCA1                  ; SET NEW LAST ALLOCATED
                STA         DCBATK                  ; PUT IN DCB
;
                TAY                                 ; ALLOCATED TRACK
                ASL         A                       ; TIME 4
                ASL         A
                TAY
                LDX         #4
                CLC
GS6             LDA         VSECAL+3,Y              ; MOVE BIT MAP BYTE
                STA         DCBABM-1,X
                BEQ         GS7                     ; BR IF NO BITS ON
                SEC                                 ; SET HAVE A SECTOR
                LDA         #0                      ; CLEAR VTOC BYTE
                STA         VSECAL+3,Y
GS7             DEY
                DEX
                BNE         GS6                     ; BR IF MORE TO MOVE
                BCC         GS2
                JSR         WRVTOC                  ; GO WRITE VTOC
                LDA         VNOSEC                  ; GET NO SECTORS
                STA         DCBALS                  ; SET IN DCB SECTOR BYTE
                BNE         GS0                     ; GO ALLOCATED SECTOR
ERR9            JMP         ERROR9
                PAGE
;
;FRETRK - FREE TRACK OF SECTORS
;
FRETRK          EQU         *
                LDA         DCBATK                  ; GET ALLOCATED TRACK
                BNE         FT1                     ; BR IF NONE
                RTS                                 ; DONE
FT1             PHA
                JSR         RDVTOC                  ; GET VTOC
                LDY         DCBALS                  ; GET SECTOS
                PLA                                 ; GET TRACK
                CLC                                 ; SET FREE
                JSR         FRESEC                  ; GO FREE
                LDA         #0                      ; CLEAR ALLOCATED TRK
                STA         DCBATK
                JMP         WRVTOC                  ; WRITE VTOC
;
;FRESEC - FREE A SECTOR
;A=TRK, Y=SECTOR, C=ON/OFF
;
FRESEC          EQU         *
FS1             LDX         #252                    ; 4 BYTE SHIFT
FS2             ROR         DCBABM-252,X            ; SHIFT IN CARRY
                INX                                 ; NEXT BYTE
                BNE         FS2                     ; BR IF NOT DONE
                INY                                 ; INC SECTOR NO
                CPY         VNOSEC                  ; NORMAL
                BNE         FS1                     ; BR IF NOT
;
```

```
                ASL         A                       ; TRACK*4
                ASL         A
                TAY
                BEQ         FS4
                LDX         #4
FS3             LDA         DCBABM-1,X              ; GET BIT MAP BYTE
                ORA         VSECAL+3,Y              ; OR WITH VTOC BM
                STA         VSECAL+3,Y
                DEY
                DEX
                BNE         FS3
FS4             RTS                                 ; DONE
                PAGE
;
;LOCSEC - LOCATE SECTOR FOR RECORD I/O
;
;RELSEC = (REL REC * RECLEN + RELBYTE)/256
;SECBYT = REMAINDER
;
LOCSEC          EQU         *
                LDA         CCBRRN                  ; RELATIVE RECORD NUMBER
                STA         DCBCSB                  ; TO CSB FOR MULT
                STA         DCBCRR                  ; AND CRR FOR SAVE
                LDA         CCBRRN+1
                STA         DCBCRS
                STA         DCBCRR+1
                LDA         #0
                STA         DCBCRS+1                ; HIGH CRS=0
                LDY         #16                     ; 16 BIT MULT
;
LS1             TAX                                 ; SAVE MS BYTE
                LDA         DCBCSB
                LSR         A                       ; IF NO CARRY THEN NO PART PROD
                BCS         LS1A
                TXA
                BCC         LS2
LS1A            CLC
                LDA         DCBCRS+1                ; FPORM PARTIAL PROD
                ADC         DCBRCL
                STA         DCBCRS+1
                TXA
                ADC         DCBRCL+1
;
LS2             ROR         A                       ; MULT BY 2
                ROR         DCBCRS+1
                ROR         DCBCRS
                ROR         DCBCSB
                DEY                                 ; DEC BIT COUNT
                BNE         LS1                     ; BR IF MORE BITS
;
                DO          DOS33B
                CLC                                 ; FOR FILE LENGTH > $7FFF BYTES
                FIN
                LDA         CCBBYT                  ; ADD REL BYTE RESULT
                STA         DCBCRB                  ; (SAVE REL BYTE)
                ADC         DCBCSB
                STA         DCBCSB
                LDA         CCBBYT+1
                STA         DCBCRB+1                ; (SAVE REL BYTE)
                ADC         DCBCRS
                STA         DCBCRS
                DO          DOS33B
                BCC         DONTINC
```

```
                INC           DCBCRS+1
DONTINC         RTS
                DS            2,$00
                ELSE
                LDA           #0
                ADC           DCBCRS+1
                STA           DCBCRS+1
                RTS
                FIN
                PAGE
ERROR1          LDA           #CREFUN
                BNE           ERRORA
ERROR2          LDA           #CRERR
                BNE           ERRORA
ERROR3          LDA           #CREMRE
                BNE           ERRORA
ERROR4          LDA           #CREPRO
                BNE           ERRORA
ERROR5          LDA           #CREEOF
                BNE           ERRORA
ERROR6          LDA           #CREFNF
                BNE           ERRORA
ERROR9          JMP           ERROR9X                   ;MUST CLOSE ALL FILES (WAS LDA
#CRENSA)
                NOP
ERRR10          LDA           #CREFLK
                BNE           ERRORA
GOODIO          LDA           CCBSTA
                CLC                                     ; CARRY=CLR
                BCC           RETURN                    ; GO RETURN
ERRORA          EQU           *
ERRORB          SEC                                     ; CARRY=SET
RETURN          EQU           *
                PHP
                STA           CCBSTA                    ; SET STA
                LDA           #0                        ;(FIX FOR APPLE SYS MONITOR $48 USED
BY RWTS)
                STA           $48                       ;(THIS ADDED 11/1/78)
                JSR           RTNFCB                    ; GO RTN FCB
                PLP                                     ; GET STATUS
                LDX           ENTSTK                    ; GET ENT STACK
                TXS                                     ; RESTORE STACK
                RTS                                     ; DONE
EC2             EQU           *

; ###############################################################################
; #   END OF FILE:  TRASH
; #   LINES      :  284
; #   CHARACTERS :  12221
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################
```

```
================================================================================
DOCUMENT WRITADR.pretty
================================================================================

; ###############################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  WRITADR
; ###############################################################################################

                SBTL         '16-SECTOR WRITE ADDRESS'
*******************************
*                             *
*  WRITE ADR FIELD SUBROUTINE  *
*      (16-SECTOR FORMAT)      *
*  WRITES SPECIFIED NUMBER OF  *
*  40-USEC (10-BIT) SELF-SYNC  *
*  NIBLS, ADR FIELDS 16-SECTOR *
*  START MARKS ($D5,$AA,$96),  *
*  BODY (VOLUME, TRACK, SECTOR,*
*  CHECKSUM), END FIELD MARKS, *
*  AND THE WRITE TURN-OFF NIBL.*
*                             *
*******************************
*                             *
*   ------- ON ENTRY -------   *
*                             *
*  THE LOCATIONS VOLUME, TRK,  *
*  AND NSECT MUST CONTAIN THE  *
*  DESIRED VOLUME, TRACK, AND  *
*  SECTOR VALUES DESIRED.      *
*                             *
*  THE PROPER DRIVE MUST BE    *
*  ENABLED AND UP TO SPEED IN  *
*  READ MODE (Q7L, Q6L).       *
*                             *
*  X-REG CONTAINS SLOTNUM      *
*                    TIMES 16. *
*                             *
*  Y-REG CONTAINS NUMBER OF    *
*    SELF-SYNC NIBLS DESIRED   *
*    MINUS 1.                  *
*      (0 FOR 256 NIBLS)       *
*                             *
*******************************
*                             *
*   ------- REQUIRES -------   *
*                             *
*         1 USEC CYCLE        *
*                             *
*******************************
*                             *
*   ------- CAUTION --------   *
*                             *
*  MOST OF THIS CODE IS TIME   *
*  CRITICAL.  OBSERVE ALL      *
*  'NO PAGE CROSS!' WARNINGS   *
*  ON BRANCHES.                *
*                             *
*******************************
SWADR1          EQU          *                          ;TELL RELOCATOR WHERE TO BEGIN
*******************************
WADR16          SEC          ANTICIPATE                 WR PROT ERR.
```

```
                LDA         Q6H,X                   INTO 'WR PROT SENSE' MODE.
                LDA         Q7L,X                   SENSE IT (NEG=PROTECTED)
                BMI         WADRTS                  ERR EXIT IF PROTECTED.
                LDA         #$FF                    SELF-SYNC NIBL.
                STA         Q7H,X                   WRITE FIRST NIBL.
                CMP         Q6L,X                   (4) BACK TO WRITE MODE.
                PHA         (3)                     FOR DELAY.
                PLA         (4)
WSYNC1          JSR         WADRTS1                 (12) FOR 40-USEC NIBLS.
                JSR         WADRTS1                 (12)
                STA         Q6H,X                   (5) WRITE NIBL.
                CMP         Q6L,X                   (4) (BACK TO WRITE MODE)
                NOP         (2)                     FOR DELAY.
                DEY         (2)                     NEXT OF 'N' NIBLS.
                BNE         WSYNC1                  (3) *** NO PAGE CROSS! ***
                LDA         #$D5                    (2) ADR MARK 1.
                JSR         WNIBLB2                 (15,9,6) WRITE IT.
                LDA         #$AA                    (2) ADR MARK 2.
                JSR         WNIBLB2                 (15,9,6) WRITE IT.
                LDA         #$96                    (2) 16-SECTOR ADR MARK 3.
                JSR         WNIBLB2                 (15,9,6) WRITE IT.
                LDA         NVOL                    (3)
                JSR         WBYTE                   (14,9,6) WRITE NVOL (ODD, THEN EVEN,
BITS.)
                LDA         TRK                     (3) WRITE TRACK NUMBER.
                JSR         WBYTE                   (14,9,6) ODD, THEN EVEN, BITS)
                LDA         NSECT                   (3) WRITE SECTOR NUMBER.
                JSR         WBYTE                   (14,9,6) (ODD, THEN EVEN, BITS)
                LDA         NVOL                    (3)
                EOR         TRK                     (3) FORM ADR FIELD CHECKSUM.
                EOR         NSECT                   (3)
                PHA         (3)                     SAVE FOR EVEN BITS.
                LSR         A                       (2) ALIGHN ODD BITS.
                ORA         AA                      (3) SET CLOCK BITS.
*    (PRECISE TIMING, 32 CYCLES PER NIBL)
                STA         Q6H,X                   (5) WRITE CHECKSUM ODD BITS.
                LDA         Q6L,X                   (4) BACK TO WRITE MODE.
                PLA         (4)                     RECOVER FOR EVEN BITS.
                ORA         #$AA                    (2) SET CLOCK BITS.
                JSR         WNIBLA                  (17,9,6) WRITE THEM.
                LDA         #$DE                    (2) END MARK 1.
                JSR         WNIBLB2                 (15,9,6) WRITE IT.
                LDA         #$AA                    (2) END MARK 2.
                JSR         WNIBLB2                 (15,9,6) WRITE IT.
                LDA         #$EB                    (2) END MARK 3.
                JSR         WNIBLB2                 (15,9,6) 'WRITE TURN-OFF'
                CLC         INDICATE                NO WR PROT ERR.
WADRTS          LDA         Q7L,X                   OUT OF WRITE MODE.
                LDA         Q6L,X                   TO READ MODE.
WADRTS1         RTS         RETURN
WBYTE           PHA         (3)                     PRESERVE FOR EVEN BITS.
                LSR         A                       (2) ALIGN ODD BITS.
                ORA         AA                      (3) SET CLOCK BITS.
                STA         Q6H,X                   (5) WRITE NIBL.
                CMP         Q6L,X                   (4)
                PLA         (4)                     RECOVER FOR EVEN BITS.
                NOP         (2)
                NOP         (2)                     FOR DELAY.
                NOP         (2)
                ORA         #$AA                    (2) SET CLOCK BITS.
WNIBLA          NOP         (2)                     (17,9,6) ENTRY.
WNIBLB2         NOP         (2)                     (15,9,6) ENTRY.
                PHA         (3)                     FOR
```

```
                PLA             (4)                     DELAY.
WRNIBL          STA             Q6H,X                   (5) WRITE NIBL.
                CMP             Q6L,X                   (4)
                RTS             (6)                     RETURN.
EWADR1          EQU             *                       ;TELL RELOCTR WHERE TO STOP
XP2             EQU             <*+255                  ;H.O. ADDRESS NEXT PAGE
XP2H            EQU             256*XP2                 ;NOW AS 16-BITS
                DS              XP2H-*,0                ;PAD OUT TO PAGE BOUNDARY..

; ###########################################################################################
; #   END OF FILE:  WRITADR
; #   LINES      :  123
; #   CHARACTERS :  6714
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT WRITRTN.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  WRITRTN
; ###########################################################################################

                SBTL         '16-SECTOR WRITE'
************************
*                      *
*      WRITE SUBR      *
*  (16-SECTOR FORMAT)  *
*                      *
************************
*                      *
*   WRITES DATA FROM   *
*    NBUF1 AND NBUF2   *
*   CONVERTING 6-BIT   *
*    TO 7-BIT NIBLS    *
*   VIA 'NIBL' TABLE.  *
*                      *
*  FIRST NBUF2,        *
*      HIGH TO LOW.    *
*  THEN NBUF1,         *
*      LOW TO HIGH.    *
*                      *
*  ---- ON ENTRY ----  *
*                      *
*   X-REG: SLOTNUM     *
*        TIMES $10.    *
*                      *
*   NBUF1 AND NBUF2    *
*    HOLD NIBLS FROM   *
*    PRENIBL SUBR.     *
*    (00ABCDEF)        *
*                      *
*  ---- ON EXIT -----  *
*                      *
*  CARRY SET IF ERROR. *
*   (W PROT VIOLATION) *
*                      *
*  IF NO ERROR:        *
*                      *
*    A-REG UNCERTAIN.  *
*    X-REG UNCHANGED.  *
*    Y-REG HOLDS $00.  *
*    CARRY CLEAR.      *
*                      *
*    SLOTABS, SLOTZ,   *
*     AND WTEMP USED.  *
*                      *
*  ---- ASSUMES ----   *
*                      *
*  1 USEC CYCLE TIME   *
*                      *
************************
WRITE16         SEC          ANTICIPATE               WPROT ERR.
                STX          SLOTZ                    FOR ZERO PAGE ACCESS.
                STX          SLOTABS                  FOR NON-ZERO PAGE.
                LDA          Q6H,X
```

```
                LDA         Q7L,X                   SENSE WPROT FLAG.
                BMI         WEXIT                   IF HIGH, THEN ERR.
                LDA         NBUF2
                STA         WTEMP                   FOR ZERO-PAGE ACCESS.
                LDA         #$FF                    SYNC DATA.
                STA         Q7H,X                   (5)  WRITE 1ST NIBL.
                ORA         Q6L,X                   (4)
                PHA         (3)
                PLA         (4)                     CRITICAL TIMING!
                NOP         (2)
                LDY         #4                      (2)  FOR 5 NIBLS.
WSYNC           PHA         (3)                     EXACT TIMING.
                PLA         (4)                     EXACT TIMING.
                JSR         WNIBL7                  (13,9,6)  WRITE SYNC.
                DEY         (2)
                BNE         WSYNC                   (2*)  MUST NOT CROSS PAGE!
                LDA         #$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)                     CLEAR CHKSUM.
                LDY         #$56                    (2)  NBUF2 INDEX.
                BNE         WDATA1                  (3)  ALWAYS.  NO PAGE CROSS!!
WDATA0          LDA         NBUF2,Y                 (4)  PRIOR 6-BIT NIBL.
WDATA1          EOR         NBUF2-1,Y               (5)  XOR WITH CURRENT.
*   (NBUF2 MUST BE ON PAGE BOUNDARY FOR TIMING!!)
                TAX         (2)                     INDEX TO 7-BIT NIBL.
                LDA         NIBL,X                  (4)  MUST NOT CROSS PAGE!
                LDX         SLOTZ                   (3)  CRITICAL TIMING!
                STA         Q6H,X                   (5)  WRITE NIBL.
                LDA         Q6L,X                   (4)
                DEY         (2)                     NEXT NIBL.
                BNE         WDATA0                  (2*)  MUST NOT CROSS PAGE!
                LDA         WTEMP                   (3)  PRIOR NIBL FROM BUF6.
                NOP         (2)                     CRITICAL TIMING.
WDATA2          EOR         NBUF1,Y                 (4)  XOR NBUF1 NIBL.
                TAX         (2)                     INDEX TO 7-BIT NIBL.
                LDA         NIBL,X                  (4)
                LDX         SLOTABS                 (4)  TIMING CRITICAL.
                STA         Q6H,X                   (5)  WRITE NIBL.
                LDA         Q6L,X                   (4)
                LDA         NBUF1,Y                 (4)  PRIOR 6-BIT NIBL.
                INY         (2)                     NEXT NBUF1 NIBL.
                BNE         WDATA2                  (2*)  MUST NOT CROSS PAGE!
                TAX         (2)                     LAST NIBL AS CHKSUM.
                LDA         NIBL,X                  (4)  INDEX TO 7-BIT NIBL.
                LDX         SLOTZ                   (3)
                JSR         WNIBL                   (6,9,6)  WRITE CHKSUM.
                LDA         #$DE                    (2)  DM4, BIT SLIP MARK.
                JSR         WNIBL9                  (15,9,6)    WRITE IT.
                LDA         #$AA                    (2)  DM5, BIT SLIP MARK.
                JSR         WNIBL9                  (15,9,6)    WRITE IT.
                LDA         #$EB                    (2)  DM6, BIT SLIP MARK.
                JSR         WNIBL9                  (15,9,6)    WRITE IT.
                LDA         #$FF                    (2) TURN-OFF BYTE.
                JSR         WNIBL9                  (15,9,9)  WRITE IT.
                LDA         Q7L,X                   OUT OF WRITE MODE.
WEXIT           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

; ###########################################################################################
; #   END OF FILE:  WRITRTN
; #   LINES      :  128
; #   CHARACTERS :  6272
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###########################################################################################
```

```
================================================================================
DOCUMENT XLODSAV.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  XLODSAV
; ###########################################################################################

               PAGE
;
; EBSV - EXECUTE BINARY SAVE
;
EBSV           EQU          *
               LDA          #ADR+L                    ; IF A&L
               AND          INOPTS                    ; NOT GIVEN
               CMP          #ADR+L
               BEQ          EBSV1
               JMP          CNF                       ; THEN ERROR
EBSV1          EQU          *
               LDA          #4                        ; SET BINARY FILE
               JSR          SV1                       ; GO OPEN & TEST
               LDA          CA+1                      ; OUTPUT ADR OF BLOCK
               LDY          CA
               JSR          SV2
               LDA          CL+1                      ; GO OPEN AND TEST
               LDY          CL
               JSR          SV2                       ; OUTPUT LENGTH
               LDA          CA+1                      ; GET ADR GIVEN
               LDY          CA
               JMP          SV3                       ; OUTPUT BLOCK
;
; EBLD - EXECUTE BINARY LOAD
;
EBLD           EQU          *
               JSR          EOPN1                     ;(CHANGED 11/1/78 FOR TYPE MISMATCH)
EBLD2          EQU          *
               LDA          #$7F
               AND          CCBFUC
               CMP          #4
               BEQ          EBLD3
               JMP          ETYP                      ;TYPE MISMATCH ERROR MESSAGE
EBLD3          EQU          *
               LDA          #4                        ; SET BINARY FILE
               JSR          SV1                       ; GO OPEN & TEST
               JSR          LD2                       ; GO GET ADR
               TAX
               LDA          INOPTS
               AND          #ADR                      ; IF ADR NOT GIVEN
               BNE          EBLD1
               STX          CA                        ; THEN USE ADR FROM FILE
               STY          CA+1
EBLD1          EQU          *
               JSR          LD2                       ; GET LENGTH
               LDX          CA                        ; GET GIVEN ADR
               LDY          CA+1
               JMP          LD3                       ; GO GET BLOCK
;
; EBRUN - EXECUTE BINARY RUN
;
EBRUN          JSR          EBLD                      ;DO A BINARY LOAD
               JSR          MVCSW                     ; GO RESTORE CHAR I/O SW
```

```
                JMP         (CA)                    ; GO EXEC THE STUFF
                PAGE
;
; ESAVE - EXECUTE SAVE REQUEST
;
ESAVE           EQU         *
                LDA         ASIBSW                  ; IF IB THEN
                BEQ         EIBSV                   ; GO TO IB SAVE
                LDA         ASRNX                   ;CANNOT DO AS SAVE WHEN RUN-ONLY
PROG.
                BPL         EASAV                   ;BRANCH IF OK TO SAVE, OTHERWISE
                JMP         MFERR                   ;PRINT "PROGRAM TOO LARGE", THAT
OUGHT TO GET 'EM.
EASAV           LDA         #2                      ; GET APPLESOFT PGM
                JSR         SV1                     ; GO OPEN AND TEST
;
                SEC                                 ; BLOCK LENGTH
                LDA         ASEOP                   ; =EOP-SOP
                SBC         ASSOP
                TAY
                LDA         ASEOP+1
                SBC         ASSOP+1
                JSR         SV2                     ; GO OUTPUT LENGTH
;
                LDA         ASSOP+1                 ; BLOCK ADR
                LDY         ASSOP                   ; =SOP
                JMP         SV3                     ; GO OUTPUT BLOCK
;
EIBSV           EQU         *
                LDA         #1                      ; SET IB PGM
                JSR         SV1                     ; GO OPEN AND TEST
;
                SEC                                 ; BLOCK LENGTH
                LDA         IBHMEM                  ; =HIMEM-SOP
                SBC         IBSOP
                TAY
                LDA         IBHMEM+1
                SBC         IBSOP+1
                JSR         SV2                     ; GO OUTPUT LENGTH
;
                LDA         IBSOP+1                 ; BLOCK ADR
                LDY         IBSOP                   ; =SOP
                JMP         SV3                     ; GO OUTPUT BLOCK
;
SV1             EQU         *
SV1A            EQU         *
                STA         CCBFUC                  ; SET PGM TYPE
                PHA                                 ; SAVE PGM TYPE
                JSR         EOPN1                   ; GO OPEN FILE  (CHGED 11/1/78)
                PLA                                 ; GET SAVE TYPE
                JMP         TSTFUC                  ; GO CHECK
;
SV2             EQU         *
                STY         CCBBLN                  ; SET BLOCK LENGTH
                STY         CCBDAT                  ; AND DATA BYTE
                STA         CCBBLN+1
                LDA         #CRQWR                  ; INDICATE WRITE
                STA         CCBREQ
                LDA         #CRMNBT                 ; NEXT BYTE
                STA         CCBRQM
                JSR         DOSGO                   ; GO WRITE
                LDA         CCBBLN+1                ; OTHER BYTE TOO
                STA         CCBDAT
```

```
                JMP         DOSGO
;
SV3             STY         CCBBBA                  ; SET BLOCK ADR
                STA         CCBBBA+1
                LDA         #CRMNBL                 ; INDICATE BLOCK I/O
                JMP         VPATCH                  ;VERIFY AFTER SAVE
GODOS           JSR         DOSGO                   ; GO DO IT
                JMP         ECLOSE                  ; CLOSE FILE
                PAGE
NBPER           JMP         ERNU1
;
; ELOAD - EXECUTE LOAD REQUEST
;
ELOAD           EQU         *
                JSR         CLALL                   ; GO CLOSE ALL
ELOAD0          JSR         EOPN1                   ; OPEN FILE (CHGED 11/1/78)
;
ELD1            EQU         *
                LDA         #$23                    ; STRIP UNRELATED STUFF
                AND         CCBFUC                  ; OUT OF FUC
                BEQ         NBPER                   ; BR IF ERROR
; ISOLOLATE IB & AS
ELD2            EQU         *
                STA         CCBFUC                  ; SAVE IB/AS ONLY
                LDA         ASIBSW                  ; IF IB THEN
                BEQ         EIBL                    ; GO TO IB LOAD
                LDA         #2
                JSR         LD1                     ; GO OPEN AND TEST
;
                JSR         LD2                     ; GO GET BLOCK LENGTH
;
                CLC
                ADC         ASSOP                   ; ADD BLOCK LENGTH TO SOP
                TAX
                TYA
                ADC         ASSOP+1
;
                CMP         ASHM1+1                 ; IF BL+SOP >= HMEM
                BCS         MFULL                   ; THEN WON'T FIT
;
EASL1           EQU         *
                STA         ASEOP+1                 ; SET NEW EOP ADR
                STA         ASEOP2+1
                STX         ASEOP
                STX         ASEOP2
                LDX         ASSOP                   ; GET ADR WHERE TO LOAD
                LDY         ASSOP+1
                JSR         LD3                     ; GO LOAD
                JSR         MVCSW                   ;RESTORE I/O
                JMP         (ASEQ)                  ;RELOC FOR THIS VERSION OF APPLSOFT
;
EIBL            EQU         *
                LDA         #1                      ; SET IB PGM
                JSR         LD1                     ; GO OPEN AND TEST
;
                JSR         LD2                     ; GO GET BLOCK LENGTH
;
                SEC                                 ; HMEM - BLOCK LENGTH
                LDA         IBHMEM                  ; IS NEW SOP
                SBC         SVBL
                TAX
                LDA         IBHMEM+1
                SBC         SVBL+1
```

```
                BCC         MFULL
                TAY
;
                CPY         IBLMEM+1                ; IF NEW SOP <= LMEM
                BCC         MFULL
                BEQ         MFULL
                STY         IBSOP+1                 ; SET NEW SOP
                STX         IBSOP
LD3             EQU         *
                STX         CCBBBA                  ; SET BLOCK ADR
                STY         CCBBBA+1
                JMP         GODOS                   ; GET BLOCK & CLOSE
;
LD2             EQU         *
                LDA         SVBLA                   ; MOVE ADR OF WHERE
                STA         CCBBBA                  ; TO PUT DATA TO
                LDA         SVBLA+1                 ; CCBN
                STA         CCBBBA+1
                LDA         #0
                STA         CCBBLN+1                ; READ INTO
                LDA         #2
                STA         CCBBLN
                LDA         #CRQRD                  ; READ
                STA         CCBREQ
                LDA         #CRMNBL                 ; BLOCK
                STA         CCBRQM
                JSR         DOSGO
                LDA         SVBL+1
                STA         CCBBLN+1
                TAY
                LDA         SVBL
                STA         CCBBLN
                RTS
;
MFULL           EQU         *
                JSR         ECLOSE                  ; GO CLOSE FILE
                JMP         MFERR                   ; AND GIVE ERR MSG
LD1             EQU         *
                CMP         CCBFUC                  ; TEST TYPE
                BEQ         LD1C                    ; BR IF MATCH
                LDX         CMDNO
                STX         SVCMD
                LSR         A
                BEQ         LD1A                    ; BR IF PGM IS AS
                JMP         EINT                    ; GO FOR INTG BASIC
;
LD1A            EQU         *
                LDX         #29                     ; SAVE FILE NAME
LD1B            LDA         FNAME1,X                ; INCASE IS RAM APPLESOFT
                STA         FNAME2,X
                DEX
                BPL         LD1B
                JMP         EAS                     ; GO FOR AS
;
LD1C            RTS
                PAGE
;
; ERUN - EXECUTE RUN REQUEST
;
ERUN            EQU         *
                LDA         ASIBSW                  ;IF APPLESOFT THEN RELOC FLAG SET
                BEQ         ERUN0
                STA         RSTATE                  ;INDICATE APSFT RUN
```

```
ERUN0           JSR         ELOAD                   ; LOAD PGM
ERUN1           JSR         PRCRIF                  ;REENTRY POINT FOR ASFT RELOC
                JSR         MVCSW                   ; GO RESTORE CHAR I/O SW
                JMP         (RUN)
;
; IBRUN - INT BASIC RUN
;
IBRUN           EQU         *
                LDA         IBLMEM                  ; RESET START OF VARS
                STA         IBSOV
                LDA         IBLMEM+1
                STA         IBSOV+1
                JMP         (CHAIN)
;
; EHCAIN - EXECUTE CHAIN REQUEST
;
ECHAIN          EQU         *
                JSR         ELOAD0                  ; LOAD PGM WITHOUT CLOSING READ
FILES
                JSR         PRCRIF
                JSR         MVCSW                   ; GO RESTORE CHAR I/O SW
                JMP         (CHAIN)
ASRUN1          JSR         $D665                   ; ROM
                STA         PROMPT                  ;INSURES APPLESOFT RUN DETECT (A=0)
                STA         ASONERR                 ;RESET APPLESOFT ONERR FLAG
                JMP         $D7D2
ASRUN2          JSR         $E65                    ; RAM
                STA         PROMPT                  ;INSURES APPLESOFT RUN DETECT (A=0)
                STA         ASONERR                 ;RESET APPLESOFT ONERR FLAG
                JMP         $FD4

; ##############################################################################
; #   END OF FILE:  XLODSAV
; #   LINES      :  269
; #   CHARACTERS :  12028
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ##############################################################################
```

```
================================================================================
DOCUMENT XMISCMD.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  XMISCMD
; ###########################################################################################

               PAGE
;
; EWRITE - WRITE CMD EXECUTE
;
EWRITE         EQU         *
               JSR         RWPOSN                  ; GO POSITION FILE IF REQD
               LDA         #5
               STA         OSTATE                  ; SET OSTATE=5
               JMP         CERTN                   ; DONE
;
; EREAD - READ COMD EXECUTE
;
EREAD          EQU         *
               JSR         RWPOSN                  ; GO POSITION FILE IF REQD
               LDA         #1
               STA         ISTATE                  ; SET I STATE = DISK INPUT
               JMP         CERTN                   ; DONE
;
; RWPOSN - POSTION FOR READ/ WRITE
;
RWPOSN         EQU         *
               JSR         FILSRC                  ; FIND THE FILE
               BCC         RWP1                    ; BR IF FILE FOUND
               JSR         EOPEN                   ; GO OPEN FOR KLUTZ
               JMP         RWP2                    ; THEN SKIP NEXT LINE
RWP1           EQU         *
               JSR         MVBUFP                  ; MOVE BUFF POINTERS
RWP2           EQU         *
               LDA         INOPTS                  ; GET IN OPTIONS
               AND         #R+B                    ; WAS IT B OR R
               BEQ         RWPR                    ; BR IF NOT
               LDX         #3
RWP2A          LDA         CR,X                    ; MOVE REL REC
               STA         CCBRRN,X                ; AND REL BYTE
               DEX
               BPL         RWP2A
RWP3           EQU         *
               LDA         #CRQPOS                 ; INDICATE POISTION REQUEST
               STA         CCBREQ
               JSR         DOSGO
RWPR           RTS                                 ; DONE
               PAGE
;
;
; EINIT - EXECUTE INIT COMMAND
;
EINIT          EQU         *
               LDA         #V                      ; MUST HAVE
               AND         INOPTS                  ; VOL OPTION
               BEQ         DFVOL                   ;GO SET DEFAULT VOLUME=254
               LDA         CV                      ;CAN'T SPECIFY VOL=0
               BNE         EINITA
DFVOL          LDA         #$FE                    ;SET DEFAULT VOLUME NUMBER.
```

```
                STA         CV
EINITA          LDA         ASTART+1
                STA         CCBBSA
                LDA         #CRQFMT
                JSR         OPEN
                JMP         ESAVE
;
;
; ECAT - PRINT CATALOG
;
ECAT            EQU         *
                LDA         #CRQDIR
                JSR         OPEN                    ; GO PRETEND OPEN
                LDA         CCBVOL
                STA         CV
                RTS
                PAGE
;
; EAS - EXECUTE APPLESOFT REQUEST
;
EAS             EQU         *
                LDA         #ATSTV                  ; GET APPLESOFT TEST VALUE
                JSR         SWTST                   ; GO SWITCH AND TEST
                BEQ         GOINIT                  ; BR IF APPLESOFT
                LDA         #0
                STA         ASIBSW
;
EAS0            EQU         *
                LDY         #30
                JSR         CLRFNA
                LDX         #FASBL
EAS1            LDA         FASB-1,X                ; MOVE SYSTEM FILE NAME
                STA         FNAME1-1,X
                DEX
                BNE         EAS1
;
EAS2            EQU         *
                LDA         #$C0
                STA         ISTATE                  ; FOR RAM APPLESOFT
                JMP         ERUN                    ; GO LOAD AND RUN
;
; EINT - EXECUTE INTEGER REQUEST
;
EINT            EQU         *
                LDA         #ITSTV                  ; GET IB TEST VALUE
                JSR         SWTST                   ; GO SWITCH AND TEST
                BEQ         GOINT                   ;BR IF INTIGER BASIC...
                LDA         #1                      ;LANGUAGE NOT AVIALABLE, TOO BAD...
                JMP         ERROR
GOINT           LDA         #0                      ;RESET RSTATE
                STA         RSTATE                  ; FOR NON APPLESOFT PROG.
GOINIT          EQU         *
                JMP         DBINIT                  ; GO INIT DOS
SWTST           EQU         *
                CMP         AITSTL                  ; TEST CURRENT VALUE
                BEQ         SWTR
                STA         $C080                   ; TRY SWITCH 1
                CMP         AITSTL                  ; TEST AGAIN
                BEQ         SWTR                    ; BR IF NOW SAME
                STA         $C081                   ; TRY SWITCH 2
                CMP         AITSTL                  ; TEST AND
SWTR            RTS                                 ; RETURN
;
```

```
               PAGE
;
; EEXEC - EXECUTE EXEC CMD
;
EEXEC          EQU       *
               JSR       EOPEN                ; OPEN FILE
               LDA       CFTABA               ; MOVE TABLE POINTERS
               STA       EFTABA
               LDA       CFTABA+1
               STA       EFTABA+1
               LDA       FNAME1               ; USE FILNAME
               STA       ESTATE               ; SET EX STATE NON ZERO
               BNE       EXP2
;
;
; EPOS - EXECUTE POSITION
;
EPOS           EQU       *
               JSR       FILSRC
               BCC       EXP1
               JSR       EOPEN
               JMP       EXP2
EXP1           JSR       MVBUFP
EXP2           EQU       *
               LDA       INOPTS               ; GET OPTIONS
               AND       #R                   ; TEST R
               BEQ       EX2                  ; BR NOT R
;
EX0            LDA       CR                   ; IF CR NOT ZERO
               BNE       EX1A                 ; THEN DECREMENT
               LDX       CR+1
               BEQ       EX2
               DEC       CR+1
EX1A           DEC       CR
EX1            JSR       RBYTE                ; AND READ A RCORD
               BEQ       ICFD4
               CMP       #$8D                 ; UNTIL CR
               BNE       EX1
               BEQ       EX0                  ; THEN TEST CR AGAIN
;
EX2            RTS                            ; DONE
               PAGE
;
; OCTD - OUTPUT A CHAR TO DISK
;
OCTD           EQU       *
               JSR       TSTRUN               ; GO TEST RUN
               BCS       ICFDB                ;BRANCH IF NOT RUN MODE
               LDA       SVA                  ; CHAR IN SAVED ACU
               STA       CCBDAT               ; PUT INTO CCBDATA AREA
               LDA       #CRQWR               ; SET WRITE
               STA       CCBREQ
               LDA       #CRMNBT              ; SET NEXT BYTE
               STA       CCBRQM
               JMP       DOSGO                ; GO WRITE BYTE
;
; INCFD - INPUT A CHAR FROM DISK
;
ICFD           EQU       *
               JSR       TSTRUN               ; GO TEST RUN
               BCS       ICFDB                ;BRANCH IF NOT RUN MODE
               LDA       #6                   ; SET OUT STE = 6
ICFD3          EQU       *
```

```
                STA         OSTATE                  ; TO CATCH ECHO
                JSR         RBYTE
                BNE         ICFD1                   ; BR IF NOT ZERO CHAR
ICFD2           EQU         *
                JSR         CLOSE
                LDA         #3
                CMP         OSTATE
                BEQ         EX2
ICFD4           EQU         *
                LDA         #CREEOF
                JMP         ERROR                   ; GO TO ERROR
ICFD1           EQU         *
                CMP         #$E0                    ;CHECK FOR LOWER CASE
                BCC         ICFNLC                  ;BRANCH IF NOT LOWER-CASE
                AND         #$7F                    ;STRIP HI BIT TO FOOL GETLINE
ICFNLC          EQU         *
                STA         SVA                     ; PUT INTO SAVED ACU
                LDX         SVX                     ; MUST CHECK LAST FOR LOWER CASE
                BEQ         ICFD0                   ;IGNORE IF FIRST CHAR.
                DEX
                LDA         LBUFF,X
                ORA         #$80                    ;RESET HI BIT
                STA         LBUFF,X                 ; EVEN THOUGH IT MAY NOT NEED IT
ICFD0           EQU         *
                JMP         ORTN                    ; GO RESTORE REGS AND RTS
;
TSTRUN          PHA
                LDA         ASIBSW                  ; GET AS/INT BASIC SWITCH
                BEQ         TR1                     ; BR IF INT
                LDX         $76                     ;CHECK APPLESOFT RUN FLAG
                INX                                 ;(NOT RUN=0 AFTER INCREMENT)
                BEQ         NOTRUN                  ;IF SAYS RUNNING MAKE SURE WITH
PROMPT.
                LDX         PROMPT                  ;TEST APPLESOFT RUNNING.
                CPX         #']'+$80
                BEQ         NOTRUN                  ; BR IF NOT RUN
TR0             PLA
                CLC                                 ;INDICATE PROGRAM RUNNING
                RTS
TR1             EQU         *
                LDA         $D9                     ; GET INT RUN FLAG
                BMI         TR0                     ; BR IF RUN
NOTRUN          PLA                                 ;INDICATE PROGRAM NOT RUNNING
                SEC                                 ; WITH CARRY SET.
                RTS
;
ICFDB           EQU         *                       ; NOT RUN MODE
                JSR         CLOSE                   ; GO CLOSE FILE
                JSR         CLRSTS                  ; GO CLEAR STATES
                JMP         ORTN
                PAGE
;
; NXTEXC - NEXT EXECUTE CHAR
;
NXTEXC          EQU         *
                JSR         MVEFTA
                JSR         MVBUFP                  ; GO MOVE PTRS
                LDA         #3
                BNE         ICFD3
;
; RBYTE - READ NEXT BYTE
;
RBYTE           EQU         *
```

```
                LDA         #CRQRD                  ; SET READ
                STA         CCBREQ
                LDA         #CRMNBT                 ; SET NEXT BYTE
                STA         CCBRQM
                JSR         DOSGO                   ; GO TO DOS
                LDA         CCBDAT                  ; GET THE DATA BYTE
                RTS
MVEFTA          EQU         *
                LDA         EFTABA+1                ; MOVE TABLE ADR
                STA         ZPGWRK+1                ; NO ZPG
                LDA         EFTABA
                STA         ZPGWRK
                RTS

; ###############################################################################################
; #   END OF FILE:  XMISCMD
; #   LINES      :  254
; #   CHARACTERS :  10954
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################################
```

```
================================================================================
DOCUMENT XOPNCLS.pretty
================================================================================

; ###########################################################################################
; # PROJECT  :  APPLE ][ DOS 3.3 C SOURCE CODE LISTING -- (C) APPLE COMPUTER INC. 1983
; # FILE NAME:  XOPNCLS
; ###########################################################################################

                PAGE
;
; GNXTC - GET NEXT CHAR
;
GNXTC           EQU         *
                LDX         LBUFD
                LDA         LBUFF,X                 ; GET NEXT CHAR AND IF
                CMP         #$8D                    ; IT IS A CR
                BEQ         GNXTCR                  ; THEN RETURN WITHOUT
                INX                                 ; INCR TO NEXT CHAR
                STX         LBUFD
                CMP         #','+$80                ; TEST FOR COMMA
GNXTCR          RTS
;
; GNBC - GET NON BLANK CHAR
;
GNBC            EQU         *
                JSR         GNXTC                   ; GO GET NEXT CHAR
                BEQ         GNXTCR                  ; BR IF COMMA OR CR
                CMP         #$A0                    ; IS IT BLANK
                BEQ         GNBC                    ; BR IF BLANK
                RTS                                 ; DONE
;
; CLRCCB - CLEAR CCB
;
CLRCCB          EQU         *
                LDA         #0
                LDY         #CCBLEN                 ; CCBLENGTH
CLC1            STA         CCB-1,Y                 ; CLEAR BYTE
                DEY
                BNE         CLC1
                RTS
                PAGE
;
; GETNUM - CONVERT ASCII INPUT TO NUMERIC
;
GETNUM          EQU         *
                LDA         #0                      ; CLEAR WORK AREA
                STA         CNUM
                STA         CNUM+1
                JSR         GNBC
                PHP
                CMP         #$A4
                BEQ         HEXNUM
                PLP
                JMP         GN2A
;
GN2             JSR         GNBC                    ; GET NEXT NON BLANK
GN2A            EQU         *
                BNE         GN3                     ; BR NOT COMMA OR CR
                LDX         CNUM                    ; X=RESULT LOW
                LDA         CNUM+1                  ; Y=RESULT HI
                CLC
```

```
                RTS                                     ; DONE
;
GN3             SEC
                SBC         #$B0                        ; SUBTRACT ASCII 0
                BMI         GN4                         ; BR IF NOT NUM
                CMP         #10
                BCS         GN4                         ; BR IF NOT NUM
                JSR         GN5                         ; OLD*2
                ADC         CNUM                        ; PLUS NEW
                TAX
                LDA         #0
                ADC         CNUM+1
                TAY
                JSR         GN5                         ; OLD*4
                JSR         GN5                         ; OLD*8
                TXA                                     ; OLD*8 + OLD*2 + NEW
                ADC         CNUM
                STA         CNUM                        ; =OLD*10 + NEW
                TYA
                ADC         CNUM+1
                STA         CNUM+1
                BCC         GN2
;
GN4             EQU         *
                SEC
                RTS                                     ; DONE
GN5             EQU         *
                ASL         CNUM                        ; CNUM * 2
                ROL         CNUM+1
                RTS
                PAGE
;
HEXNUM          EQU         *
                PLP
HN0             EQU         *
                JSR         GNBC                        ; GO GET CHAR
                BEQ         GN2A                        ; BR IF CR OR COMMA
;
                SEC
                SBC         #$B0                        ; CHAR - ASCII0
                BMI         GN4                         ; BR IF LT0
                CMP         #10                         ; IS IT LT10
                BCC         HN1                         ; BR IF LT
                SBC         #$7                         ; SUB 7 FOR ASCII A
                BMI         GN4                         ; BR IF LT A
                CMP         #16                         ; TEST GT 15
                BCS         GN4                         ; BR GT 15
HN1             LDX         #4
HN2             JSR         GN5                         ; OLD*16
                DEX                                     ;LOOP 4 TIMES ONLY
                BNE         HN2
                ORA         CNUM                        ; OR IN NEW
                STA         CNUM                        ; SAVE NEW
                JMP         HN0                         ; GO FOR NEXT CHAR
                PAGE
;
; EPR - EXECUTE PR#
;
EPR             EQU         *
                LDA         CNUM                        ; GET PORT
                JMP         OUTPRT                      ; GO DO IT
;
; EIN - EXECUTE IN#
```

```
;
EIN             EQU         *
                LDA         CNUM                    ; GET PORT
                JMP         INPRT                   ; GO DO IT
;
; EMON - EXECUTE MONITOR CMD
;
EMON            EQU         *
                LDA         MONMOD                  ; GET CURRENT BITS
                ORA         IMBITS                  ; OR IN NEW BITS
                STA         MONMOD                  ; SET NEW MODE
                RTS
;
; ENONON - EXECUTE NO MONITOR CMD
;
ENOMON          EQU         *
                BIT         IMBITS
                BVC         ENM1
                JSR         PRCRIF
ENM1            EQU         *
                LDA         #$70
                EOR         IMBITS                  ; INVERT INPUT BITS
                AND         MONMOD                  ; AND WITH CURRENT
                STA         MONMOD                  ; SET NEW MODE
                RTS
                PAGE
;
; EMAXF - EXECUTE MAX FILES
;
EMAXF           EQU         *
                LDA         #0                      ; RESET EXECUTE
                STA         ESTATE
                LDA         CNUM                    ; SAVE NEW NO FILES
                PHA
                JSR         CLALL                   ; GO CLOSE ALL FILES
                PLA
                STA         CNFTBS                  ; SET NEW NO FILE TBLS
                JMP         BLDFTB                  ; GO BUILD NEW ONES
;
; EDEL - DELETE A FILE
;
EDEL            EQU         *
                LDA         #CRQDEL                 ; DELETE REQUEST
                JSR         OPEN                    ; GO OPEN
                JSR         FILSRC                  ; FIND FILE
                LDY         #0
                TYA
                STA         (ZPGWRK),Y              ; RESET FN
                RTS
;
; ELOCK - LOCK A FILE
;
ELOCK           EQU         *
                LDA         #CRQLCK                 ; SET LOCK
                BNE         ELGO
;
; EUNLK - UNLOCK A FILE
;
EUNLK           EQU         *
                LDA         #CRQUNL                 ; SET UNLOCK
ELGO            EQU         *
                JSR         OPEN                    ; OPEN FILE & UNLOCK
                JMP         ECLOSE                  ; CLOSE IT
```

```
;
; EVAR - VERIFY A FILE
;
EVAR            EQU         *
                LDA         #CRQVAR                 ; SET VARIFY
                BNE         ELGO
                PAGE
;
; EREN - RENAME A FILE
;
EREN            EQU         *
                LDA         FN2ADR                  ; MOVE FILE NAME2
                STA         CCBFN2
                LDA         FN2ADR+1
                STA         CCBFN2+1
                LDA         #CRQRNM
                STA         TEMP1A                  ; SET RENAME
                JSR         EO3                     ; GO OPEN AND RENAME
                JMP         ECLOSE                  ; GO CLOSE
;
; EAPND - OPEN FILE FOR APPEND
;
EAPND           EQU         *
                JSR         EOPEN                   ; GO OPEN
AP1             EQU         *
                JSR         RBYTE                   ; READ A BYTE
                BNE         AP1                     ; BR IF NOT ZERO
;
                JMP         BUMPER                  ; GO TO PATCH FOR APPEND FIX
                PAGE
;
; EOPEN - OPEN A FILE
;
EOPEN           LDA         #0                      ;FIX TYPE MISMATCH DETECTION
                JMP         SV1                     ;(CALLS EOPN1)
EOPN1           LDA         #CRQOPN
OPEN            EQU         *
                STA         TEMP1A
                LDA         CL                      ; IF NO LENGTH ENTERED
                BNE         EO1                     ; THEN SET DEFAULT OF 1
                LDA         CL+1
                BNE         EO1
                LDA         #1
                STA         CL
EO1             EQU         *
                LDA         CL                      ; MOVE REC LENGTH
                STA         CCBRLN
                LDA         CL+1
                STA         CCBRLN+1
EO3             EQU         *
                JSR         ECLOSE                  ; GO CLOSE IF OPEN
EO4             EQU         *
                LDA         CNUM+1                  ; GET AVALL ENTRY
                BNE         EO5                     ; BR IF ONE AVAIL
                JMP         ENFA                    ; DONE - NO FILES AVAIL
EO5             EQU         *
                STA         ZPGWRK+1                ; MOVE AVAIL SLOT TO ZPG
                LDA         CNUM
                STA         ZPGWRK
EO6             EQU         *
                JSR         MVFN1                   ; GO MOVE FILE NAME
                JSR         MVBUFP                  ; GO MOVE BUF PTRS
                JSR         OPNSUP                  ; GO SET UP OPEN
```

```
                LDA         TEMP1A                  ; SET OPEN REQ
                STA         CCBREQ
                JMP         DOSGO                   ; GO OPEN
                PAGE
;
; ECLOSE - EXECUTE CLOSE FILE COMMAND
;
ECLOSE          EQU         *
                LDA         FNAME1
                CMP         #$A0
                BEQ         CLALL
                JSR         FILSRC                  ; GO FIND FILE
                BCS         CL2                     ; BR IF NOT FOUND
                JSR         CLOSE                   ; GO CLOSE
                JMP         ECLOSE                  ; GO SEE IF ANY MORE OPEN
;
; CLOSE - CLOSE A FILE
;
CLOSE           EQU         *
                JSR         TSTEXC
                BNE         CLX
                LDA         #0
                STA         ESTATE
CLX             EQU         *
                LDY         #0                      ; CLEAR 1ST FN
                TYA                                 ; CHAR TO ZERO
                STA         (ZPGWRK),Y
                JSR         MVBUFP                  ; MOVE BUFFER PTRS
                LDA         #CRQCLS                 ; SET CLOSE
                STA         CCBREQ
                JMP         DOSGO                   ; GO CLOSE
;
; CLALL - CLOSE ALL FILES
;
CLALL           EQU         *
                JSR         TSINIT                  ; GO INIT FILE SEARCH
                BNE         CL1
CL0             EQU         *
                JSR         TSNXT                   ; NEXT ENTRY
                BEQ         CL2                     ; BR IF NO MORE
CL1             EQU         *
                JSR         TSTEXC
                BEQ         CL0
                JSR         TSTOPN                  ; GO TEST OPEN
                BEQ         CL0                     ; BR NOT OPEN
                JSR         CLOSE                   ; GO CLOSE
                JMP         CLALL                   ; START OVER
CL2             RTS                                 ; DONE

; ###############################################################################
; #   END OF FILE:  XOPNCLS
; #   LINES      :  290
; #   CHARACTERS :  11533
; #   Formatter  :  Assembly Language Reformatter 1.0.2 (07 January 1998)
; ###############################################################################


                                THE END
```