```
PAGE -    0
Current memory available:    24976
00001                                          .ABSOLUTE
2 blocks for procedure code  23041 words left
```

```
0000:                                   .PROC   PROFILEDRIVER, 0
Current memory available:   24451
0000:                                   .MACROLIST
0000:                                   .NOPATCHLIST
0000:                                   .TITLE  "  APPLE II PROFILE DRIVER PROM code  Feb 16, 1984   "
```

```
0000:                                       .PAGE
0000:                                       .INCLUDE        PROMX.A.TEXT
0000:                                       .ORG    0C800
C800:
C800: 28 43 29 20 41 50 50  COPYRIGHT       .ASCII  "(C) APPLE COMPUTER INC.  1982"
C807: 4C 45 20 43 4F 4D 50
C80E: 55 54 45 52 20 49 4E
C815: 43 2E 20 31 39 38 32
C81C: 00D1                  DEVTYPE         .EQU    0D1
C81C: 0001                  SUBTYPE         .EQU    01
C81C: 0001                  MANUF           .EQU    0001            ;Apple Computer Inc.
C81C: 1001                  RELEASE         .EQU    1001
C81C: 0000                  MAXBLOCK        .EQU    0000            ; variable
```

```
C81C:                           .PAGE
C81C:                   ;---------------------------------------------------------------------------
C81C:                   ;
C81C:                   ;  The macro SWITCH performs an N way branch based on a switch index.
C81C:                   ;
C81C:                   ;               SWITCH  [index], [bounds], adrs_table, [*]
C81C:                   ;
C81C:                   ;---------------------------------------------------------------------------
C81C:
C81C:                           .MACRO  SWITCH
C81C:                           .IF     "%1" <> ""      ;If PARM1 is present,
C81C:                           LDA     %1              ;  Load A with switch index
C81C:                           .ENDC
C81C:                           .IF     "%2" <> ""      ;If PARM2 is present,
C81C:                           CMP     #%2+1           ;  Perform bounds checking
C81C:                           BCS     $010            ;  on switch index
C81C:                           .ENDC
C81C:                           ASL     A
C81C:                           TAY
C81C:                           LDA     %3+1,Y          ;Get switch address from table
C81C:                           PHA                     ;  and push onto stack
C81C:                           LDA     %3,Y
C81C:                           PHA
C81C:                           .IF     "%4" <> "*"     ;If PARM4 is omitted,
C81C:                           RTS                     ;  Exit to code
C81C:                           .ENDC                   ;Otherwise, drop through
C81C:                   $010    .ENDM
```

```
C81C:                           .PAGE
C81C:                   ;-------------------------------------------------------------------------------
C81C:                   ;
C81C:                   ;   Macro INC16.  Increments a 16 bit unsigned number.
C81C:                   ;
C81C:                   ;-------------------------------------------------------------------------------
C81C:
C81C:                           .MACRO  INC16
C81C:                           INC     %1
C81C:                           BNE     $10
C81C:                           INC     %1+1
C81C:                   $10     .ENDM
C81C:
C81C:                   ;-------------------------------------------------------------------------------
C81C:                   ;
C81C:                   ;   Macro CMP16.  Compares two 16 bit unsigned numbers.
C81C:                   ;   If %1 < %2 then carry clear, else carry set.
C81C:                   ;
C81C:                   ;-------------------------------------------------------------------------------
C81C:
C81C:                           .MACRO  CMP16
C81C:                           LDA     %1
C81C:                           CMP     %2
C81C:                           LDA     %1+1
C81C:                           SBC     %2+1
C81C:                           .ENDM
C81C:                   ;-------------------------------------------------------------------------------
C81C:                   ;
C81C:                   ;   Macro JSRI - JSR indirect via the address indicated by the
C81C:                   ;   argument.
C81C:                   ;
C81C:                   ;-------------------------------------------------------------------------------
C81C:
C81C:                           .MACRO  JSRI
C81C:                           LDA     $1+1
C81C:                           PHA
C81C:                           LDA     $1
C81C:                           PHA
C81C:                           JMP     @%1
C81C:                   $1      .WORD   *+1
C81C:                           .ENDM
```

```
C81C:                                   .PAGE
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;          Error Codes
C81C:                           ;
C81C:                           ;-------------------------------------------------------------------------------
C81C:
C81C: 0003                      BADMODE         .EQU    3               ;Pascal error code
C81C: 0003                      XREQCODE        .EQU    3               ;Invalid request code
C81C: 0001                      PRTYCRC         .EQU    1
C81C: 0005                      LOSTDVC         .EQU    5
C81C: 0009                      OFFLINE         .EQU    9
C81C: 0021                      XCTLCODE        .EQU    21              ;Invalid control/status code
C81C: 0025                      XNORESRC        .EQU    25              ;resource not available
C81C: 0026                      XBADOP          .EQU    26              ;Invalid operation
C81C: 0027                      XIOERROR        .EQU    27              ;I/O error
C81C: 0028                      XNODRIVE        .EQU    28              ;No drive connected
C81C: 002B                      XNOWRITE        .EQU    2B              ;Device write protected.(not supported)
C81C: 002C                      XBYTECNT        .EQU    2C              ;Byte count <> a multiple of 512
C81C: 002D                      XBLKNUM         .EQU    2D              ;Block number too large
C81C: 0027                      BADOLDDATA      .EQU    27              ;block has bad data from previous read
C81C: 0040                      DEVERR          .EQU    40
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  Hardware I/O Addresses
C81C:                           ;
C81C:
C81C: C080                      WR_PORT         .EQU    0C080           ;Write to Z8 RAM (byte at a time)
C81C: C081                      RD_PORT         .EQU    0C081           ;Read from Z8 RAM (byte at a time)
C81C: C082                      BUSY            .EQU    0C082           ;Z8 not ready
C81C: C082                      RSTAT           .EQU    0C082
C81C: C083                      CLR_PARITY      .EQU    0C083           ;Clear parity error.
C81C:
C81C:                           ;
C81C:                           ;  CONSTANTS
C81C:                           ;
C81C:
C81C: 0000                      NOTCMD          .EQU    0               ;Command line Low
C81C: 0004                      SETCMD          .EQU    4               ;Command line High
C81C: 0001                      SETWRT          .EQU    1               ;Low
C81C: 0005                      SETRD           .EQU    5               ;High
C81C: 0002                      INTDSABL        .EQU    2
C81C: 0006                      INTENABL        .EQU    6
C81C: 0003                      RWLO            .EQU    3
C81C: 0007                      RWHI            .EQU    7
C81C: 000C                      RST             .EQU    0C
C81C: 0008                      CLRRST          .EQU    8
C81C: 000A                      RTRYCNT         .EQU    0A
C81C: 0003                      RTRYTHRESH      .EQU    3
```

```
C81C:
C81C:                          ;Command codes:
C81C:
C81C: 0000                     WDGTRD          .EQU    0
C81C: 0002                     WDGTWRTVER      .EQU    2
C81C: 0001                     WDGTWRT         .EQU    1
C81C:
```

```
C81C:                                   .PAGE
C81C:                           ;---------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  Device Information Block (DIB)
C81C:                           ;
C81C:                           ;---------------------------------------------------------------------------
C81C:
C81C: 0000                      DIBBLKLO    .EQU    0
C81C: 0000                      DIBBLKHI    .EQU    0
```

```
C81C:                                   .PAGE
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  zero page temps
C81C:                           ;
C81C:                           ;-------------------------------------------------------------------------------
C81C: 0000                      MAPTEMP1        .EQU    00      ;used in Pascal routines
C81C: 0002                      MAPTEMP2        .EQU    02      ;           "
C81C: 0004                      MAPTEMP3        .EQU    04      ;           "
C81C: 0006                      MAPTEMP4        .EQU    06      ;           "
C81C: 0008                      MAPTEMP5        .EQU    08      ;           "
C81C: 000A                      MAPRET          .EQU    0A      ;return addr for MAP routine
C81C: 000C                      SLOTNUM         .EQU    0C      ;ON for slot N
C81C: 000D                      BUSYALL         .EQU    0D      ;some Profile unit is ready
C81C: 000E                      MUNIT           .EQU    0E      ;Pascal unit number
C81C:
C81C: 0030                      MAPENTRY        .EQU    30      ;local copy of one entry of the mapping table,
C81C:                                                           ;six bytes, 30 - 35
C81C:
C81C:                           ;Profile routines use these zero page locations
C81C: 003A                      SWTCH           .EQU    3A
C81C: 003B                      COUNTR          .EQU    3B
C81C: 003C                      BCLO            .EQU    3C
C81C: 003D                      ERROR           .EQU    3D
C81C: 003E                      INDRCN          .EQU    3E
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  PRODOS PARAMETERS on zero page
C81C:                           ;
C81C:                           ;-------------------------------------------------------------------------------
C81C:
C81C: 0042                      CMD             .EQU    42
C81C: 0042                      Z8CMD           .EQU    42
C81C: 0043                      UNIT            .EQU    43
C81C: 0043                      SLOTX           .EQU    43
C81C: 0044                      BUFADLO         .EQU    44
C81C: 0045                      BUFADHI         .EQU    45
C81C: 0046                      BLOCKLO         .EQU    46
C81C: 0047                      BLOCKHI         .EQU    47
C81C:
C81C:                           ;table of ATTACH driver pointers to driver information.
C81C: 0048                      PTRTABL     .EQU 048
C81C: 0048                      MAPTABLADR  .EQU PTRTABL+0      ;addr of MAPTABL mapping table
C81C: 004A                      SLOTTAB     .EQU PTRTABL+2      ;addr of peripheral card slot status table
C81C: 004C                      CONCK       .EQU PTRTABL+4      ;addr of ATTACH routine to call CONCK
C81C: 004E                      INITINFOADR.EQU PTRTABL+6
C81C:
C81C:                           ;offsets of INITINFO table in ATTACH driver, pointed to by INITINFOADR.
C81C: 0000                      MAINENTRY   .EQU 0              ;addr of ATTACH driver main entry
C81C: 0002                      INITIALIZE  .EQU 2              ;addr of master driver init+MAPINIT routine
C81C: 0004                      ORIG4ADR    .EQU 4              ;nontemporary data space for ROM code
C81C: 0006                      ROMRETURN   .EQU 6              ;           "     "
```

```
C81C:                           ;
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  scratchpad RAM variables
C81C:                           ;
C81C:                           ;-------------------------------------------------------------------------------
C81C:
C81C: 0478                      BCHISV          .EQU    478
C81C: 04F8                      ORGADRLO        .EQU    4F8
C81C: 0578                      ORGADRHI        .EQU    578
C81C: 05F8                      PASCAL          .EQU    5F8             ;uses bit 7
C81C: 05F8                      BLK_RTRYCNT     .EQU    5F8             ;uses bits 0 and 1
C81C: 0678                      BCHI            .EQU    678
C81C: 06F8                      WAITTIME        .EQU    6F8
C81C: 0778                      PROSTAT         .EQU    778
C81C:
C81C:                           ;-------------------------------------------------------------------------------
C81C:                           ;
C81C:                           ;  slot dependent variables (loc + n)
C81C:                           ;
C81C:                           ;-------------------------------------------------------------------------------
C81C:
C81C:                           ;       The actual addresses used are the first 5 scratchpad locations
C81C:                           ;       for the slot, but they are addressed by ABS, Y where Y = Cn.
C81C:
C81C: 03B8                      STATUS1         .EQU    3B8
C81C: 0438                      STATUS2         .EQU    438
C81C: 04B8                      STATUS3         .EQU    4B8
C81C: 0538                      STATUS4         .EQU    538
C81C: 05B8                      BCLOSV          .EQU    5B8
C81C:
C81C:                           ;
C81C:                           ;       PASCAL field information
C81C:                           ;
C81C:
C81C:                           ;default mapping table offsets - table has 30 table entries, 6 bytes each
C81C:                           ;note - some code ASSUMES the order given here!
C81C: 0000                      MTDRIVE .EQU    0               ;fields of the mapping table
C81C: 0001                      L       .EQU    1
C81C: 0001                      P       .EQU    1
C81C: 0002                      MTSTART .EQU    2
C81C: 0004                      MTLENG  .EQU    4
C81C:
C81C:
C81C:
```

```
C81C:                                   .PAGE
C81C:                                   .INCLUDE        PROMX.B.TEXT
C81C:
C81C: C81C                PRODOS        .EQU    *
C81C: 85 3F                             STA     INDRCN+1        ;acc. has Cn
C81E: A9 00               PD1           LDA     #0
C820: 8D F805                           STA     PASCAL          ;tell driver PRODOS called it
C823: 85 3E                             STA     INDRCN
C825: BD 82C0                           LDA     BUSY,X
C828: 4A                                LSR     A               ;is a Profile attached?
C829: 90**                              BCC     $10             ;branch if so
C82B: A9 28                             LDA     #XNODRIVE
C82D: 60                                RTS                     ;error return
C82E: A4 42               $10           LDY     CMD
C830: D0**                              BNE     NOTSTAT         ;branch if not status command
C832: 0A                                ASL     A
C833: 10**                              BPL     BSYHI           ;branch if Profile not ready
C835: A9 01                             LDA     #01
C837: 85 42                             sta     cmd
C839: A5 47                             lda     blockhi
C83B: 48                                pha
C83C: A5 46                             lda     blocklo
C83E: 48                                pha
C83F: A5 45                             lda     bufadhi
C841: 48                                pha
C842: A5 44                             lda     bufadlo
C844: 48                                pha
C845: A9 FF                             lda     #0FF
C847: 85 44                             sta     bufadlo
C849: 85 45                             sta     bufadhi
C84B: 85 46                             sta     blocklo
C84D: 85 47                             sta     blockhi
C84F: 20 ****                           jsr     notstat
C852: 85 3D                             sta     error
C854: A9 00                             lda     #00
C856: 85 42                             sta     cmd
C858: 68                                pla
C859: 85 44                             sta     bufadlo
C85B: 68                                pla
C85C: 85 45                             sta     bufadhi
C85E: A6 43                             ldx     slotx
C860: BD 82C0                           lda     busy,x
C863: 85 3B                             sta     countr
C865: A6 46                             ldx     blocklo
C867: A4 47                             ldy     blockhi
C869: 68                                pla
C86A: 85 46                             sta     blocklo
C86C: 68                                pla
C86D: 85 47                             sta     blockhi
C86F: A5 3D                             lda     error
C871: D0**                              bne     errrtn
C873: A5 3B                             lda     countr
C875: 18                                clc
C876: 60                                rts
```

```
C877: A9 27                    BSYHI           LDA     #XIOERROR
C879: 38                       errrtn          SEC
C87A: 60                                       RTS
C87B:                          ;
C87B: C0 03                    NOTSTAT         cpy     #3              ; is the command one we handle?
C87D: B0F8                                     bcs     bsyhi           ; branch if not
C87F: A2 00                                    LDX     #0
C881: A0 02                                    LDY     #2
C883: 86 3C                                    STX     BCLO
C885: 8C 7806                                  STY     BCHI            ;setup to read 512 bytes
C888: A5 42                                    LDA     CMD
C88A: 48                                       PHA                     ;save CMD for PRODOS
C88B: 46 42                                    LSR     CMD
C88D: 20 ****                                  JSR     RWSU1           ;do read or write command
C890: A6 43                                    LDX     SLOTX
C892: BD 83C0                                  LDA     CLR_PARITY,X    ;turn off 9334 on card
C895: 68                                       PLA
C896: 85 42                                    STA     CMD             ;restore CMD for PRODOS
C898: 18                                       CLC
C899: A5 3D                                    LDA     ERROR
C89B: D0DA                                     BNE     BSYHI           ;branch if there was an error
C89D: 60                                       RTS
C89E:
C89E:                          ;---------------------------------------------------------------
C89E:                          ;
C89E:                          ; PASCAL MAIN ENTRY POINT (entered via vector in CnOO ROM)
C89E:                          ; Entered with Pascal params and local return addr on stack
C89E:                          ; Areg = Pascal unitnum, Xreg = operation.
C89E:                          ;
C89E:                          ; ... (illegible) ...
C89E:
C89E:                          PASCMAIN ;STA    MUNIT
C89E:                                   ;STX    Z8CMD
C89E:                                   ;LDA    #80
C89E:                                   ;STA    PASCAL          ;tell routines Pascal called them
C89E:                                   ;PLA
C89E:                                   ;LDY    #ROMRETURN      ;stash return address in driver space
C89E:                                   ;STA    @INITINFOADR,Y
C89E:                                   ;PLA
C89E:                                   ;INY
C89E:                                   ;STA    @INITINFOADR,Y
C89E:                                   ;CPX    #4              ;branch, based on operation
C89E:                                   ;BEQ    STATIT          ;status
C89E:                                   ;CPX    #2              ;init
C89E:                                   ;BEQ    INITIT
C89E:                                   ;BCS    BADREQ          ;>=2 (except 2,4) invalid
C89E:                                                           ;values 0 (read), 1 (write) fall into MAPPIT
C89E:
C89E:                          MAPPIT   ;map block number, then go to driver.
C89E:                                   ;JSR    MAP             ;return with Drive# in Areg.  IORSLT in Xreg.
C89E:                                   ;CPX    #0              ;any errs?
C89E:                                   ;BNE    DUMP
C89E:                                   ;LDY    Z8CMD           ;no, go do read/write as indicated by Z8CMD
C89E:                                   ;       JMP     DOIO            ;always branch
C89E:                          DUMP                             ;otherwise dump stack and return.
```

```
C89E:                                   ;LDY    #10
C89E:                                   ;$1     PLA
C89E:                                   ;DEY
C89E:                                   ;BNE    $1
C89E:
C89E:                           RTN     ;enter here when no further action required. Clean up and leave the ROM.
C89E:                                   ;Don't destroy Xreg (ioresult).
C89E:                                   ;LDY    #OFF                    ;indicate no further I/O required
C89E:
C89E:                           DOIO    ;enter here when further I/O action must be executed via a RAM
C89E:                                   ;based call. Yreg=I/O command, Areg=Drive#. Xreg=0 to indicate no
C89E:                                   ;IORESULT so far.  Leaves the ROM.
C89E:                                   ;STY    MAPTEMP1
C89E:                                   ;STA    MAPTEMP2
C89E:                                   ;LDY    #ROMRETURN+1
C89E:                                   ;LDA    @INITINFOADR,Y
C89E:                                   ;PHA
C89E:                                   ;DEY
C89E:                                   ;LDA    @INITINFOADR,Y
C89E:                                   ;PHA
C89E:                                   ;LDY    MAPTEMP1
C89E:                                   ;LDA    MAPTEMP2
C89E:                                   ;RTS
C89E:
C89E:                           INITIT
C89E:                           PROCESSUNIT
C89E:
C89E:                                   ;LDA    MUNIT
C89E:                                   ;JSR    INDEX           ;find drive# associated with this unit
C89E:                                   ;LDX    #9
C89E:                                   ;BCS    RTN             ;set IORESULT if bad unit
C89E:                                   ;JSR    GETENTRY        ;get maptable entry for this unit
C89E:                                   ;BIT    MAPENTRY+P      ;is it mounted?
C89E:                                   ;BPL    NOTPRESENT      ;branch if no
C89E:                                   ;LDA    MAPENTRY+MTDRIVE;else go request init with Areg:= drive#
C89E:                                   ;LDY    Z8CMD
C89E:                                   ;LDX    #0
C89E:                                   ;BEQ    DOIO
C89E:
C89E:                           NOTPRESENT
C89E:                                   ;LDX    #9
C89E:                                   ;BNE    RTN             ;always branches
C89E:
C89E:                           STATIT  ;TSX
C89E:                                   ;LDA    0104,X          ;fetch hi byte of control word from stack.
C89E:                                   ;AND    #0E0            ;get top 3 bits
C89E:                                   ;CMP    #040            ;top 3 bits = 010 ?
C89E:                                   ;BEQ    MAPSTAT         ;yes, do map status and return directly to BIOS
C89E:                                   ;BNE    PROCESSUNIT     ;otherwise do driver's status request
C89E:
C89E:
C89E: A2 03                     BADREQ  LDX     #XREQCODE       ;Invalid request code
C8A0: D0FC                              BNE     RTN             ;always branch
C8A2:
```

```
C8A2:                                .PAGE
C8A2:                          ; status call permits retrieval or setting of the maptable.
C8A2:                          ; 'old12' address of unit 12 driver is also set/returned in a word immediately
C8A2:                          ; following the status table.
C8A2:                          ; Status Control Word -
C8A2:                          ;    bit 0 -      ignored.
C8A2:                          ;    bit 1 -      =0 to get status
C8A2:                          ;                 =1 to change status (update maptable)
C8A2:                          ;    bits 13-15 - =010 to load or save the maptable. All other combinations
C8A2:                          ;                  represent unimplemented status operations.
C8A2:                          ; standard status return items as described in UCSD BIOS implementation guide
C8A2:                          ; not implemented.
C8A2:                          ;
C8A2:                          ; Stack on entry - pointer to statrec (180 bytes), control word.
C8A2:                          ; Return address in ROMRETURN.
C8A2:                          ;
C8A2:                          ; Pascal temp usage - MAPTEMP1, MAPTEMP3
C8A2: 68                       MAPSTAT PLA                     ;buffer addr
C8A3: 85 00                            STA     MAPTEMP1
C8A5: 68                               PLA
C8A6: 85 01                            STA     MAPTEMP1+1
C8A8: 68                               PLA                     ;control word lsb
C8A9: 29 02                            AND     #2              ;isolate get/change bit
C8AB: 85 04                            STA     MAPTEMP3
C8AD: 68                               PLA                     ;control word msb
C8AE: A2 00                            LDX     #0              ;no errors!
C8B0: A0 B6                            LDY     #182.
C8B2: A5 04                            LDA     MAPTEMP3        ;nonzero if changing status
C8B4: D0**                             BNE     $4
C8B6:
C8B6:                                  ;getting status
C8B6: 88                       $3      DEY
C8B7: B1 48                            LDA     @MAPTABLADR,Y
C8B9: 91 00                            STA     @MAPTEMP1,Y
C8BB: C0 00                            CPY     #0
C8BD: D0F7                             BNE     $3
C8BF: F0**                             BEQ     STATRTN
C8C1:                          $4
C8C1:                                  ;setting status
C8C1: 88                       $5      DEY
C8C2: B1 00                            LDA     @MAPTEMP1,Y
C8C4: 91 48                            STA     @MAPTABLADR,Y
C8C6: C0 00                            CPY     #0
C8C8: D0F7                             BNE     $5
C8CA:
C8CA: 4C 9EC8                  STATRTN JMP     RTN
C8CD:
```

```
C8CD:                                   .PAGE
C8CD:                           INDEX   ;subroutine to convert the unitnum in Areg to an index into MAPTABL.
C8CD:                                   ;carry set on exit if error.
C8CD:                                   ;
C8CD:                                   ; Pascal temp usage - MAPTEMP1
C8CD:
C8CD: C9 04                             CMP     #4
C8CF: 90**                              BCC     $100
C8D1: C9 06                             CMP     #6
C8D3: 90**                              BCC     $200
C8D5: C9 09                             CMP     #9              ;test unit number for valid range
C8D7: 90**                              BCC     $100
C8D9: C9 15                             CMP     #21.
C8DB: 90**                              BCC     $200-2
C8DD: C9 80                             CMP     #128.
C8DF: 90**                              BCC     $100
C8E1: C9 90                             CMP     #144.
C8E3: 90**                              BCC     $200-4
C8E5:
C8E5: 38                        $100    SEC
C8E6: 60                                RTS
C8E7:
C8E7:                                   ;the first subtract processed subtracts one more than the number shown
C8E7:                                   ;because the carry is clear on entry. for subsequent subtracts the c
C8E7:                                   ;carry is set.
C8E7:                                   ;the result is that the unitnum in the Areg is converted to a record
C8E7:                                   ;index into the maptable.  After multiplying by 6, it becomes a byte
C8E7:                                   ;index.
C8E7: E9 68                             SBC     #107.           ;if entered here, subtract a total of 114.
C8E9: E9 03                             SBC     #3              ;if entered here, subtract a total of 7.
C8EB: E9 03                     $200    SBC     #3              ;if entered here, subtract a total of 4.
C8ED: 0A                                ASL     A               ;multiply by 6 for byte index into maptable
C8EE: 85 00                             STA     MAPTEMP1
C8F0: 0A                                ASL     A
C8F1: 65 00                             ADC     MAPTEMP1
C8F3: 60                                RTS
C8F4:
C8F4:
C8F4:                           ;-------------------------------------------------------------------------
C8F4:                           ;
C8F4:                           GETENTRY  ;subroutine to transfer one map table entry to a local area
C8F4:                           ;   On entry, carry must be clear and A register must index to the desired
C8F4:                           ;   entry.
C8F4:                           ;   Destroys all registers.
C8F4:
C8F4: 69 05                             ADC     #5              ;else prepare to transfer 6-byte record to zpage
C8F6: A8                                TAY                     ;Yreg=index of last byte
C8F7: A2 05                             LDX     #5
C8F9:                           $3
C8F9: B1 48                             LDA     @MAPTABLADR,Y ;loop to transfer map tbl entry to local area
C8FB: 95 30                             STA     MAPENTRY,X
C8FD: 88                                DEY
C8FE: CA                                DEX
C8FF: 10F8                              BPL     $3
```

```
C901: 60                                RTS
C902:
C902:                           ;-----------------------------------
C902:                           ;
C902:                           MAP     ;map read/write params to physical block # and drive#,
C902:                           ;        and verify write-protect and non-overrun.
C902:                           ;        On entry, stack contains params as follows:
C902:                           ;         TOS = return address to this routine  101
C902:                           ;               BLOCK NUMBER                    103
C902:                           ;               BYTE COUNT                      105
C902:                           ;               BUFFER ADDRESS                  107
C902:                           ;               DRIVE NUMBER from rsp           109
C902:                           ;               CONTROL WORD                    10B
C902:                           ;
C902:                           ;       Return with drive# in A register.
C902:                           ;
C902:                           ;        Pascal temp usage - MAPTEMP1, MAPTEMP2
C902:
C902: BA                                TSX
C903: 86 02                             STX     MAPTEMP2
C905: BD 0B01                           LDA     010B,X          ;get lsb control word off stack
C908: 4A                                LSR     A               ;see if physical sector mode (Areg bit 1)
C909: 4A                                LSR     A               ;bit 1 into carry
C90A: A5 0E                             LDA     MUNIT           ;Areg := Pascal unit#
C90C: 90**                              BCC     $1              ;now branch if not phys sect mode
C90E:
C90E: BD 0C01                           LDA     010C,X          ;it's phys sect mode, get hi byte control wd
C911: 18                                CLC                     ;top 3 bits form drive number
C912: 29 E0                             AND     #0E0
C914: 2A                                ROL     A
C915: 2A                                ROL     A
C916: 2A                                ROL     A
C917: 2A                                ROL     A               ;move to bits 0-2
C918: 90**                              BCC     OKRTN           ;always branch to put physical unit # in
C91A:                                                           ;param stack & return w/o mapping
C91A:
C91A: 20 CDC8                   $1      JSR     INDEX           ;convert Areg into index into MAPTABL
C91D:
C91D: A2 09                     $2      LDX     #9              ;if error,
C91F: B0**                              BCS     MAPRTN          ;carry will be set so return with Xreg=9
C921:
C921: 20 F4C8                           JSR     GETENTRY        ;move this entry onto zero page
C924: 38                                SEC
C925: 24 31                             BIT     MAPENTRY+P      ;check presence bit
C927: 10F4                              BPL     $2              ;error return if no volume mounted
C929: A5 42                             LDA     Z8CMD           ;writing?
C92B: C9 01                             CMP     #1
C92D: D0**                              BNE     $3
C92F: 24 31                             BIT     MAPENTRY+L      ;then check write-protect bit
C931: A2 10                             LDX     #16.
C933: 70**                              BVS     MAPRTN          ;return with IORSLT=16 if write protected
C935: A6 02                     $3      LDX     MAPTEMP2        ;restore stack ptr
C937:
C937:                           ;check the length of the transfer, make sure no overrun.
C937:                           ;if the last block to transfer is beyond the last block on the vol
```

```
C937:                                ;then overrun exists: set IORESULT=64.
C937: BD 0601                        LDA     106,X           ;get hi byte of byte count
C93A: 30**                           BMI     OVRRUN          ; >32K not supported
C93C: 4A                             LSR     A               ;Areg:= # of full blocks to xfer
C93D: A8                             TAY
C93E: F0**                           BEQ     $5
C940: B0**                           BCS     $5
C942: BD 0501                        LDA     105,X           ;get lo byte of byte count
C945: D0**                           BNE     $5              ;decrement #blocks to xfer if count was
C947: 88                             DEY                     ;one or more blocks even, so that...
C948:                          $5
C948: 98                             TYA                     ;..here, starting-block+Yreg=index of last block
C949: 18                             CLC
C94A: 7D 0301                        ADC     103,X           ;add starting-block+Yreg
C94D: A8                             TAY
C94E: BD 0401                        LDA     104,X           ;calculate last block to
C951: 69 00                          ADC     #0              ;xfer:lo byte in Yreg, hi byte in
C953: 85 00                          STA     MAPTEMP1        ;MAPTEMP1
C955: B0**                           BCS     OVRRUN          ;error if last block is beyond $FFFF
C957:
C957: 98                             TYA                     ;unsigned compare of last-block and # blocks on
C958: C5 34                          CMP     MAPENTRY+MTLENG ;pseudovolume
C95A: A5 00                          LDA     MAPTEMP1
C95C: E5 35                          SBC     MAPENTRY+MTLENG+1
C95E: 90**                           BCC     NO_OVRRUN       ;carry set if last-block >= volume-blocklength
C960:                          OVRRUN
C960: A2 40                          LDX     #64.
C962: D0**                           BNE     MAPRTN
C964:                          NO_OVRRUN
C964:
C964:                                ;map the starting-block-number parameter to the physical block number
C964: BD 0301                        LDA     0103,X          ;get block number from stack
C967: 18                             CLC
C968: 65 32                          ADC     MAPENTRY+MTSTART;adjust block count
C96A: 9D 0301                        STA     0103,X
C96D: BD 0401                        LDA     0104,X
C970: 65 33                          ADC     MAPENTRY+MTSTART+1
C972: 9D 0401                        STA     0104,X
C975:
C975: A5 30                          LDA     MAPENTRY+MTDRIVE;get physical drive index (leave in Areg)
C977:
C977: A2 00                    OKRTN LDX     #0                ;IORESULT
C979: 60                       MAPRTN RTS
```

```
C97A:                                   .PAGE
C97A:
C97A:                           ;-----------------------------------------------------------
C97A:                           ;
C97A:                           ;       middle loop of map initialization.  Called from
C97A:                           ;       ATTACH driver. Current-drive counter on TOS, below
C97A:                           ;       return address.  Pseudovolumes directory at $6000.
C97A:                           ;       Loads map table with default volume entries and
C97A:                           ;       returns original address of driver for unit 4
C97A:                           ;       in ORIG4ADR.
C97A:                           ;
C97A:                           ;-----------------------------------------------------------
C97A:                           ;
C97A:                           ; Pascal temp usage - MAPTEMP1, MAPTEMP3, MAPTEMP4, MAPTEMP5
C97A:
C97A: 0008                      INITCNT         .EQU    MAPTEMP5
C97A:
C97A: 68                        SCAN_DFMT       PLA
C97B: 85 06                                     STA     MAPTEMP4
C97D: 68                                        PLA
C97E: 85 07                                     STA     MAPTEMP4+1;save return address
C980: A2 00                                     LDX     #0
C982: BD 0260                                   LDA     6000+2,X;get number of vols
C985: 85 08                                     STA     INITCNT
C987: 8A                        $8              TXA             ;loop on 8 byte records (skipping header record)
C988: 18                                        CLC
C989: 69 08                                     ADC     #8
C98B: AA                                        TAX
C98C: C6 08                                     DEC     INITCNT ;all done?
C98E: 10**                                      BPL     $16
C990: A5 07                                     LDA     MAPTEMP4+1;if so, return
C992: 48                                        PHA
C993: A5 06                                     LDA     MAPTEMP4
C995: 48                                        PHA
C996: 60                                        RTS
C997:
C997: BD 0460                   $16             LDA     6000+4,X;get DEFAULT_UNIT field
C99A: F0EB                                      BEQ     $8      ;branch if no unit
C99C: C9 90                                     CMP     #144.
C99E: B0E7                                      BCS     $8      ;branch if out of range
C9A0:
C9A0:                                   ;Areg=unit#; use to index into maptable.
C9A0: 20 CDC8                                   JSR     INDEX   ;convert into maptable index
C9A3: B0E2                                      BCS     $8      ;branch if invalid (NOTE: clear-carry used below)
C9A5: 48                                        PHA             ;save index to this entry
C9A6: 69 01                                     ADC     #P
C9A8: A8                                        TAY                     ;index to field P of this entry
C9A9: B1 48                                     LDA     @MAPTABLADR,Y   ;already have this unit present?
C9AB: 30DA                                      BMI     $8              ;if so, treat this mount as invalid
C9AD: BD 0560                                   LDA     6000+5,X        ;get writeprot bit (bit 7)
C9B0: 6A                                        ROR     A               ;move into L field (bit 6)
C9B1: 29 40                                     AND     #40             ;(assumed in same byte as P!)
C9B3: 09 80                                     ORA     #80             ;set presence bit
C9B5: 91 48                                     STA     @MAPTABLADR,Y
```

```
C9B7: 68                    PLA
C9B8: 48                    PHA
C9B9: 69 02                 ADC     #MTSTART        ;index to next field
C9BB: A8                    TAY
C9BC: BD 0060               LDA     6000,X          ;get starting block#
C9BF: 91 48                 STA     @MAPTABLADR,Y
C9C1: C8                    INY
C9C2: BD 0160               LDA     6000+1,X
C9C5: 91 48                 STA     @MAPTABLADR,Y
C9C7: 68                    PLA
C9C8: 48                    PHA
C9C9: 69 04                 ADC     #MTLENG         ;index to next field
C9CB: A8                    TAY
C9CC: BD 0260               LDA     6000+2,X        ;get length
C9CF: 91 48                 STA     @MAPTABLADR,Y
C9D1: C8                    INY
C9D2: BD 0360               LDA     6000+3,X
C9D5: 91 48                 STA     @MAPTABLADR,Y
C9D7: 68                    PLA
C9D8: 69 00                 ADC     #MTDRIVE
C9DA: A8                    TAY
C9DB: 68                    PLA                     ;get drive #
C9DC: 48                    PHA
C9DD: 91 48                 STA     @MAPTABLADR,Y
C9DF:
C9DF:                       ;put driver addr in BIOS and save old addr in def mt table
C9DF: A0 00                 LDY     #MAINENTRY
C9E1: B1 4E                 LDA     @INITINFOADR,Y  ;get our driver addr from the init table
C9E3: 85 04                 STA     MAPTEMP3        ;MAPTEMP3:=our driver address
C9E5: 38                    SEC
C9E6: E9 01                 SBC     #1
C9E8: 85 00                 STA     MAPTEMP1        ;MAPTEMP1:=driver address - 1
C9EA: C8                    INY
C9EB: B1 4E                 LDA     @INITINFOADR,Y
C9ED: 85 05                 STA     MAPTEMP3+1
C9EF: E9 00                 SBC     #0
C9F1: 85 01                 STA     MAPTEMP1+1
C9F3:
C9F3: BC 0460               LDY     6000+4,X;get DEFAULT_UNIT field
C9F6: C0 15                 CPY     #21.    ;system disk unit (1-20)?
C9F8: B0**                  BCS     *10     ;branch if a user unit
C9FA: 88                    DEY             ;adjust for table index
C9FB: 98                    TYA
C9FC: 0A                    ASL     A
C9FD: A8                    TAY
C9FE: B1 EA                 LDA     @0EA,Y  ;get disknum table entry for this unit
CA00: 18                    CLC
CA01: 69 01                 ADC     #1      ;normalize to true driver address
CA03: 9D 0660               STA     6000+6,X;save in table
CA06: A5 00                 LDA     MAPTEMP1;get addr of ourself (-1 for disknum tbl offset)
CA08: 91 EA                 STA     @0EA,Y  ;save in disknum table
CA0A: C8                    INY
CA0B: B1 EA                 LDA     @0EA,Y
CA0D: 69 00                 ADC     #0
CA0F: 9D 0760               STA     6000+6+1,X
```

```
CA12: A5 01                                     LDA     MAPTEMP1+1
CA14: 91 EA                                     STA     @0EA,Y
CA16: C0 07                                     CPY     #7      ;system unit (4)?
CA18: D0**                                      BNE     $9      ;if not, branch to end of loop
CA1A: BD 0660                                   LDA     6000+6,X;else get old driver addr again
CA1D: A0 04                                     LDY     #ORIG4ADR
CA1F: 91 4E                                     STA     @1NITINFOADR,Y      ;and save for later
CA21: BD 0760                                   LDA     6000+6+1,X
CA24: C8                                        INY
CA25: 91 4E                                     STA     @INITINFOADR,Y
CA27: B0**                                      BCS     $9      ;always branch
CA29:                           $10
CA29: 98                                        TYA             ;adjust for usertable index
CA2A: 49 80                                     EOR     #80
CA2C: 85 00                                     STA     MAPTEMP1
CA2E: 0A                                        ASL     A
CA2F: 38                                        SEC             ;to add +1
CA30: 65 00                                     ADC     MAPTEMP1
CA32: A8                                        TAY
CA33: B1 E8                                     LDA     @0E8,Y  ;get usernum table entry
CA35: 9D 0660                                   STA     6000+6,X;save in table
CA38: A5 04                                     LDA     MAPTEMP3;our driver address, calc'd above
CA3A: 91 E8                                     STA     @0E8,Y  ;save addr of ourself in BIOS
CA3C: C8                                        INY
CA3D: B1 E8                                     LDA     @0E8,Y
CA3F: 9D 0760                                   STA     6000+6+1,X
CA42: A5 05                                     LDA     MAPTEMP3+1
CA44: 91 E8                                     STA     @0E8,Y
CA46:
CA46: 4C 87C9                   $9              JMP     $8
CA49:
```

```
CA49!                                   .PAGE
CA49!                           ;---------------------------------------------------------------------------
CA49!                           ;
CA49!                           ;  Profile driver -- entry point from Pascal after mapping.  Parameters
CA49!                           ;                    same as in call from BIOS (Xreg gives command)
CA49!                           ;
CA49!                           ;---------------------------------------------------------------------------
CA49!
CA49! E0 04                     PROFILE         CPX     #4      ;branch, based on operation
CA4B! F0**                                      BEQ     $10     ;status
CA4D! E0 02                                     CPX     #2      ;init
CA4F! F0**                                      BEQ     PROINIT
CA51! 90**                                      BCC     RWSU
CA53!
CA53!                                           ;bad request #
CA53! A2 03                     $10             LDX     #XREQCODE;set IORESULT value
CA55! 60                                        RTS
CA56!
```

```
CA56:                                   .PAGE
CA56:
CA56:                           ;-------------------------------------------------------------------------------
CA56:                           ;
CA56:                           ;  Profile driver -- initialization routines (called from master init
CA56:                           ;                     routine in ATTACH driver)
CA56:                           ;
CA56:                           ;-------------------------------------------------------------------------------
CA56:
CA56:                           PROINIT         ;initialization of a single Profile.
CA56:                                           ;No initialization necessary here since Profile gets
CA56:                                           ;inited directly by ATTACH driver
CA56: A2 00                                     LDX     #0
CA58: 60                                        RTS
CA59:
CA59:                           HASCARD         ;clear parity on card. Check to make sure it's present and online
CA59: A6 43                                     LDX     SLOTX
CA5B: A4 0C                                     LDY     SLOTNUM
CA5D: 9D 83C0                                   STA     CLR_PARITY,X    ;Clear any previous parity errors.
CA60: BD 82C0                                   LDA     BUSY,X          ;get present, busy indicators
CA63: AA                                        TAX
CA64: 29 01                                     AND     #1              ;bit 0 off? (drive present & connected
CA66: D0**                                      BNE     $50             ;                           to card)
CA68: A9 01                                     LDA     #1
CA6A: 05 0D                                     ORA     BUSYALL         ;BUSYALL bit 0: some drive is present in
CA6C: 85 0D                                     STA     BUSYALL         ;some slot.
CA6E: 8A                                        TXA
CA6F: 30**                                      BMI     $42             ;bit 7 on? (ready & online)
CA71: B1 4A                     $40             LDA     @SLOTTAB,Y      ;ever been online? (maybe just busy now)
CA73: 10**                                      BPL     $45             ;branch if not
CA75:
CA75: A9 80                     $42             LDA     #080
CA77: 05 0D                                     ORA     BUSYALL         ;mark that some unit was online
CA79: 85 0D                                     STA     BUSYALL
CA7B: A9 80                                     LDA     #80
CA7D: 11 4A                                     ORA     @SLOTTAB,Y      ;mark this unit has been online
CA7F: 91 4A                                     STA     @SLOTTAB,Y
CA81: 60                        $45             RTS
CA82:
CA82: B1 4A                     $50             LDA     @SLOTTAB,Y
CA84: 29 7F                                     AND     #07F            ;drive no longer present
CA86: 91 4A                                     STA     @SLOTTAB,Y
CA88: 60                                        RTS
CA89:
CA89:                           INITSLOT        ;initialize a single slot
CA89: A0 02                                     LDY     #INTDSABL
CA8B: B1 3E                                     LDA     (INDRCN),Y
CA8D: 20 ****                   DI_EXIT         JSR     SETUPREAD
CA90: 20 ****                                   JSR     SETUPWRITE
CA93: 20 ****                                   JSR     SETUPREAD
CA96: 60                                        RTS
CA97:
CA97:                           SLOTVARS        ;compute SLOTNUM, SLOTX from INDRCN.
CA97: A5 3F                                     LDA     INDRCN+1
```

```
CA99: 29 0F                     AND     #0F
CA9B: 85 0C                     STA     SLOTNUM
CA9D: 0A                        ASL     A
CA9E: 0A                        ASL     A
CA9F: 0A                        ASL     A
CAA0: 0A                        ASL     A               ;make CN --> NO
CAA1: 85 43                     STA     SLOTX
CAA3: 60                        RTS
```

```
CAA4:                                   .PAGE
CAA4:
CAA4:                           ;-----------------------------------------------
CAA4:                           ;
CAA4:                           ; Header routine for reads and writes
CAA4:                           ; Entered with local return addr and Pascal params on stack
CAA4:                           ; and XREG = 0 for read, = 1 for write
CAA4:                           ;
CAA4:                           ;-----------------------------------------------
CAA4:
CAA4:                           RWSU                                    ;THIS ROUTINE GETS THE PARMS FOR
CAA4:                                                                   ;READS AND WRITES
CAA4: 86 42                                     STX     Z8CMD
CAA6: 68                                        PLA
CAA7: 85 0A                                     STA     MAPRET          ;get local return addr
CAA9: 68                                        PLA
CAAA: 85 0B                                     STA     MAPRET+1
CAAC: 68                                        PLA
CAAD: 85 46                                     STA     BLOCKLO
CAAF: 68                                        PLA
CAB0: 85 47                                     STA     BLOCKHI
CAB2: 68                                        PLA
CAB3: 85 3C                                     STA     BCLO            ;BYTE COUNT LOW
CAB5: 68                                        PLA
CAB6: 8D 7806                                   STA     BCHI            ;BYTE COUNT HI
CAB9: 68                                        PLA
CABA: 85 44                                     STA     BUFADLO
CABC: 68                                        PLA
CABD: 85 45                                     STA     BUFADHI
CABF: 68                                        PLA                     ;DRIVE NO.
CAC0: 09 C0                                     ORA     #0C0
CAC2: 85 3F                                     STA     INDRCN+1
CAC4: 68                                        PLA
CAC5: 68                                        PLA                     ;CONTROL WORD
CAC6: 68                                        PLA
CAC7: A5 0B                                     LDA     MAPRET+1        ;ready to return
CAC9: 48                                        PHA
CACA: A5 0A                                     LDA     MAPRET
CACC: 48                                        PHA
CACD:
CACD: 20 97CA                                   JSR     SLOTVARS        ;set up slot-dependent values
CAD0: A4 0C                                     LDY     SLOTNUM
CAD2: B1 4A                                     LDA     @SLOTTAB,Y      ;check if card there
CAD4: 29 01                                     AND     #01
CAD6: F0**                                      BEQ     $4              ;bit 1=1 if card there, branch if not
CAD8:
CAD8: A9 00                                     LDA     #0
CADA: 85 0D                                     STA     BUSYALL
CADC: 20 59CA                                   JSR     HASCARD         ;set up online table
CADF: A4 0C                                     LDY     SLOTNUM
CAE1: B1 4A                                     LDA     @SLOTTAB,Y      ;get online indicator (bit 7)
CAE3: 30**                                      BMI     RWSUCONTINUE
CAE5:
CAE5: A5 0D                                     LDA     BUSYALL         ;check which error
```

```
CAE7: 6A                                ROR     A               ;bit 0 ON = Profile is present
CAE8: 90**                              BCC     $4              ;branch if no Profiles present
CAEA: A2 09                             LDX     #OFFLINE        ;otherwise they were all just offline
CAEC: D0**                              BNE     $5
CAEE: A2 28             $4              LDX     #XNODRIVE
CAF0: 60                $5              RTS
CAF1:
CAF1:                   RWSUCONTINUE
CAF1: A6 3C                             LDX     BCLO
CAF3: AC 7806                           LDY     BCHI
CAF6: A5 42                             LDA     Z8CMD
CAF8:
CAF8:
CAF8: D0**              RWSU1           BNE     TSTBLKNUM       ;BRANCH IF A WRITE
CAFA: 8A                                TXA
CAFB: F0**                              BEQ     $10
CAFD: C8                                INY
CAFE: 88                $10             DEY
CAFF: 98                                TYA
CB00: 4A                                LSR     A
CB01: 48                                PHA
CB02: 18                                CLC
CB03: 65 46                             ADC     BLOCKLO
CB05: 85 46                             STA     BLOCKLO
CB07: A5 47                             LDA     BLOCKHI
CB09: 69 00                             ADC     #0
CB0B: 85 47                             STA     BLOCKHI         ;WE'LL READ BLOCKS IN
CB0D:                                                           ;REVERSE ORDER, HIGHEST 1ST
CB0D: 68                                PLA
CB0E: 0A                                ASL     A
CB0F: 18                                CLC
CB10: 65 45                             ADC     BUFADHI
CB12: 85 45                             STA     BUFADHI         ;POINT TO HIGHEST BUFFER ADDRESS
CB14:
```

```
CB14!                                   .PAGE
CB14!                           ;---------------------------------------------------------------------------
CB14!                           ;
CB14!                           ;  This routine tests the validity of block number requested.
CB14!                           ;
CB14!                           ;---------------------------------------------------------------------------
CB14!
CB14! A6 43                     TSTBLKNUM       LDX     SLOTX
CB16! 9D 83C0                                   STA     CLR_PARITY,X    ;enable 9334 on card
CB19! A9 00                                     LDA     #0
CB1B! 85 3A                                     STA     SWTCH
CB1D! 85 3E                                     STA     INDRCN
CB1F! 85 3D                                     STA     ERROR
CB21! F0**                                      beq     $20             ; always taken -- skip TSTBLKNUM code
CB23! A9 FF                                     LDA     #0FF
CB25! C5 47                                     CMP     BLOCKHI
CB27! 18                                        CLC
CB28! D0**                                      BNE     $10
CB2A! A5 46                                     LDA     BLOCKLO
CB2C! C9 FE                                     CMP     #0FE            ;RAM or status block?
CB2E! B0**                                      BCS     $20             ;BRANCH IF SO
CB30! A5 46                     $10             LDA     BLOCKLO         ;Make sure Block number is
CB32! C9 00                                     CMP     #DIBBLKLO       ;  within widget's range.
CB34! A5 47                                     LDA     BLOCKHI
CB36! E9 00                                     SBC     #DIBBLKHI       ;If carry clear then within range.
CB38! B0**                                      BCS     LDXBM           ;BRANCH IF INVALID BLK#
CB3A! A6 42                     $20             LDX     Z8CMD
CB3C! F0**                                      BEQ     RDBLOCK
CB3E! E6 42                                     INC     Z8CMD
CB40! 4C ****                                   JMP     DWRITE
CB43!
CB43! A2 03                             LDXBM   LDX     #BADMODE
CB45! 4C ****                                   JMP     STXERR
CB48!
```

```
CB48!                                   .PAGE
CB48! A5 44                 ARBADR      LDA     BUFADLO
CB4A! 8D F804                           STA     ORGADRLO
CB4D! A5 45                             LDA     BUFADHI
CB4F! 8D 7805                           STA     ORGADRHI
CB52! A5 3C                             LDA     BCLO
CB54! A4 3F                             LDY     INDRCN+1
CB56! 99 B805                           STA     BCLOSV,Y
CB59! AD 7806                           LDA     BCHI
CB5C! 8D 7804                           STA     BCHISV          ;BUFFER ADDRESS AND BYTE COUNT SAVED
CB5F! 60                                RTS
CB60!
CB60!                       ;-----------------------------------------------------------------------------
CB60!                       ;
CB60!                       ;  Profile driver -- read request.  Transfers "bytes/512" blocks from
CB60!                       ;  disk to buffer..buffer+bytes-1.
CB60!                       ;
CB60!                       ;-----------------------------------------------------------------------------
CB60!
CB60! CB60                  DREAD       .EQU    *
CB60! 20 48CB               RDBLOCK     JSR     ARBADR          ;move bytes parm to length var.
CB63! A9 03                             LDA     #3              ;Allow 2 retries
CB65! 0D F805                           ORA     BLK_RTRYCNT
CB68! 8D F805                           STA     BLK_RTRYCNT
CB6B! A9 00                             LDA     #0
CB6D! 8D 7807                           STA     PROSTAT
CB70! CB70                  RPAR_RETRY  .EQU    *               ;Parity error retries enter here
CB70! A9 BF                             LDA     #0BF
CB72! 25 3A                             AND     SWTCH
CB74! 85 3A                             STA     SWTCH
CB76! CB76                  RCOM_RETRY  .EQU    *               ;CMD-BSY bad response retries enter here
CB76! A9 03                             LDA     #3              ; approx 2.1 sec. wait for BSY lo
CB78! 8D F806                           STA     WAITTIME
CB7B! A9 01                             lda     #1              ; expected response byte from Profile
CB7D! 20 ****                           JSR     SNDCMD          ;do a CMD-BSY handshake
CB80! 90**                              BCC     L70
CB82!
CB82! A5 3A                 DR_ERR1     LDA     SWTCH           ;has Profile already been reset?
CB84! 30**                              BMI     $10             ;branch if so
CB86! 20 ****                           JSR     RESET_PROFILE   ;reset Profile
CB89! 38                                SEC
CB8A! 66 3A                             ROR     SWTCH           ;set Profile was reset flag
CB8C! 30E8                              BMI     RCOM_RETRY      ;try read again
CB8E! 20 ****               $10         JSR     NOTCMDLN
CB91! 38                                SEC
CB92! 4C ****               LDAXIO      JMP     LDXDE
CB95!
CB95! D0DF                  L70         BNE     RCOM_RETRY      ;bad response - try again.
CB97!
CB97!                       ;           Send command bytes
CB97!
CB97! 20 ****                           JSR     SND_CMDBYTES
CB9A! B0**                              BCS     RDRETRY         ;try again if parity error
CB9C! A9 0A                             LDA     #0A
```

```
CB9E: 8D F806                           STA     WAITTIME        ;adjust timeout wait
CBA1: A9 02                             LDA     #2
CBA3: 20 ****                           JSR     SNDCMD          ;2nd CMD-BSY for read opn
CBA6: B0DA                              BCS     DR_ERR1         ;timeout on cmd-bsy
CBA8: D0CC                              BNE     RCOM_RETRY      ;bad response - try again.
CBAA:
CBAA:                           ;       read should be complete at this point
CBAA:
CBAA: 20 ****                          JSR     GETSTAT         ;get status bytes first
CBAD: B0**                             BCS     RDRETRY         ;try again if parity error
CBAF: 10**                             BPL     $10
CBB1: E6 3B                            INC     COUNTR          ;time to reset PIPPIN?
CBB3: F0C1                             BEQ     RCOM_RETRY      ;branch if not
CBB5: D0CB                             BNE     DR_ERR1         ;reset PIPPIN
CBB7: A5 44                 $10        lda     bufadlo
CBB9: 25 45                            and     bufadhi
CBBB: 69 01                            adc     #1
CBBD: D0**                             bne     $20
CBBF: 4C ****                          jmp     getsize
CBC2: 20 ****               $20        JSR     JMPDTRNS
CBC5: B0**                             BCS     RDRETRY
CBC7: A9 00                 chkstat    LDA     #0
CBC9: 85 3D                            STA     ERROR
CBCB: B9 B803                          LDA     STATUS1,Y       ;Now check status for good read
CBCE: AA                               TAX
CBCF: 29 01                            AND     #1
CBD1: F0**                             BEQ     LDAST2          ;BRANCH IF NO ERROR
CBD3: 8A                               TXA
CBD4: 29 08                            AND     #8
CBD6: F0**                             BEQ     LDXDE           ;BRANCH IF NOT A CRC ERROR
CBD8: A2 01                            LDX     #PRTYCRC
CBDA: D0**                             BNE     STXERR
CBDC: A2 40                 LDXDE      LDX     #DEVERR
CBDE: 86 3D                 STXERR     STX     ERROR
CBE0: D0**                             BNE     EXIT
CBE2: B9 3804               LDAST2     LDA     STATUS2,Y
CBE5: 29 58                            AND     #58
CBE7: D0A9                             BNE     LDAXIO
CBE9: B9 B804                          lda     status3,Y       ; check for bad block #
CBEC: 29 40                            and     #40
CBEE: F0**                             beq     $1              ; branch if block # ok
CBF0: 4C 43CB                          jmp     ldxbm           ; tell user block # out of bounds
CBF3: A5 3C                 $1         LDA     BCLO
CBF5: 0D 7806                          ORA     BCHI
CBF8: F0**                             BEQ     EXIT            ;EXIT IF NO MORE BYTES TO READ
CBFA: A5 46                            LDA     BLOCKLO
CBFC: 29 03                            AND     #3
CBFE: D0**                             BNE     $5
CC00: 20 ****                          JSR     CCK             ;CHECK KEYBOARD EVERY FIFTH BLOCK READ
CC03: 38                    $5         SEC
CC04: AD 7805                          LDA     ORGADRHI
CC07: E9 02                            SBC     #2
CC09: 85 45                            STA     BUFADHI         ;SET BUFFER ADDRESS FOR NEXT BLOCK
CC0B: C6 46                            DEC     BLOCKLO         ;NEXT BLOCK TO READ
CC0D: A4 46                            LDY     BLOCKLO
```

```
CC0F: C8                              INY
CC10: D0**                            BNE     $10
CC12: C6 47                           DEC     BLOCKHI
CC14: 4C 60CB           $10           JMP     RDBLOCK         ;GO READ NEXT (PREVIOUS) BLOCK
CC17: 20 ****           RDRETRY       JSR     RSTORADR        ;restore 'ORGADR' and try again
CC1A: 90**                            BCC     $10
CC1C: 4C 70CB                         JMP     RPAR_RETRY
CC1F: 4C 82CB           $10           JMP     DR_ERR1
CC22:
CC22:
CC22: A9 03             FINISHIO      LDA     #03             ; approx 2.1 sec wait
CC24: 8D F806                         STA     WAITTIME
CC27: A9 01                           lda     #1              ; expected response from profile
CC29: 20 ****                         JSR     SNDCMD          ; ANY PENDING INFO TO DISK.
CC2C: A9 FF                           LDA     #0FF
CC2E: 85 42                           STA     Z8CMD
CC30: 20 ****                         JSR     SND_CMDBYTES    ;SEND PROFILE AN INVLID COMMAND
CC33: 20 ****                         JSR     SETCMDLN
CC36: 20 ****                         JSR     SETCMDLN
CC39: 20 ****                         JSR     NTCMDLN1        ;LOWER COMMAND
CC3C: BD 83C0                         LDA     CLR_PARITY,X    ;deactivate card
CC3F: 60                              RTS
CC40:
CC40: 20 22CC           EXIT          JSR     FINISHIO
CC43: A6 3D                           LDX     ERROR
CC45: 60                              RTS
CC46:
CC46: AD F804           RSTORADR      LDA     ORGADRLO        ;Reset addresses
CC49: 85 44                           STA     BUFADLO
CC4B: AD 7805                         LDA     ORGADRHI
CC4E: 85 45                           STA     BUFADHI
CC50: A4 3F                           LDY     INDRCN+1
CC52: B9 B805                         LDA     BCLOSV,Y
CC55: 85 3C                           STA     BCLO
CC57: AD 7804                         LDA     BCHISV
CC5A: 8D 7806                         STA     BCHI
CC5D: 18                              CLC
CC5E: AD F805                         LDA     BLK_RTRYCNT
CC61: 10**                            BPL     $10
CC63: 38                              SEC
CC64: 6A                $10           ROR     A
CC65: 29 83                           AND     #83
CC67: 8D F805                         STA     BLK_RTRYCNT
CC6A: 60                              RTS
CC6B:
CC6B:                   CCK           ;Console type-ahead check
CC6B:                                 ;From here transfer control to CONCK-caller in ATTACH routine.
CC6B:                                 ;Return directly to caller
CC6B:                                 ;TRASHES zpage $0-35, screen slot 0 temps and registers!
CC6B: 6C 4C00                         JMP     @CONCK
CC6E:
```

```
CC6E:                                   .PAGE
CC6E:                           ;---------------------------------------------------------------------------
CC6E:                           ;
CC6E:                           ;  Profile driver -- write request.  Transfers "bytes/512" blocks
CC6E:                           ;  from buffer to block..block+(bytes/512).
CC6E:                           ;  Error status on return from subroutines is the same as for DREAD.
CC6E:                           ;
CC6E:                           ;---------------------------------------------------------------------------
CC6E:
CC6E: A5 3C                     DWRITE          LDA     BCLO
CC70: F0**                                      BEQ     $10
CC72: EE 7806                                   INC     BCHI
CC75: AD 7806                   $10             LDA     BCHI
CC78: 4A                                        LSR     A
CC79: 90**                                      BCC     $20
CC7B: EE 7806                                   INC     BCHI            ;BYTE COUNT MOD 512
CC7E: A9 00                     $20             LDA     #0
CC80: 85 3C                                     STA     BCLO
CC82: A9 00                     WRBLOCK         LDA     #0
CC84: 8D 7807                                   STA     PROSTAT
CC87: 20 48CB                                   JSR     ARBADR          ;move bytes parm to length var.
CC8A: A9 03                                     LDA     #3
CC8C: 0D F805                                   ORA     BLK_RTRYCNT
CC8F: 8D F805                                   STA     BLK_RTRYCNT
CC92:
CC92: A9 BF                     WPAR_RETRY      LDA     #0BF
CC94: 25 3A                                     AND     SWTCH
CC96: 85 3A                                     STA     SWTCH
CC98:
CC98:                           ;CMD-BSY retries enter here
CC98:
CC98:                           WCOM_RETRY
CC98: A9 03                                     LDA     #3              ; approx 2.1 sec wait time
CC9A: 8D F806                                   STA     WAITTIME
CC9D: A9 01                                     lda     #1              ; expected response from profile
CC9F: 20 ****                                   JSR     SNDCMD          ;1st cmd-bsy handshake
CCA2: B0**                                      BCS     DW_ERR1         ;cmd-bsy timeout error
CCA4: D0F2                                      BNE     WCOM_RETRY      ;wrong response - try again
CCA6: 20 ****                                   JSR     SND_CMDBYTES    ;send write command string
CCA9: B0**                                      BCS     $12             ;retry if parity error
CCAB:
CCAB:                           ;Now set up to send write data to widget
CCAB:
CCAB: A9 04                                     LDA     #4
CCAD: 20 ****                                   JSR     SNDCMD          ;2nd cmd-bsy handshake
CCB0: B0**                                      BCS     DW_ERR1         ;timeout on cmd-bsy
CCB2: D0E4                                      BNE     WCOM_RETRY      ;retry the handshake if taken
CCB4: 20 ****                                   JSR     JMPDTRNS         ;now transfer data to widget
CCB7: A9 00                                     LDA     #0
CCB9: 85 3D                                     STA     ERROR
CCBB: B0**                      $12             BCS     WRRETRY         ;retry on parity error
CCBD:
CCBD:                           ;now get status from write - 3rd cmd-bsy handshake
CCBD:
```

```
CCBD: A9 0A                     $15             LDA     #0A
CCBF: 8D F806                                   STA     WAITTIME        ;adjust timeout value
CCC2: A9 06                     $20             LDA     #6
CCC4: 20 ****                                   JSR     SNDCMD
CCC7: B0**                                      BCS     DW_ERR1         ;timeout error - cmd-bsy
CCC9: F0**                                      BEQ     WCONT           ;continue if good return status
CCCB: 20 46CC                                   JSR     RSTORADR        ;restore 'ORGADR' since already
CCCE:                                                                   ;wrote bytes to z8
CCCE: 4C 98CC                                   JMP     WCOM_RETRY      ;and retry communication
CCD1:
CCD1: A5 3A                     DW_ERR1         LDA     SWTCH           ;has Profile been reset?
CCD3: 30**                                      BMI     $10             ;branch if so
CCD5: 20 ****                                   JSR     RESET_PROFILE
CCD8: 38                                        SEC
CCD9: 46 3A                                     LSR     SWTCH           ;set Profile was reset flag
CCDB: 30BB                                      BMI     WCOM_RETRY
CCDD: 20 ****                   $10             JSR     NOTCMDLN
CCE0: 4C DCCB                   LDAXIOERR       JMP     LDXDE
CCE3:
CCE3:
CCE3: 20 ****                   WCONT           JSR     GETSTAT
CCE6: B0**                                      BCS     WRRETRY         ;parity error - try again
CCE8: 10**                                      BPL     $10
CCEA: E6 3B                                     INC     COUNTR          ;time to reset PROFILE?
CCEC: F0**                                      BEQ     WRRETRY         ;branch if not
CCEE: D0E1                                      BNE     DW_ERR1
CCF0: 29 41                     $10             AND     #41
CCF2: D0EC                                      BNE     LDAXIOERR
CCF4: B9 3804                                   LDA     STATUS2,Y
CCF7: 29 48                                     AND     #48
CCF9: D0E5                                      BNE     LDAXIOERR       ;if pippin couldnt read its status
CCFB: CE 7806                                   DEC     BCHI
CCFE: CE 7806                                   DEC     BCHI
CD01: D0**                                      BNE     $20
CD03: 4C 40CC                                   JMP     EXIT            ;IF NORE MORE BYTES TO READ
CD06: 20 6BCC                   $20             JSR     CCK             ;CHECK KEYBOARD FOR INPUT
CD09: E6 46                                     INC     BLOCKLO
CD0B: D0**                                      BNE     $30
CD0D: E6 47                                     INC     BLOCKHI
CD0F: 4C 82CC                   $30             JMP     WRBLOCK
CD12:
CD12: 20 46CC                   WRRETRY         JSR     RSTORADR        ;restore 'ORGADR' and try again
CD15: 90BA                                      BCC     DW_ERR1
CD17: 4C 92CC                                   JMP     WPAR_RETRY
CD1A:
CD1A: CD1A                      getsize         .equ    *
CD1A: A6 43                                     ldx     slotx
CD1C: A0 13                                     ldy     #13
CD1E: BD 81C0                   $1              lda     rd_port,x
CD21: 88                                        dey
CD22: D0FA                                      bne     $1
CD24: BD 81C0                                   lda     rd_port,x
CD27: 85 47                                     sta     blockhi
CD29: BD 81C0                                   lda     rd_port,x
```

```
CD2C: 85 46                             sta     blocklo
CD2E: A9 00                             lda     #0
CD30: 8D 7806                           sta     bchi
CD33: A4 3F                             ldy     indrcn+1
CD35: 4C C7CB                           jmp     chkstat
CD38:
CD38:
CD38:                                   .INCLUDE        PROM.C.TEXT
CD38:
```

```
CD38:                                   .PAGE
CD38: A9 E8                 JMPINDRCT   LDA     #0E8
CD3A: 85 3E                             STA     INDRCN
CD3C: 6C 3E00                           JMP     @INDRCN
CD3F: A5 44                 JMPDTRNS    LDA     BUFADLO         ;save buffer address, modified in
CD41: 48                                PHA                     ;DATRANS
CD42: A6 43                             LDX     SLOTX
CD44: BD 83C0                           LDA     CLR_PARITY,X    ;turn off card's 9334
CD47: 20 38CD                           JSR     JMPINDRCT       ;jsr to slot dependent DATRANS
CD4A:                                                           ;goes to next statement when done
CD4A: 84 3C                 TRANSDNE    STY     BCLO
CD4C: 68                                PLA
CD4D: 85 44                             STA     BUFADLO
CD4F: A5 42                             LDA     Z8CMD
CD51: D0**                              BNE     L100
CD53:
CD53:                       ;CHKPARITY checks the parity error line and shifts it into Carry, so
CD53:                       ;Carry = 1 is a parity error on return to caller.
CD53:
CD53:
CD53: AD 7807               .CHKPARITY  LDA     PROSTAT
CD56: 29 FE                             AND     #0FE
CD58: 8D 7807                           STA     PROSTAT
CD5B: A4 3F                             LDY     INDRCN+1
CD5D: A6 43                             LDX     SLOTX
CD5F: BD 82C0                           LDA     BUSY,X          ;get parity error - on bit 6
CD62: 9D 83C0                           STA     CLR_PARITY,X    ;clear it for next transfer
CD65: 0A                                ASL     A
CD66: 0A                                ASL     A               ;shift it into carry
CD67: 90**                              BCC     $10
CD69: A9 01                             LDA     #1
CD6B: 0D 7807                           ORA     PROSTAT
CD6E: 8D 7807                           STA     PROSTAT
CD71: 19 8804                           ORA     STATUS3,Y
CD74: 99 8804                           STA     STATUS3,Y       ;set parity error bit
CD77: 60                    $10         RTS
CD78:
CD78: A9 00                 L100        LDA     #0
CD7A: 9D 80C0                           STA     WR_PORT,X       ;this code appends a 3 byte block # and
CD7D:                       ;                                    complement to the data sent
CD7D: A5 47                             LDA     BLOCKHI
CD7F: 9D 80C0                           STA     WR_PORT,X
CD82: A5 46                             LDA     BLOCKLO
CD84: 9D 80C0                           STA     WR_PORT,X
CD87: 48                                PHA
CD88: A9 FF                             LDA     #0FF
CD8A: 9D 80C0                           STA     WR_PORT,X
CD8D: 45 47                             EOR     BLOCKHI
CD8F: 9D 80C0                           STA     WR_PORT,X
CD92: 68                                PLA
CD93: 49 FF                             EOR     #0FF
CD95: 9D 80C0                           STA     WR_PORT,X
CD98: 20 53CD                           JSR     CHKPARITY       ;Test for parity error in transfer.
CD9B: 08                    L999        PHP
```

```
CD9C: 78                                SEI
CD9D: 20 ****                           JSR     SETUPREAD       ;restore read state
CDA0: 28                                PLP
CDA1: 60                                RTS
CDA2:
```

```
CDA2:                                   .PAGE
CDA2:
CDA2:
CDA2:                           ;-------------------------------------------------------------------------------
CDA2:                           ;
CDA2:                           ;  The following are various routines for handling the communications
CDA2:                           ;  protocol of sending commands, and receiving result codes.
CDA2:                           ;
CDA2:                           ;-------------------------------------------------------------------------------
CDA2:
CDA2:
CDA2:                           ;-------------------------------------------------------------------------------
CDA2:                           ;
CDA2:                           ;  SNDCMD performs a CMD-BSY handshake with the Z8 and checks for a
CDA2:                           ;  correct response.  If the Z8 responds with an incorrect code, a
CDA2:                           ;  'no go' code is sent by the Apple and the handshake is retried
CDA2:                           ;  up to 2 times.  On return, Carry=1 means a handshake timeout or
CDA2:                           ;  three retries attempted.  A non-zero return means an incorrect
CDA2:                           ;  response from the Z8 that may be retried.
CDA2:                           ;
CDA2:                           ;-------------------------------------------------------------------------------
CDA2:
CDA2: 48                        SNDCMD          PHA                     ;save expected response
CDA3: 20 ****                                   JSR     WAITBSYLO
CDA6: B0**                                      BCS     SENDERR         ;error exit if BSY isn't low
CDA8: A0 05                                     LDY     #SETRD
CDAA: B1 3E                                     LDA     (INDRCN),Y      ;set crw hi
CDAC: A0 07                                     LDY     #RWHI
CDAE: B1 3E                                     LDA     (INDRCN),Y      ;set datarw hi
CDB0: A0 04                                     LDY     #SETCMD
CDB2: B1 3E                                     LDA     (INDRCN),Y
CDB4:                           WAITBSYHI
CDB4: 18                        ALOOPINIT       CLC
CDB5: A0 00                                     LDY     #0              ;set .7 sec timeout
CDB7: A6 43                                     LDX     SLOTX
CDB9: BD 82C0                   ALOOP           LDA     BUSY,X
CDBC: 10**                                      BPL     SC1             ;done if taken
CDBE: 88                                        DEY
CDBF: D0F8                                      BNE     ALOOP
CDC1: AD F805                                   LDA     PASCAL          ;was driver called from Pascal?
CDC4: 10**                                      BPL     $5              ;branch if not
CDC6: A5 3E                                     LDA     INDRCN
CDC8: 29 03                                     AND     #3
CDCA: D0**                                      BNE     $5
CDCC: 20 6BCC                                   JSR     CCK             ;check keyboard for input
CDCF: C6 3E                     $5              DEC     INDRCN
CDD1: D0E1                                      BNE     ALOOPINIT
CDD3: 38                                        SEC                     ;timeout
CDD4: 68                        SENDERR         PLA
CDD5: 60                                        RTS
CDD6: A9 00                     SC1             LDA     #0
CDD8: 85 3E                                     STA     INDRCN
CDDA: 68                                        PLA                     ;get expected response
CDDB: DD 81C0                                   CMP     RD_PORT,X       ;is that what the Z8 sent?
```

```
CDDE: D0**                              BNE     NCONT           ;BRANCH IF NOT
CDE0: A9 55                 CONT        LDA     #055            ;indicate good response
CDE2: 20 ****                          JSR     BSYACK
CDE5: A9 00                             LDA     #0              ;indicate good return to caller
CDE7: 60                                RTS
CDE8:
CDE8: A9 02                 NCONT       LDA     #2
CDEA: 0D 7807                           ORA     PROSTAT
CDED: 8D 7807                           STA     PROSTAT         ;set bad response bit
CDF0: A4 3F                             LDY     INDRCN+1
CDF2: 19 B804                           ORA     STATUS3,Y
CDF5: 99 B804                           STA     STATUS3,Y
CDF8: A9 AA                             LDA     #0AA            ;tell Z8 that response not OK
CDFA: 20 ****                           JSR     BSYACK          ;drop cmd, wait for bsy to go lo
CDFD: B0D5                              BCS     SENDERR         ;timeout on bsy going lo
CDFF: A5 3A                             LDA     SWTCH
CE01: 0A                                ASL     A
CE02: 10**                              BPL     NSE
CE04: 38                                SEC                     ;set error flag
CE05: 60                                RTS
CE06: 09 80                 NSE         ORA     #80
CE08: 6A                                ROR     A               ;bit 7 restored, carry clear
CE09: 85 3A                             STA     SWTCH
CE0B: 60                                RTS
CE0C:
CE0C:
CE0C: A0 01                 WAITBSYLO   LDY     #1
CE0E: AD F806                           LDA     WAITTIME
CE11: 48                                PHA
CE12: 18                    BLOOPINIT   CLC
CE13: A0 00                             LDY     #0              ;timeout = .7 sec * waittime
CE15: A6 43                             LDX     SLOTX
CE17: BD 82C0               BLOOP       LDA     BUSY,X
CE1A: 30**                              BMI     BSYLORET        ;done if taken
CE1C: 88                                DEY
CE1D: D0F8                              BNE     BLOOP
CE1F: AD F805                           LDA     PASCAL          ;was driver called from Pascal?
CE22: 10**                              BPL     $5              ;branch if not
CE24: A5 3E                             LDA     INDRCN
CE26: 29 03                             AND     #3
CE28: D0**                              BNE     $5
CE2A: 20 6BCC                           JSR     CCK             ;check for input from keyboard
CE2D: C6 3E                 $5          DEC     INDRCN
CE2F: D0E1                              BNE     BLOOPINIT
CE31: CE F806                           DEC     WAITTIME
CE34: D0DC                              BNE     BLOOPINIT
CE36: 38                                SEC                     ;timeout
CE37: A9 00                 BSYLORET    LDA     #0
CE39: 85 3E                             STA     INDRCN
CE3B: 68                                PLA
CE3C: 8D F806                           STA     WAITTIME
CE3F: 60                                RTS
CE40:
CE40:
```

```
CE40:                          ;SND_CMDBYTES sends the command string to widget.  Enter with cmd=bsy=lo.
CE40:                          ;Error return if get parity error - Carry = 1
CE40:
CE40: 20 ****                  SND_CMDBYTES    JSR     SETUPWRITE      ;get in proper state
CE43: A5 42                                    LDA     Z8CMD           ;send command string - cmd, blockhi
CE45: 9D 80C0                                  STA     WR_PORT,X       ;blocklo, retries, retry threshold
CE48: A9 00                                    LDA     #0
CE4A: A4 47                                    LDY     BLOCKHI
CE4C: C8                                       INY
CE4D: D0**                                     BNE     $10
CE4F: A9 FF                                    LDA     #0FF
CE51: 9D 80C0                  $10             STA     WR_PORT,X
CE54: A5 47                                    LDA     BLOCKHI
CE56: 9D 80C0                                  STA     WR_PORT,X
CE59: A5 46                                    LDA     BLOCKLO
CE5B: 9D 80C0                                  STA     WR_PORT,X
CE5E: A9 0A                                    LDA     #RTRYCNT
CE60: 9D 80C0                                  STA     WR_PORT,X
CE63: A9 03                                    LDA     #RTRYTHRESH
CE65: 9D 80C0                                  STA     WR_PORT,X
CE68: A0 05                                    LDY     #SETRD
CE6A: B1 3E                                    LDA     (INDRCN),Y      ;set crw hi
CE6C: A0 07                                    LDY     #RWHI
CE6E: B1 3E                                    LDA     (INDRCN),Y      ;set datarw hi
CE70: 4C 53CD                                  JMP     CHKPARITY       ;check for parity error
```

```
CE73:                                   .PAGE
CE73:                           ;GETSTAT retrieves the status bytes from widget.
CE73: A6 43                     GETSTAT         LDX     SLOTX           ;get slot #
CE75: A4 3F                                     LDY     INDRCN+1
CE77: BD 81C0                                   LDA     RD_PORT,X
CE7A: 99 B803                                   STA     STATUS1,Y
CE7D: BD 81C0                                   LDA     RD_PORT,X
CE80: 99 3804                                   STA     STATUS2,Y
CE83: BD 81C0                                   LDA     RD_PORT,X
CE86: 48                                        PHA
CE87: BD 81C0                                   LDA     RD_PORT,X
CE8A: 99 3805                                   STA     STATUS4,Y
CE8D: 20 53CD                                   JSR     CHKPARITY
CE90: 68                                        PLA
CE91: 0D 7807                                   ORA     PROSTAT
CE94: 99 B804                                   STA     STATUS3,Y
CE97: B9 B803                                   LDA     STATUS1,Y
CE9A: 60                                        RTS
CE9B:
CE9B:
CE9B:                           ;SETUPWRITE sets CRW and DATRW lo on the Apple // interface board
CE9B:                           ;to prepare for a write operation to Profile
CE9B:
CE9B:
CE9B: A0 01                     SETUPWRITE      LDY     #SETWRT
CE9D: B1 3E                                     LDA     (INDRCN),Y      ;set crw lo
CE9F: A0 03                     SET_WRITEDIR    LDY     #RWLO
CEA1: B1 3E                                     LDA     (INDRCN),Y      ;set datarw lo
CEA3: 60                                        RTS
CEA4:                           ;
CEA4: 9D 83C0                   CSUW            STA CLR_PARITY,X        ;turn on 9334
CEA7: 20 9BCE                                   JSR SETUPWRITE
CEAA: BD 83C0                                   LDA CLR_PARITY,X        ;turn off 9334
CEAD: 60                                        RTS                     ;return to CnOO space
```

```
CEAE!                                   .PAGE
CEAE!
CEAE!
CEAE!                           ;BSYACK completes the cmd-bsy handshake by outputting the response
CEAE!                           ;byte to widget, dropping cmd, and waiting for bsy to go lo.
CEAE!                           ;Enter with the widget response ($55 or $AA) in A.
CEAE!
CEAE!
CEAE! 9D 80C0           BSYACK  STA     WR_PORT,X       ;store response byte
CEB1! A0 03                     LDY     #RWLO
CEB3! B1 3E                     LDA     (INDRCN),Y      ;set datarw lo
CEB5! A0 00                     LDY     #NOTCMD
CEB7! B1 3E                     LDA     (INDRCN),Y
CEB9! 20 0CCE                   JSR     WAITBSYLO
CEBC!
CEBC! A0 05             SETUPREAD LDY   #SETRD
CEBE! B1 3E                     LDA     (INDRCN),Y      ;set crw hi
CEC0! A0 07                     LDY     #RWHI
CEC2! B1 3E                     LDA     (INDRCN),Y      ;set datarw hi
CEC4! 60                        RTS
CEC5!
CEC5! CEC5              SETCMDLN .EQU   *
CEC5! 20 BCCE                   JSR     SETUPREAD
CEC8! A0 04                     LDY     #SETCMD
CECA! B1 3E                     LDA     (INDRCN),Y
CECC! 60                        RTS
CECD!
CECD! 20 BCCE           NOTCMDLN JSR    SETUPREAD
CED0! A0 00             NTCMDLN1 LDY    #NOTCMD
CED2! B1 3E                     LDA     (INDRCN),Y
CED4! 60                        RTS
CED5!
CED5! A9 04             RESET_PROFILE LDA #4
CED7! 0D 7807                   ORA     PROSTAT
CEDA! 8D 7807                   STA     PROSTAT
CEDD! A4 3F                     LDY     INDRCN+1
CEDF! 19 B804                   ORA     STATUS3,Y
CEE2! 99 B804                   STA     STATUS3,Y
CEE5! A0 0C                     LDY     #RST
CEE7! B1 3E                     LDA     (INDRCN),Y
CEE9! A0 25                     LDY     #25
CEEB! 88                $10     DEY
CEEC! D0FD                      BNE     $10
CEEE! A0 08                     LDY     #CLRRST
CEF0! B1 3E                     LDA     (INDRCN),Y      ;clear reset
CEF2! 60                RTRN    RTS
CEF3!
CEF3!
CEF3!
```

```
CEF3:                                   .PAGE
CEF3:                                   .INCLUDE        BOOT.TEXT
CEF3:
CEF3:                           ;
CEF3:                           ;               PROFILE BOOT CODE AND CnOO ROM
CEF3:                           ;
CEF3:
CEF3: 00 00 00 00 00 00 00              .ORG    0CF00
CF00:                           ;
CF00: A2 20                             LDX     #20
CF02: A9 00                             LDA     #0
CF04: A2 03                             LDX     #3
CF06: A2 55                             LDX     #55             ;so monitor recognizes card
CF08:                                                           ;as boot device, but pascal doesn't.
CF08: 95 41                     $10     STA     CMD-1,X         ;status command
CF0A: 95 44                             STA     BUFADLO,X
CF0C: CA                                DEX
CF0D: D0F9                              BNE     $10
CF0F: 20 58FF                           JSR     0FF58
CF12: BA                                TSX                     ;FIND WHICH SLOT...?
CF13: BD 0001                           LDA     0100,X
CF16: 85 3F                             STA     INDRCN+1        ;SAVE CN.
CF18: 45 01                             EOR     01              ;IS THIS AUTO BOOT?
CF1A: 05 00                             ORA     00
CF1C: D0**                              BNE     SKPL1           ;SKIP STARTUP DELAY IF NOT...
CF1E: CA                        L1      DEX
CF1F: CA                                DEX
CF20: CA                                DEX
CF21: D0FB                              BNE     L1
CF23: 88                                DEY                     ;DELAY .9 second since ProFile is
CF24: D0F8                              BNE     L1              ; reset during Apple power up.
CF26: AD FFCF                   SKPL1   LDA     0CFFF           ;now we know what slot we're in
CF29: A5 3F                             LDA     INDRCN+1        ; make slot CN into unitnum NO
CF2B: 0A                                ASL     A
CF2C: 0A                                ASL     A
CF2D: 0A                                ASL     A
CF2E: 0A                                ASL     A
CF2F: 85 43                             STA     UNIT
CF31: AA                                TAX
CF32: 20 1EC8                           JSR     PD1             ;do a status call
CF35: B0**                              BCS     CANTBOOT        ;branch if error
CF37: E6 42                             INC     CMD             ;for a read
CF39: A9 08                             LDA     #8
CF3B: 85 45                             STA     BUFADHI
CF3D: 20 1EC8                           JSR     PD1             ;read block 0 into $800
CF40: B0**                              BCS     CANTBOOT        ;if error
CF42: AD 0008                           LDA     0800
CF45: C9 01                             CMP     #01             ;is the 1st byte read a CLD ?
CF47: D0**                              BNE     CANTBOOT        ;branch if not
CF49: 4C 0108                           JMP     0801            ;go execute boot code
CF4C:
CF4C: A5 3F                     CANTBOOT        LDA     INDRCN+1        ;DETERMINE IF AUTO BOOT...
CF4E: 45 01                             EOR     01
CF50: 05 00                             ORA     00
```

```
CF52: D0**                              BNE     $10
CF54: 4C BAFA                           JMP     0FABA
CF57: 4C 59FF               $10         JMP     0FF59
CF5A:
```

```
CF5A:                                .PAGE
CF5A:                          ;-------------------------------------------------------------------------------
CF5A:                          ;
CF5A:                          ;  Profile Block I/O transfer routine.  This routine will transfer 512
CF5A:                          ;  bytes OR LESS to/from users buffer from/to RAM buffer of Profile's Z8.
CF5A:                          ;-------------------------------------------------------------------------------
CF5A:
CF5A: A5 3C                DTO         LDA     BCLO
CF5C: D0**                             BNE     DT1
CF5E: CE 7806                          DEC     BCHI            ;SO COUNT WILL BE ACCURATE AFTER
CF61: E6 3D                            INC     ERROR
CF63:                                                          ;1ST DECREMENT
CF63: D0**                             BNE     JSUR
CF65: 49 FF                DT1         EOR     #0FF
CF67: A8                               TAY
CF68: C8                               INY
CF69: A5 3C                            LDA     BCLO
CF6B: 18                               CLC
CF6C: 65 44                            ADC     BUFADLO
CF6E: 85 44                            STA     BUFADLO
CF70: B0**                             BCS     JSUR
CF72: C6 45                            DEC     BUFADHI
CF74:
CF74: 98                   JSUR        TYA
CF75: 29 03                            AND     #3
CF77: F0**                             BEQ     MOVIN
CF79: C9 03                            CMP     #3
CF7B: F0**                             BEQ     MI4
CF7D: C9 02                            CMP     #2
CF7F: F0**                             BEQ     MI3
CF81: D0**                             BNE     MI2
CF83:
CF83: AD FFCF              DATRANS     LDA     0CFFF           ;turn off any other slots
CF86: A0 00                            LDY     #0
CF88: 84 3D                            STY     ERROR           ;INIT PAGE COUNTER
CF8A: 84 3E                            STY     INDRCN          ;restore z page pointer
CF8C: A5 42                            LDA     Z8CMD
CF8E: F0CA                             BEQ     DTO             ;branch if a read
CF90: 20 A4CE              JSUW        JSR     CSUW
CF93: A0 00                            LDY     #0
CF95: C6 3D                            DEC     ERROR
CF97: B1 44                MOVOUT      LDA     (BUFADLO),Y     ;GET A BYTE FROM USER'S BUFFER
CF99: 9D 80C0                          STA     WR_PORT,X       ;SEND IT TO PROFILE
CF9C: C8                               INY
CF9D: D0F8                             BNE     MOVOUT
CF9F: E6 45                            INC     BUFADHI
CFA1: E6 3D                            INC     ERROR
CFA3: F0F2                             BEQ     MOVOUT
CFA5: 60                   JTD         RTS
CFA6:
CFA6: BD 81C0              MOVIN       LDA     RD_PORT,X       ;GET A BYTE FROM PROFILE
CFA9: 91 44                            STA     (BUFADLO),Y     ;PUT IT IN USER'S BUFFER
CFAB: C8                               INY
```

```
CFAC: BD 81C0           MI2         LDA     RD_PORT,X       ;GET A BYTE FROM PROFILE
CFAF: 91 44                         STA     (BUFADLO),Y     ;PUT IT IN USER'S BUFFER
CFB1: C8                            INY
CFB2: BD 81C0           MI3         LDA     RD_PORT,X       ;GET A BYTE FROM PROFILE
CFB5: 91 44                         STA     (BUFADLO),Y     ;PUT IT IN USER'S BUFFER
CFB7: C8                            INY
CFB8: BD 81C0           MI4         LDA     RD_PORT,X       ;GET A BYTE FROM PROFILE
CFBB: 91 44                         STA     (BUFADLO),Y     ;PUT IT IN USER'S BUFFER
CFBD: C8                            INY
CFBE: D0E6                          BNE     MOVIN
CFC0: E6 3D                         INC     ERROR
CFC2: E6 45                         INC     BUFADHI
CFC4: AD 7806           $10         LDA     BCHI
CFC7: 4A                            LSR     A
CFC8: 90**                          BCC     $20
CFCA: CE 7806                       DEC     BCHI
CFCD: B0D7                          BCS     MOVIN
CFCF: A4 3C             $20         LDY     BCLO
CFD1: F0**                          BEQ     $40             ;BRANCH IF WE HAVEN'T READ WHOLE BUF
CFD3: BD 81C0           $30         LDA     RD_PORT,X
CFD6: C8                            INY
CFD7: D0FA                          BNE     $30
CFD9: E6 3D                         INC     ERROR
CFDB: A5 3D             $40         LDA     ERROR
CFDD: C9 03                         CMP     #3
CFDF: 90F2                          BCC     $30
CFE1: B0C2                          BCS     JTD
CFE3:
CFE3: 00 00 00 00 00                .ORG    0CFE8
CFE8: D099                          BNE     DATRANS         ;entry branch to DATRANS
CFEA:
CFEA: AD FFCF           PDENTRY     LDA     0CFFF           ;turn off all C800 space
CFED: A5 43                         LDA     UNIT
CFEF: AA                            TAX
CFF0: 4A                            LSR     A
CFF1: 4A                            LSR     A
CFF2: 4A                            LSR     A
CFF3: 4A                            LSR     A
CFF4: 09 C0                         ORA     #0C0
CFF6: 8D F807                       STA     7F8             ;save slot # for interrupts
CFF9: 4C 1CC8                       JMP     PRODOS          ;jump to our cards C800 space
CFFC: 0000                          .WORD   MAXBLOCK
CFFE: 47                            .BYTE   47
CFFF: EA                            .BYTE   0EA
D000:
D000:
D000:                               .END
```

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

ALOOP	LB	CDB9	ALOOPINI	LB	CDB4	ARBADR	LB	CB48	BADMODE	AB	0003	BADOLDDA	AB	0027	BADREQ	LB	C89E	BCHI	AB	0678
BCHISV	AB	0478	BCLO	AB	003C	BCLOSV	AB	05B8	BLKRTRYC	AB	05F8	BLOCKHI	AB	0047	BLOCKLO	AB	0046	BLOOP	LB	CE17
BLOOPINI	LB	CE12	BSYACK	LB	CEAE	BSYHI	LB	C877	BSYLORET	LB	CE37	BUFADHI	AB	0045	BUFADLO	AB	0044	BUSY	AB	C082
BUSYALL	AB	000D	CANTBOOT	LB	CF4C	CCK	LB	CC6B	CHKPARIT	LB	CD53	CHKSTAT	LB	CBC7	CLRPARIT	AB	C083	CLRRST	AB	0008
CMD	AB	0042	CMP16	MC	-----	CONCK	AB	004C	CONT	LB	CDE0	COPYRIGH	LB	C800	COUNTR	AB	003B	CSUW	LB	CEA4
DATRANS	LB	CF83	DEVERR	AB	0040	DEVTYPE	AB	00D1	DIBBLKHI	AB	0000	DIBBLKLO	AB	0000	DIEXIT	LB	CA8D	DOIO	LB	C89E
DREAD	LB	CB60	DRERR1	LB	CB82	DTO	LB	CF5A	DT1	LB	CF65	DUMP	LB	C89E	DWERR1	LB	CCD1	DWRITE	LB	CC6E
ERROR	AB	003D	ERRRTN	LB	C879	EXIT	LB	CC40	FINISHIO	LB	CC22	GETENTRY	LB	C8F4	GETSIZE	LB	CD1A	GETSTAT	LB	CE73
HASCARD	LB	CA59	INC16	MC	-----	INDEX	LB	C8CD	INDRCN	AB	003E	INITCNT	AB	0008	INITIALI	AB	0002	INITINFO	AB	004E
INITIT	LB	C89E	INITSLOT	LB	CA89	INTDSABL	AB	0002	INTENABL	AB	0006	JMPDTRNS	LB	CD3F	JMPINDRC	LB	CD38	JSRI	MC	----
JSUR	LB	CF74	JSUW	LB	CF90	JTD	LB	CFA5	L	AB	0001	L1	LB	CF1E	L100	LB	CD78	L70	LB	CB95
L999	LB	CD9B	LDAST2	LB	CBE2	LDAXIO	LB	CB92	LDAXIOER	LB	CCE0	LDXBM	LB	CB43	LDXDE	LB	CBDC	LOSTDVC	AB	0005
MAINENTR	AB	0000	MANUF	AB	0001	MAP	LB	C902	MAPENTRY	AB	0030	MAPPIT	LB	C89E	MAPRET	AB	000A	MAPRTN	LB	C979
MAPSTAT	LB	C8A2	MAPTABLA	AB	0048	MAPTEMP1	AB	0000	MAPTEMP2	AB	0002	MAPTEMP3	AB	0004	MAPTEMP4	AB	0006	MAPTEMP5	AB	0008
MAXBLOCK	AB	0000	MI2	LB	CFAC	MI3	LB	CFB2	MI4	LB	CFB8	MOVIN	LB	CFA6	MOVOUT	LB	CF97	MTDRIVE	AB	0000
MTLENG	AB	0004	MTSTART	AB	0002	MUNIT	AB	000E	NCONT	LB	CDE8	NOOVRRUN	LB	C964	NOTCMD	AB	0000	NOTCMDLN	LB	CECD
NOTPRESE	LB	C89E	NOTSTAT	LB	C87B	NSE	LB	CE06	NTCMDLN1	LB	CED0	OFFLINE	AB	0009	OKRTN	LB	C977	ORGADRHI	AB	0578
ORGADRLO	AB	04F8	ORIG4ADR	AB	0004	OVRRUN	LB	C960	P	AB	0001	PASCAL	AB	05F8	PASCMAIN	LB	C89E	PD1	LB	C81E
PDENTRY	LB	CFEA	PROCESSU	LB	C89E	PRODOS	LB	C81C	PROFILE	LB	CA49	PROFILED	PR	-----	PROINIT	LB	CA56	PROSTAT	AB	0778
PRTYCRC	AB	0001	PTRTABL	AB	0048	RCOMRETR	LB	CB76	RDBLOCK	LB	CB60	RDPORT	AB	C081	RDRETRY	LB	CC17	RELEASE	AB	1001
RESETPRO	LB	CED5	ROMRETUR	AB	0006	RPARRETR	LB	CB70	RST	AB	000C	RSTAT	AB	C082	RSTORADR	LB	CC46	RTN	LB	C89E
RTRN	LB	CEF2	RTRYCNT	AB	000A	RTRYTHRE	AB	0003	RWHI	AB	0007	RWLO	AB	0003	RWSU	LB	CAA4	RWSU1	LB	CAF8
RWSUCONT	LB	CAF1	SC1	LB	CDD6	SCANDFMT	LB	C97A	SENDERR	LB	CDD4	SETCMD	AB	0004	SETCMDLN	LB	CEC5	SETRD	AB	0005
SETUPREA	LB	CEBC	SETUPWRI	LB	CE9B	SETWRITE	LB	CE9F	SETWRT	AB	0001	SKPL1	LB	CF26	SLOTNUM	AB	000C	SLOTTAB	AB	004A
SLOTVARS	LB	CA97	SLOTX	AB	0043	SNDCMD	LB	CDA2	SNDCMDBY	LB	CE40	STATIT	LB	C89E	STATRTN	LB	C8CA	STATUS1	AB	03B8
STATUS2	AB	0438	STATUS3	AB	04B8	STATUS4	AB	0538	STXERR	LB	CBDE	SUBTYPE	AB	0001	SWITCH	MC	-----	SWTCH	AB	003A
TRANSDNE	LB	CD4A	TSTBLKNU	LB	CB14	UNIT	AB	0043	WAITBSYH	LB	CDB4	WAITBSYL	LB	CE0C	WAITTIME	AB	06F8	WCOMRETR	LB	CC98
WCONT	LB	CCE3	WDGTRD	AB	0000	WDGTWRT	AB	0001	WDGTWRTV	AB	0002	WPARRETR	LB	CC92	WRBLOCK	LB	CC82	WRPORT	AB	C080
WRRETRY	LB	CD12	XBADOP	AB	0026	XBLKNUM	AB	002D	XBYTECNT	AB	002C	XCTLCODE	AB	0021	XIOERROR	AB	0027	XNODRIVE	AB	0028
XNORESRC	AB	0025	XNOWRITE	AB	002B	XREQCODE	AB	0003	ZBCMD	AB	0042									

```
Current minimum space is 20878 words.

Assembly complete:      1677 lines
     0   Errors flagged on this Assembly
```