BOOT PROM LISTINGS

THE CORVUS CONCEPT

CORVUS SYSTEMS

BOOT PROM LISTINGS

THE CORVUS CONCEPT

PART NO.: 7100-03294

DOCUMENT NO.: CCC/30-00/1.1

RELEASE DATE: February, 1983

CORVUS CONCEPT™ is a trademark of Corvus Systems, Inc.

```
; CC.PROM.TEXT ---------------------------------------------------------
;
;       CC.PROM -- Corvus CONCEPT Workstation PROM
;
;       Copyright 1982 Corvus Systems, Inc.
;                      San Jose, California
;
;       All Rights Reserved
;
;       v 0.1  05-06-82  LEF  Original program
;       v 0.2  05-27-82  LEF  Add keyboard driver (kb)
;                             Add display driver (mb)
;                             Add 8" Corvus floppy disk driver (kb)
;                             Add 8" Corvus floppy disk boot
;                             Finds first local disk for booting
;                             OMNINET disk driver modifications
;       v 0.3  06-21-82  LEF  Add MACSBUG interface
;                             Add 5" floppy driver
;                             Add 5" floppy boot
;                             Finds first floppy disk for booting
;                             Local disk driver modifications
;                             OMNINET disk driver modifications
;       v 0.4  06-29-82  LEF  Add time out to OMNINET short commands
;       v 0.5  08-18-82  KB   Modified MacsBug interface
;                             Modified Apple 5" floppy driver
;       v 0.6  10-29-82  LEF  Swap BACKSPACE and \ keys
;                             Modified display driver
;
;-----------------------------------------------------------------------
;
; File: CC.PROM
; Date: 29-Oct-82
; By:   L. Franklin changes by K. Ball
;
;                              +-----+----------------+
; Stag PROM checksums.         :     :                :
;                              :  H  :  362F          :
;                              :     :                :
;                              +-----+----------------+
;                              :     :                :
;                              :  L  :  F310          :
;                              :     :                :
;                              +-----+----------------+
;
```

```
; CC.PROM.TEXT -----------------------------------------------------
;
;       CC.PROM -- Corvus CONCEPT Workstation PROM
;
;       Copyright 1982 Corvus Systems, Inc.
;                      San Jose, California
;
;       All Rights Reserved
;
;       v 0.1  05-06-82  LEF  Original program
;       v 0.2  05-27-82  LEF  Add keyboard driver (kb)
;                             Add display driver (mb)
;                             Add 8" Corvus floppy disk driver (kb)
;                             Add 8" Corvus floppy disk boot
;                             Finds first local disk for booting
;                             OMNINET disk driver modifications
;       v 0.3  06-21-82  LEF  Add MACSBUG interface
;                             Add 5" floppy driver
;                             Add 5" floppy boot
;                             Finds first floppy disk for booting
;                             Local disk driver modifications
;                             OMNINET disk driver modifications
;       v 0.4  06-29-82  LEF  Add time out to OMNINET short commands
;       v 0.5  08-18-82  KB   Modified MacsBug interface
;                             Modified Apple 5" floppy driver
;       v 0.6  10-29-82  LEF  Swap BACKSPACE and \ keys
;                             Modified display driver
;
;-------------------------------------------------------------------
;
; File: CC.PROM
; Date. 29-Oct-82
; By:   L. Franklin changes by K. Ball
;
;                         +-----+----------------+
; Stag PROM checksums.    :     :                :
;                         :  H  :  362F          :
;                         :     :                :
;                         +-----+----------------+
;                         :     :                :
;                         :  L  :  F310          :
;                         :     :                :
;                         +-----+----------------+
;


```

```
                          include 'CC PROM.EQ'      ,PROM equates
                ,
                , File. CC.PROM.EQ.TEXT
                , Date. 28-Oct-82
                ,

00000000        PROMvers equ    0                  ,Current PROM version number
                ,ROMvers equ    15                 ,Temporary PROM version number ("?")
00000006        PROMlevl equ    6                  ,Current PROM level number
                                                   ,
00000000        RAMbase  equ    $00000             ,Base address of low RAM
00001000        RAMlen   equ    $1000              ,Length of low RAM (4k bytes)
00000300        RAMkbbuf equ    RAMbase+$300       ,Start of keyboard buffer
00000100        RAMkblen equ    $100               ,Length of keyboard buffer
00000400        RAMmxbug equ    RAMbase+$400       ,Start of MACSBUG RAM
00000700        RAMwksta equ    RAMbase+$700       ,Start of workstation RAM
00001000        RAMend   equ    RAMbase+RAMLen     ,End address + 1 of low RAM
                                                   ,
00010000        ROMbase  equ    $10000             ,Base address of workstation PROM
00002000        ROMlen   equ    $2000              ,Length of workstation PROM
00012000        ROMend   equ    ROMbase+ROMlen     ,End address + 1 of workstation PROM
                                                   ,
00020000        MXBbase  equ    $20000             ,Base address of MACSBUG (if present)
00002000        MXBlen   equ    $2000              ,Length of MACSBUG
00022000        MXBend   equ    MXBbase+MXBlen     ,End address + 1 of MACSBUG
00020004        MXBinit  equ    MXBbase+4          ,Address of MacsBug init vector        0.5
00020008        MXBentry equ    MXBbase+8          ,Address of MacsBug entry vector       0.5
                                                   ,
00030000        IOPbase  equ    $30000             ,Base address of I/O page
00030F00        VIAbase  equ    IOPbase+$0F00      ,Base address of VIA registers
                                                   ,
00080000        DSPbase  equ    $80000             ,Base address of display buffer
0000E000        DSPlen   equ    $0E000             ,Length of display buffer
0008E000        DSPend   equ    DSPbase+DSPlen     ,End address + 1 of display buffer
                                                   ,
0008E000        USRbase  equ    DSPend             ,Base address of user RAM

                ,
                , Corvus CONCEPT Workstation interrupt vector definition
                ,
00000064        IVlvl1   equ    $64                ,level 1 interrupt vector (SLOTS)
00000068        IVlvl2   equ    $68                ,level 2 interrupt vector (DC1)
0000006C        IVlvl3   equ    $6C                ,level 3 interrupt vector (OMNINET)
00000070        IVlvl4   equ    $70                ,level 4 interrupt vector (DC0)
00000074        IVlvl5   equ    $74                ,level 5 interrupt vector (TIMER)
00000078        IVlvl6   equ    $78                ,level 6 interrupt vector (KYDB)
0000007C        IVlvl7   equ    $7C                ,level 7 interrupt vector

```

```
                 ;
                 ; Corvus CONCEPT Workstation static RAM address definition
                 ;
00000700         CPbtslot equ    RAMwksta+$000    ;(700-700) boot slot number
00000701         CPbtsrvr equ    RAMwksta+$001    ;(701-701) boot server number
                                                  ;(702-705)
00000706         CPosslot equ    RAMwksta+$006    ;(706-706) OS volume slot number
00000707         CPossrvr equ    RAMwksta+$007    ;(707-707) OS volume server number
00000708         CPosdrv  equ    RAMwksta+$008    ;(708-708) OS volume drive number
00000709         CPosblk  equ    RAMwksta+$009    ;(709-70B) OS volume block number
                                                  ;(70C-70C)
0000070D         CPtprnbr equ    RAMwksta+$00D    ;(70D-70D) OMNINET transporter number
0000070E         CPdiskRC equ    RAMwksta+$00E    ;(70E-70E) disk controller return code
0000070F         CPomniRC equ    RAMwksta+$00F    ;(70F-70F) OMNINET return code
                                                  ;
                                                  ;(710-713)
00000714         CPblkio  equ    RAMwksta+$014    ;(714-717) boot disk blk i/o subr pointer
00000718         CPdskio  equ    RAMwksta+$018    ;(718-71B) boot disk i/o subr pointer
                                                  ;
00000720         CPuserID equ    RAMwksta+$020    ;(720-721) user ID
                                                  ;(722-725)
00000726         CPusernm equ    RAMwksta+$026    ;(726-72F) user name (10 bytes)
                                                  ;
00000730         CPfinlv  equ    RAMwksta+$030    ;(730-733) floppy interleave table pointer
00000734         CPfdvsz  equ    RAMwksta+$034    ;(734-735) floppy device size (blocks)
00000736         CPfbps   equ    RAMwksta+$036    ;(736-737) floppy bytes per sector
00000738         CPfspt   equ    RAMwksta+$038    ;(738-738) floppy sectors per track
00000739         CPftps   equ    RAMwksta+$039    ;(739-739) floppy tracks per side
0000073A         CPfspd   equ    RAMwksta+$03A    ;(73A-73A) floppy sides per disk
0000073B         CPfofst  equ    RAMwksta+$03B    ;(73B-73B) floppy first track offset
0000073C         CPftyp   equ    RAMwksta+$03C    ;(73C-73C) floppy type
                                                  ;
00000740         CPwndrcd equ    RAMwksta+$040    ;(740-763) system window record (36 bytes)
00000764         CPscnofs equ    RAMwksta+$064    ;(764-765) bytes per display scan line
00000766         CPdspflg equ    RAMwksta+$066    ;(766-766) display flags
                                                  ;
00000771         CPsl1typ equ    RAMwksta+$071    ;(771-771) slot device type for slot 1
00000772         CPsl2typ equ    RAMwksta+$072    ;(772-772) slot device type for slot 2
00000773         CPsl3typ equ    RAMwksta+$073    ;(773-773) slot device type for slot 3
00000774         CPsl4typ equ    RAMwksta+$074    ;(774-774) slot device type for slot 4
00000775         CPsl5typ equ    RAMwksta+$075    ;(775-775) slot device type for slot 5
                                                  ;
00000880         CPomnram equ    RAMwksta+$180    ;(880-88F) static RAM for OMNINET
00000900         CPsl1ram equ    RAMwksta+$200    ;(900-9FF) static RAM for slot 1 device
00000A00         CPsl2ram equ    RAMwksta+$300    ;(A00-AFF) static RAM for slot 2 device
00000B00         CPsl3ram equ    RAMwksta+$400    ;(B00-BFF) static RAM for slot 3 device
00000C00         CPsl4ram equ    RAMwksta+$500    ;(C00-CFF) static RAM for slot 4 device
```

```
00000D00     CPiobuf  equ    RAMwksta+$600   ;(D00-EFF) I/O buffer (512 bytes)
00000F00     CPstack  equ    RAMwksta+$800   ;(F00-FFF) initial system stack
00000F00     CPextcrt equ    RAMwksta+$800   ;(F00-F00) external CRT flag
00000F01     CPsysst  equ    RAMwksta+$801   ;(F01-F01) system initialization status
00000FFC     CPistack equ    RAMwksta+$8FC   ;(FFC-FFC) initial system stack pointer
                                             ;
0008DFD0     CPomnibf equ    DSPend-$30      ;OMNINET driver buffer (48 bytes)
                                             ;    (OMNINET can't access below $80000)

             ;
             ; Corvus CONCEPT Workstation PROM address vector definitions
             ;
00010004     CPsysrst equ    ROMbase+$004    ;(10004) system restart pointer
00010008     CPuniqid equ    ROMbase+$008    ;(10008) unique workstation ID
0001000C     CPromvrs equ    ROMbase+$00C    ;(1000C) PROM version number
0001000D     CPromlvl equ    ROMbase+$00D    ;(1000D) PROM level number
0001000E     CPcksum  equ    ROMbase+$00E    ;(1000E) PROM checksum
                                             ;
00010010     CPobootj equ    ROMbase+$010    ;(10010) jump to OMNINET disk boot subr
00010012     CPoboot  equ    ROMbase+$012    ;(10012) OMNINET disk boot subr pointer
00010016     CPoblkio equ    ROMbase+$016    ;(10016) OMNINET disk blk i/o subr pointer
0001001A     CPodskio equ    ROMbase+$01A    ;(1001A) OMNINET disk i/o subr pointer
                                             ;
00010020     CPlbootj equ    ROMbase+$020    ;(10020) jump to local disk boot subr
00010022     CPlboot  equ    ROMbase+$022    ;(10022) local disk boot subr pointer
00010026     CPlblkio equ    ROMbase+$026    ;(10026) local disk blk i/o subr pointer
0001002A     CPldskio equ    ROMbase+$02A    ;(1002A) local disk i/o subr pointer
                                             ;
00010030     CPfbootj equ    ROMbase+$030    ;(10030) jump to floppy disk boot subr
00010032     CPfboot  equ    ROMbase+$032    ;(10032) floppy boot subr pointer
00010036     CPfblkio equ    ROMbase+$036    ;(10036) Corvus floppy blk i/o subr pointer
0001003A     CPfsctio equ    ROMbase+$03A    ;(1003A) Corvus floppy sector i/o subr pointer
0001003E     CPfinit  equ    ROMbase+$03E    ;(1003E) Corvus floppy initialization
00010042     CPablkio equ    ROMbase+$042    ;(10042) Apple floppy blk i/o subr pointer
00010046     CPasctio equ    ROMbase+$046    ;(10046) Apple floppy sector i/o subr pointer
0001004A     CPainit  equ    ROMbase+$04A    ;(1004A) Apple floppy initialization
                                             ;
00010050     CPkbinit equ    ROMbase+$050    ;(10050) initialize (reset) keyboard driver
00010054     CPkbgetc equ    ROMbase+$054    ;(10054) get a keyboard character
                                             ;
00010060     CPdsinit equ    ROMbase+$060    ;(10060) initialize display driver
00010064     CPdsputc equ    ROMbase+$064    ;(10064) display a character
00010068     CPdsputs equ    ROMbase+$068    ;(10068) display a string
0001006C     CPdscvuc equ    ROMbase+$06C    ;(1006C) convert character to upper case

```

```
00010070      CPivec1  equ     ROMbase+$070     ;(10070) level 1 interrupt vector (SLOTS)
00010074      CPivec2  equ     ROMbase+$074     ;(10074) level 2 interrupt vector (DC1)
00010078      CPivec3  equ     ROMbase+$078     ;(10078) level 3 interrupt vector (OMNINET)
0001007C      CPivec4  equ     ROMbase+$07C     ;(1007C) level 4 interrupt vector (DC0)
00010080      CPivec5  equ     ROMbase+$080     ;(10080) level 5 interrupt vector (TIMER)
00010084      CPivec6  equ     ROMbase+$084     ;(10084) level 6 interrupt vector (KYBD)
00010088      CPivec7  equ     ROMbase+$088     ;(10088) level 7 interrupt vector

              ;
              ; Corvus CONCEPT Workstation I/O page definitions
              ;
00030F61      IObootsw equ     VIAbase+$61      ;(30F61) boot selection switches
00030F71      IObeepfq equ     VIAbase+$71      ;(30F71) beep frequency

              ;
              ; Slot device types  (set in CPsl1typ..CPsl5typ)
              ;
00000000      DTndev   equ     0                ;no device
00000001      DTlocl   equ     1                ;local disk
00000002      DTomni   equ     2                ;OMNINET disk
00000003      DTc8     equ     3                ;Corvus 8" floppy disk
00000004      DTc5     equ     4                ;Corvus 5" floppy disk
00000005      DTa5     equ     5                ;Apple 5" floppy disk

              ;
              ; Miscellaneous equates
              ;
00000000      off      equ     0                ;
00000001      on       equ     1                ;
                                                ;
00004EF9      jumpto   equ     $4EF9            ;"jmp" op code
                                                ;
00000032      DskRead  equ     $32              ;disk read command
00000033      DskWrit  equ     $33              ;disk write command

```

```
                                 ;
                                 ; Corvus CONCEPT Workstation PROM address vectors
                                 ;
0000                                      org    ROMbase           ;
0000  00000000                            data.l 0                 ;(10000) initial stack pointer
0004  000100B4+                           data.l setup             ;(10004) start of PROM code
0008  FFFFFFFF                            data.l $FFFFFFFF         ;(10008) unique workstation ID
000C  00                                  data.b PROMvers          ;(1000C) PROM version number
000D  06                                  data.b PROMlevl          ;(1000D) PROM level number
000E  FFFF                                data.w $FFFF             ;(1000E) PROM checksum
                                                                   ;
0010  4EF9                                data.w jumpto            ;(10010) jump to OMNINET disk boot subr
0012  000103FC+                           data.l SBomni            ;(10012) OMNINET disk boot subr pointer
0016  00011130+                           data.l ODblkIO           ;(10016) OMNINET disk blk i/o subr pointer
001A  0001119C+                           data.l ODdskIO           ;(1001A) OMNINET disk i/o subr pointer
001E  0000                                data.w 0                 ;
                                                                   ;
0020  4EF9                                data.w jumpto            ;(10020) jump to local disk boot subr
0022  00010408+                           data.l SBlocal           ;(10022) local disk boot subr pointer
0026  00010F8C+                           data.l LDblkIO           ;(10026) local disk blk i/o subr pointer
002A  00011036+                           data.l LDdskIO           ;(1002A) local disk i/o subr pointer
002E  0000                                data.w 0                 ;
                                                                   ;
0030  4EF9                                data.w jumpto            ;(10030) jump to floppy disk boot subr
0032  00010414+                           data.l SBflpy            ;(10032) floppy boot subr pointer
0036  0001144C+                           data.l FDblkIO           ;(10036) Corvus floppy blk i/o subr pointer
003A  00011464+                           data.l FDsecIO           ;(1003A) Corvus floppy sector i/o subr pointer
003E  000114DA+                           data.l FDinit            ;(1003E) Corvus floppy initialization
0042  00011856+                           data.l ADblkIO           ;(10042) Apple floppy blk i/o subr pointer
0046  0001187E+                           data.l ADsecIO           ;(10046) Apple floppy sector i/o subr pointer
004A  00011AF4+                           data.l ADinit            ;(1004A) Apple floppy initialization
004E  0000                                data.w 0                 ;
                                                                   ;
0050  00010722+                           data.l KBinit            ;(10050) initialize (reset) keyboard driver
0054  000107C0+                           data.l KBgetch           ;(10054) get a keyboard character
0058  00000000                            data.l 0                 ;(10058)
005C  00000000                            data.l 0                 ;(1005C)
```

```
0060  0001096E+          267*        data.l DSinit          ;(10060) initialize display driver
0064  000109D8+          268*        data.l DSputch         ;(10064) display a character
0068  000109C6+          269*        data.l DSputst         ;(10068) display a string
006C  000109B4+          270*        data.l DScvtuc         ;(1006C) convert character to upper case
                         271*                               ;
0070  00010630+          272*        data.l INTslot         ;(10070) level 1 interrupt vector (SLOTS)
0074  0001061E+          273*        data.l INTdc1          ;(10074) level 2 interrupt vector (DC1)
0078  00010616+          274*        data.l INTomni         ;(10078) level 3 interrupt vector (OMNINET)
007C  00010604+          275*        data.l INTdc0          ;(1007C) level 4 interrupt vector (DC0)
0080  000105FA+          276*        data.l INTtimr         ;(10080) level 5 interrupt vector (TIMER)
0084  000105E8+          277*        data.l INTkybd         ;(10084) level 6 interrupt vector (KYDB)
0088  000105E6+          278*        data.l INTlvl7         ;(10088) level 7 interrupt vector
008C  00000000           279*        data.l 0               ;(1008C)
                         280*
                         286*        list   1               ;
                         287*
```

```
                           ;
                           ; Initialise Corvus CONCEPT hardware
                           ;
                           ;        $30F71 - T2CL
                           ;        $30F75 - shift register
                           ;        $30F77 - aux control register
                           ;        $30F81 - CRTC pointer
                           ;        $30F83 - CRTC data
                           ;
                           ;        Note that lower bytes are odd!!!!
                           ;
00B4  46FC  2700           Setup   move.w  #$2700,sr        ;set priority to 7, nmi interrupt only
00B8  2E7C  0000  0FFC             move.l  #CPistack,a7     ;set system stack pointer
                           ;
                           ; delay for possible Apple floppy reset  *kb 8/23/82*          0.5
                           ;                                                             0.5
00BE  303C  FFFF                   MOVE.W  #$FFFF, D0       ;Must wait at least 1 second  0.5
00C2  51C8  FFFE           Setup1  DBRA    D0, Setup1       ;*                            0.5
00C6  51C8  FFFE           Setup2  DBRA    D0, Setup2       ;*                            0.5
00CA  51C8  FFFE           Setup3  DBRA    D0, Setup3       ;*                            0.5
                                                            ;
00CE  2C7C  0003  0F00             move.l  #VIAbase,a6      ;get pointer to VIA I/O locations
00D4  4278  0F00                   clr.w   CPextcrt.w       ;reset system flags (CPextcrt, CPsysst)
00D8  1D7C  0017  0007             move.b  #$17,$07(a6)     ;kybd control, 600 baud, 8 bit word
00DE  1D7C  000B  0005             move.b  #$0B,$05(a6)     ;kybd command, no parity, no interrupts
00E4  1D7C  003E  0027             move.b  #$3E,$27(a6)     ;dcom0 control, 9600 baud, 7 bit word
00EA  1D7C  00AB  0025             move.b  #$AB,$25(a6)     ;dcom0 command
00F0  1D7C  003E  0047             move.b  #$3E,$47(a6)     ;dcom1 control, 9600 baud, 7 bit word
00F6  1D7C  00AB  0045             move.b  #$AB,$45(a6)     ;dcom1 command
00FC  1D7C  0000  007F             move.b  #$00,$7F(a6)     ;VIA port A
0102  1D7C  0000  0061             move.b  #$00,$61(a6)     ;VIA port B
0108  1D7C  0080  0067             move.b  #$80,$67(a6)     ;VIA data direction A
010E  1D7C  0037  0065             move.b  #$37,$65(a6)     ;VIA data direction B
0114  1D7C  0010  0077             move.b  #$10,$77(a6)     ;free run shift register, counter
011A  1D7C  000F  0075             move.b  #$0F,$75(a6)     ;symmetrical wave shape
0120  1D7C  00A0  0071             move.b  #$A0,$71(a6)     ;fairly low initial frequency
0126  4A2E  00C1                   tst.b   $C1(a6)          ;turn off possible OMNINET interrupt
012A  102E  0001                   move.b  $01(a6),d0       ;clear keyboard data buffer
012E  102E  0021                   move.b  $21(a6),d0       ;clear dcom0 data buffer
0132  102E  0041                   move.b  $41(a6),d0       ;clear dcom1 data buffer
```

```
                        ;
                        ;  Check Corvus CONCEPT hardware  (test 1)
                        ;
                        ;       Verfiy ports making no data accesses
                        ;
0136 0C2E 003E 0027              cmpi.b #$3E,$27(a6)   ;dcom0 control
013C 6638                        bne.s  CHerr          ;
013E 0C2E 00AB 0025              cmpi.b #$AB,$25(a6)   ;dcom0 command
0144 6630                        bne.s  CHerr          ;
0146 0C2E 003E 0047              cmpi.b #$3E,$47(a6)   ;dcom1 control
014C 6628                        bne.s  CHerr          ;
014E 0C2E 00AB 0045              cmpi.b #$AB,$45(a6)   ;dcom1 command
0154 6620                        bne.s  CHerr          ;
0156 0C2E 0017 0007              cmpi.b #$17,$07(a6)   ;kybd control
015C 6618                        bne.s  CHerr          ;
015E 0C2E 000B 0005              cmpi.b #$0B,$05(a6)   ;kybd command
0164 6610                        bne.s  CHerr          ;
0166 0C2E 0080 0067              cmpi.b #$80,$67(a6)   ;VIA data direction A
016C 6608                        bne.s  CHerr          ;
016E 0C2E 0037 0065              cmpi.b #$37,$65(a6)   ;VIA data direction B
0174 6712                        beq.s  CHend          ;
                                                       ;
0176 303C FFFE          CHerr    move.w #$FFFE,d0      ;short delay before error tone
017A 51C8 FFFE          CHerr1   dbra   d0,CHerr1      ;*
017E 08F8 0000 0F01              bset   #0,CPsysst.w   ;set test 1 failed flag
0184 6100 0410                   bsr    Flash          ;*
                                                       ;
0188 303C FFFE          CHend    move.w #$FFFE,d0      ;short delay before clearing screen
018C 51C8 FFFE          CHend1   dbra   d0,CHend1      ;*
0190 207C 0008 0000              move.l #DSPbase,a0    ;get pointer to start of display screen
0196 227C 0008 E000              move.l #DSPend,a1     ;get pointer to end of display screen
019C 6100 03F0                   bsr    ZeroRam        ;clear display screen
01A0 1D7C 00FF 0075              move.b #$FF,$75(a6)   ;turn off initial tone

                        ;
                        ; RomTst1 -- Check Corvus CONCEPT PROM  (test 2)
                        ;
01A6 207C 0001 000E     RomTst1  move.l #CPcksum,a0    ;get pointer to start of PROM
01AC 227C 0001 2000              move.l #ROMend,a1     ;get pointer to end of PROM
01B2 6100 0320                   bsr    RomTst         ;check PROM
01B6 670A                        beq.s  RamTst1        ;PROM ok, go on
01B8 08F8 0001 0F01              bset   #1,CPsysst.w   ;set test 2 failed flag
01BE 6100 03D6                   bsr    Flash          ;*
```

```
                          ;
                          ; RamTst1 -- Check Corvus CONCEPT static RAM  (test 3)
                          ;
01C2 207C 0000 0700       RamTst1 move.l  #RAMwksta,a0     ;get pointer to start of RAM
01C8 6100 031C                    bsr     WalkBit          ;is RAM valid?
01CC 660C                         bne.s   RT1err           ;no, report error
01CE 227C 0000 0F00               move.l  #CPstack,a1      ;get pointer to end of RAM
                                                           ;   (leave room for stack)
01D4 6100 0342                    bsr     March            ;is RAM valid?
01D8 670A                         beq.s   RamTst2          ;yes, go on
                                                           ;
01DA 08F8 0002 0F01       RT1err  bset    #2,CPsysst.w     ;set test 3 failed flag
01E0 6100 03B4                    bsr     Flash            ;*

                          ;
                          ; RamTst2 -- Check Corvus CONCEPT dynamic RAM  (test 4)
                          ;
01E4 207C 0008 E000       RamTst2 move.l  #USRbase,a0      ;get pointer to start of RAM
01EA 6100 02FA                    bsr     WalkBit          ;is RAM valid?
01EE 660A                         bne.s   RT2err           ;no, report error
01F0 6100 0372                    bsr     RamSize          ;get dynamic RAM size (a1 = RAM size)
01F4 6100 0322                    bsr     March            ;is RAM valid?
01F8 670A                         beq.s   MemTest          ;yes, go on
                                                           ;
01FA 08F8 0003 0F01       RT2err  bset    #3,CPsysst.w     ;set test 4 failed flag
0200 6100 0394                    bsr     Flash            ;*

                          ;
                          ; MemTest -- Check Corvus CONCEPT dynamic RAM  (test 5)
                          ;
0204 6100 033E            MemTest bsr     IncTest          ;test user dynamic RAM
0208 670A                         beq.s   MemClr           ;no error, clear memory
020A 08F8 0004 0F01               bset    #4,CPsysst.w     ;set test 5 failed flag
0210 6100 0384                    bsr     Flash            ;*

```

```
                         ;
                         ; MemClr -- Clear memory
                         ;
0214 207C 0000 0400      MemClr  move.l  #RAMmxbug,a0    ;get pointer to start of RAM
021A 227C 0000 0EFE              move.l  #CPstack-2,a1   ;get pointer to end of RAM
                                                         ;  (leave room for stack)
0220 6100 036C                   bsr     ZeroRam         ;zero RAM
                                                         ;
0224 207C 0008 0000              move.l  #DSPbase,a0     ;get pointer to start of RAM
022A 6100 0338                   bsr     RamSize         ;get dynamic RAM size (a1 = RAM size)
022E 6100 035E                   bsr     ZeroRam         ;zero RAM

                         ;
                         ; SetMB -- Set MACSBUG RAM
                         ;
0232 41F9 0002 0004      SetMB   lea     MXBinit.L,a0    ;is debug PROM present?              0.5
0238 43F9 0002 000C              lea     MXBbase+$C.L,a1 ;* init vector should = base + $C   0.5
023E B3D0                        cmpa.l  (a0),a1         ;*                                   0.5
0240 6602                        bne.s   SetIntV         ;no, go on
0242 4E90                        jsr     (a0)            ;initialize MACSBUG

                         ;
                         ; SetIntV -- Set up interrupt vectors
                         ;
0244 207C 0001 0070      SetIntV move.l  #CPivec1,a0     ;get pointer to interrupt vector table
024A 227C 0000 0064              move.l  #IVlvl1,a1      ;get pointer to interrupt vectors
0250 7006                        moveq   #6,d0           ;get number of vectors to move
0252 22D8                SUI1    move.l  (a0)+,(a1)+     ;move pointers to interrupt vectors
0254 51C8 FFFC                   dbra    d0,SUI1         ;*
                                                         ;
0258 6100 04C8                   bsr     KBinit          ;initialize keyboard
025C 6100 0710                   bsr     DSinit          ;initialize display
                                                         ;
0260 41FA 03F8+                  lea     msg1,a0         ;mesg - "Corvus CONCEPT Initialization"
0264 6100 0760                   bsr     DSputst         ;output message
0268 41FA FE26+                  lea     msgcpy,a0       ;mesg - copyright notice
026C 6100 0758                   bsr     DSputst         ;output message
0270 41FA 040E+                  lea     msg2,a0         ;mesg - carriage returns
0274 6100 0750                   bsr     DSputst         ;output message
```

```
                         ;
                         ; SlotID -- Examine slots for known devices
                         ;
0278 2A7C 0000 0771      SlotID  movea.l #CPslltyp,a5    ;get pointer to slot types table
027E 227C 0003 0200              movea.l #$30200,a1      ;get pointer to slot 1 interface PROM
0284 7C01                        moveq   #1,d6           ;get index for slot 1
                                                         ;
0286 1229 0001           SlotID1 move.b  01(a1),d1       ;get interface prom code (ID)
028A E189                        lsl.l   #8,d1           ;*
028C 1229 0003                   move.b  03(a1),d1       ;*
0290 E189                        lsl.l   #8,d1           ;*
0292 1229 0005                   move.b  05(a1),d1       ;*
0296 E189                        lsl.l   #8,d1           ;*
0298 1229 0007                   move.b  07(a1),d1       ;*
029C 1429 0009                   move.b  09(a1),d2       ;get interface prom code (ID)
02A0 E18A                        lsl.l   #8,d2           ;*
02A2 1429 000B                   move.b  11(a1),d2       ;*
02A6 E18A                        lsl.l   #8,d2           ;*
02A8 1429 000D                   move.b  13(a1),d2       ;*
02AC E18A                        lsl.l   #8,d2           ;*
02AE 1429 000F                   move.b  15(a1),d2       ;*
02B2 4A39 0003 9FFF              tst.b   $39FFF.L        ;disable interface RAM
                                                         ;
02B8 B2BC A920 A900              cmp.l   #$A920A900,d1   ;is this a local disk?
02BE 6614                        bne.s   SlotID2         ;no, check next device
02C0 B4BC A903 A93C              cmp.l   #$A903A93C,d2   ;is this a local disk?
02C6 660C                        bne.s   SlotID2         ;no, check next device
02C8 6100 0DB8                   bsr     LDsync          ;sync with local disk
02CC 6D34                        blt.s   SlotID8         ;bypass slot if disk did not respond
02CE 1BBC 0001 60FF              move.b  #DTlocl,-1(a5,d6);set device type
                                                         ;
02D4 B2BC A220 A000      SlotID2 cmp.l   #$A220A000,d1   ;is this an Apple floppy?
02DA 660E                        bne.s   SlotID3         ;no, check next device
02DC B4BC A203 863C              cmp.l   #$A203863C,d2   ;*
02E2 6606                        bne.s   SlotID3         ;no, check next device
02E4 1BBC 0005 60FF              move.b  #DTa5,-1(a5,d6) ;set device type
                                                         ;
02EA B2BC 434F 5256      SlotID3 cmp.l   #'CORV',d1      ;is this a Corvus floppy?
02F0 661A                        bne.s   SlotID9         ;no, check next device
02F2 B4BC 5553 3031              cmp.l   #'US01',d2      ;*
02F8 6612                        bne.s   SlotID9         ;no, check next device
02FA 1BBC 0003 60FF              move.b  #DTc8,-1(a5,d6) ;set device type
0300 600A                        bra.s   SlotID9         ;check next slot
                                                         ;
0302 08F8 0005 0F01      SlotID8 bset    #5,CPsysst.w    ;set test 6 failed flag
0308 6100 028C                   bsr     Flash           ;*
                                                         ;
030C D2FC 0200           SlotID9 adda.w  #$200,a1        ;update interface PROM pointer
0310 5246                        addq    #1,d6           ;update slot number
0312 BC7C 0004                   cmp.w   #4,d6           ;have we looked at all slots?
0316 6F00 FF6E                   ble     SlotID1         ;no, check next slot
```

```
031A  7020                506*          moveq   #InitOp,d0       ,get OMNINET Transporter number
031C  6100  0DC8          507*          bsr     ODcomnd          ;*
0320  6D38                508*          blt.s   SlotIDb          ,if error, go on
0322  11C7  070D          509*          move.b  d7,CPtprnbr.w    ;save OMNINET Transporter number
                          510*
0326  7002                511*          moveq   #EchoOp,d0       ,is OMNINET Transporter number in use?
0328  1207                512*          move.b  d7,d1            ;*
032A  6100  0DBA          513*          bsr     ODcomnd          ;*
032E  0C07  00C0          514*          cmpi.b  #Echoed,d7       ;*
0332  660C                515*          bne.s   SlotIDa          ,no, go on
0334  08F8  0006  0F01    516*          bset    #6,CPsysst.w     ;set test 7 failed flag
033A  6100  025A          517*          bsr     Flash            ,*
033E  601A                518*          bra.s   SlotIDb          ,bypass disk server broadcast
                          519*                                   ,
0340  207C  0008  E000    520* SlotIDa  move.l  #USRbase,a0      ;send broadcast message to disk server
0346  10BC  00FF          521*          move.b  #$FF,(a0)        ;*  in order to get disk server
034A  7401                522*          moveq   #1,d2            ;*  Transporter number
034C  7A33                523*          moveq   #$33,d5          ,*
034E  7CFF                524*          moveq   #-1,d6           ,*
0350  6100  0E3E          525*          bsr     ODdskIO          ,*
0354  7A32                526*          moveq   #$32,d5          ,*
0356  6100  0E38          527*          bsr     ODdskIO          ;*
                          528*                                   ;
035A  11C7  0701          529* SlotIDb  move.b  d7,CPbtsrvr.w    ;save boot server number
035E  6D06                530*          blt.s   RptStat          ;if error, go on
0360  1B7C  0002  0004    531*          move.b  #DTomni,4(a5)    ,set device type
                          532*
```

```
                              ,
                              , RptStat -- Report results of system initialization tests
                              ,
0366  7200                    RptStat moveq   #0,d1              ,initialize test number
0368  4A38  0F01                      tst.b   CPsysst.w          ,any system errors?
036C  6608                            bne.s   RptSt1             ,yes, report them
036E  41FA  0398+                     lea     msg32,a0           ,mesg - All system tests passed
0372  6100  0652                      bsr     DSputst            ,output message
                              , ----  bra.s   RptSt8             ,output carriage returns
                                                                 ,
0376  0338  0F01              RptSt1  btst    d1,CPsysst.w       ,did current test pass?
037A  671A                            bofi.s  RptSt2             ,yes, go on
037C  41FA  0374+                     lea     msg30,a0           ,mesg - System test
0380  6100  0644                      bsr     DSputst            ,output message
0384  1001                            move.b  d1,d0              ,get test number
0386  0600  0031                      addi.b  #$31,d0            ,*
038A  6100  064C                      bsr     DSputch            ;output test number
038E  41FA  036F+                     lea     msg31,a0           ,mesg - failed
0392  6100  0632                      bsr     DSputst            ,output message
                                                                 ,
0396  5241                    RptSt2  addq    #1,d1              .increment test number
0398  B27C  0007                      cmp.w   #7,d1              ,finished with all tests?
039C  6FD8                            ble.s   RptSt1             ,no, process next test
                                                                 ;
039E  41FA  02E0+             RptSt8  lea     msg2,a0            ,output carriage returns
03A2  6100  0622                      bsr     DSputst            ,*
03A6  1D7C  000F  0075                move.b  #$0F,$75(a6)       ,symmetrical wave shape
03AC  1D7C  00A0  0071                move.b  #$A0,$71(a6)       ,output a low pitch tone
03B2  303C  FFFE                      move.w  #$FFFE,d0          .short delay
03B6  51C8  FFFE              RptSt9  dbra    d0,RptSt9          ,*
03BA  1D7C  00FF  0075                move.b  #$FF,$75(a6)       ,turn off tone
```

```
                       ;
                       ; SelBoot -- Select boot type
                       ;
03C0 1039 0003 0F61    SelBoot move.b  IObootsw.L,d0   ;get boot selection switches
03C6 0240 00C0                 andi.w  #$C0,d0         ;*
03CA 6700 008A                 beq     SBuser          ;00 - user select
03CE 0C00 0040                 cmpi.b  #$40,d0         ;
03D2 6734                      beq.s   SBlocal         ;01 - local disk boot
03D4 0C00 0080                 cmpi.b  #$80,d0         ;
03D8 6722                      beq.s   SBomni          ;02 - OMNINET disk boot
03DA 6038                      bra.s   SBflpy          ;03 - floppy disk boot
                                                       ;
03DC                   SBdebug                         ;
03DC 41F9 0002 0004            lea     MXBinit.L,a0    ;*kb is debug PROM present?        0.5
03E2 43F9 0002 000C            lea     MXBbase+$C.L,a1 ;*kb changed test to same as in    0.5
03E8 B3D0                      cmpa.l  (a0),a1         ;*kb SetMB init prom               0.5
03EA 666A                      bne.s   SBuser          ;no, ask user for boot device
03EC 41FA 02A7+                lea     msg4,a0         ;mesg - MACSBUG I/O on DataComm 0  0.6
03F0 6100 05D4                 bsr     DSputst         ;output message
03F4 2079 0002 0008            movea.l MXBentry.L,a0   ;*kb yes, go to debugger           0.5
03FA 4ED0                      jmp     (a0)            ;*kb                               0.5
                                                       ;
03FC 41FA 02D3+        SBomni  lea     msg11,a0        ;mesg - "OMNINET disk boot"
0400 6146                      bsr.s   SBmsg           ;output message
0402 6100 0CC8                 bsr     Oboot           ;load OS boot code
0406 603E                      bra.s   SBboot          ;transfer control to boot code
                                                       ;
0408 41FA 02CF+        SBlocal lea     msg12,a0        ;mesg - "Local disk boot"
040C 613A                      bsr.s   SBmsg           ;output message
040E 6100 0AEA                 bsr     Lboot           ;load OS boot code
0412 6032                      bra.s   SBboot          ;transfer control to boot code
                                                       ;
0414 41FA 02C9+        SBflpy  lea     msg13,a0        ;mesg - "Floppy disk boot"
0418 612E                      bsr.s   SBmsg           ;output message
041A 227C 0000 0771            movea.l #CPsl1typ,a1    ;get pointer to slot 1 type
0420 7001                      moveq   #1,d0           ;get initial slot number
                                                       ;
0422 1231 00FF         SBflpy1 move.b  -1(a1,d0),d1    ;get device type
0426 0C01 0003                 cmpi.b  #DTc8,d1        ;is this a Corvus floppy disk interface?
042A 6710                      beq.s   SBflpy2         ;yes, use it for booting
042C 0C01 0005                 cmpi.b  #DTa5,d1        ;is this an Apple floppy disk interface?
0430 6710                      beq.s   SBflpy3         ;yes, use it for booting
0432 5240                      addq    #1,d0           ;update slot number
0434 B07C 0004                 cmp.w   #4,d0           ;have we looked at all slots?
0438 6FE8                      ble.s   SBflpy1         ;no, check next slot
043A 605C                      bra.s   GoToBt1         ;output error message
```

```
0430  6100  0FD0     SBflpy2 bsr      Fboot        ;load OS boot code
0440  6054                   bra.s    GoToBt       ;transfer control to boot code
                                                   ;
0442  6100  13D0     SBflpy3 bsr      Aboot        ;load OS boot code
0446  604E           SBboot  bra.s    GoToBt       ;transfer control to boot code

0448  6100  057C     SBmsg   bsr      DSputst      ;output message
044C  41FA  0298+            lea      msg19,a0     ;mesg - "disk boot"
0450  6100  0574             bsr      DSputst      ;output message
0454  4E75                   rts                   ;return

0456  41FA  025A+    SBuser  lea      msg10,a0     ;mesg - "Select boot device"
045A  6100  056A             bsr      DSputst      ;output message
045E  6100  0360             bsr      KBgetch      ;get reply
0462  6100  0550             bsr      DScvtUC      ;convert character to upper case
0466  1F00                   move.b   d0,-(sp)     ;save reply
0468  6100  056E             bsr      DSputch      ;echo reply
046C  700D                   moveq    #DScr,d0     ;output carriage return
046E  6100  0568             bsr      DSputch      ;*
0472  101F                   move.b   (sp)+,d0     ;restore reply
0474  0C00  0044             cmpi.b   #'D',d0      ;debug?
0478  6700  FF62             beq      SBdebug      ;yes, do it
047C  0C00  0046             cmpi.b   #'F',d0      ;Corvus floppy boot?
0480  6792                   beq.s    SBflpy       ;yes, do it
0482  0C00  004C             cmpi.b   #'L',d0      ;local disk boot?
0486  6700  FF80             beq      SBlocal      ;yes, do it
048A  0C00  004F             cmpi.b   #'O',d0      ;OMNINET disk boot?
048E  6700  FF6C             beq      SBomni       ;yes, do it
0492  6000  FC20             bra      Setup        ;no, start over again
```

```
                    ;
                    ; GoToBt -- Transfer control to boot code
                    ;
                    ;       Enter:  A0.L    = Boot code entry point pointer
                    ;
                    ;       Values passed in registers to the boot are:
                    ;
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D0 | low user RAM address                                          |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D1 | high user RAM address                                         |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D2 | low user RAM address  (same as D0)                            |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D3 | high user RAM address (same as D1)                            |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D4 |             0 |             0 | boot slot     | boot server   |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D5 |                                                             0 |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D6 |                                                             0 |
                    ;       +---------------+---------------+---------------+---------------+
                    ;    D7 |                                                             0 |
                    ;       +---------------+---------------+---------------+---------------+
                    ;
0496 6C0C           GoToBt  bge.s   GoToBt2         ;go on if no boot load error
                                                    ;
0498 41FA 01E9+     GoToBt1 lea     msg3,a0         ;mesg - "Boot error"
049C 6100 0528              bsr     DSputst         ;output message
04A0 6000 FFB4              bra     SBuser          ;select boot device again
                                                    ;
04A4 6100 00BE      GoToBt2 bsr     RamSize         ;get dynamic RAM size (a1 = RAM size)  0.6
04A8 B3FC 0009 0000         cmpa.l  #$90000,a1      ;are we in PROM?                       0.6
04AE 6604                   bne.s   GoToBt3         ;yes, go on                            0.6
04B0 43FA 024E              lea     CPbtslot,a1     ;set RAM size to protect code          0.6
                                                    ;                                      0.6
04B4 203C 0008 E000 GoToBt3 move.l  #USRbase,d0     ;D0 - low user RAM address             0.6
04BA 2209                   move.l  a1,d1           ;D1 - high user RAM address
04BC 2400                   move.l  d0,d2           ;D2 - low user RAM address
04BE 2601                   move.l  d1,d3           ;D3 - high user RAM address
04C0 4284                   clr.l   d4              ;D4 - 0
04C2 1838 0700              move.b  CPbtslot.w,d4   ;D4 - boot slot
04C6 E14C                   lsl.w   #8,d4           ;
04C8 1838 0701              move.b  CPbtsrvr.w,d4   ;D4 - boot slot/boot server
04CC 4285                   clr.l   d5              ;D5 - 0
04CE 4286                   clr.l   d6              ;D6 - 0
04D0 4287                   clr.l   d7              ;D7 - 0
04D2 4ED0                   jmp     (a0)            ;enter boot code
```

```
                          ;
                          ; RomTst -- Compute checksum for PROM
                          ;           (PROM checksum is included in address range)
                          ;
                          ;         Enter   A0.L    = PROM start pointer
                          ;                 A1.L    = PROM end pointer
                          ;
                          ;         Exit.   EQ      = PROM checksum valid
                          ;                 NE      = PROM checksum error
                          ;
04D4  2448                RomTst  move.l  a0,a2           ;get starting address
04D6  4240                        clr.w   d0              ;
04D8  321A                RT1     move.w  (a2)+,d1        ;
04DA  B340                        eor.w   d1,d0           ;
04DC  B5C9                        cmpa.l  a1,a2           ;
04DE  6DF8                        blt.s   RT1             ;
04E0  B07C  FFFF                  cmp.w   #$FFFF,d0       ;
04E4  4E75                        rts                     ;return

                          ;
                          ; WalkBit -- Walking ones and zeros
                          ;
                          ;         Enter.  A0.L    = RAM start pointer
                          ;
04E6  2448                WalkBit move.l  a0,a2           ;get starting address
04E8  2248                        move.l  a0,a1           ;get ending address
04EA  D3FC  0000  0010            adda.l  #$10,a1         ;*
                                                          ;
04F0  303C  FFFE          WB1     move.w  #$FFFE,d0       ;
04F4  3480                WB2     move.w  d0,(a2)         ;
04F6  B052                        cmp.w   (a2),d0         ;
04F8  661C                        bne.s   WBerr           ;
04FA  E358                        rol     #1,d0           ;
04FC  65F6                        bcs.s   WB2             ;
                                                          ;
04FE  303C  0001                  move.w  #$0001,d0       ;
0502  3480                WB3     move.w  d0,(a2)         ;
0504  B052                        cmp.w   (a2),d0         ;
0506  660E                        bne.s   WBerr           ;
0508  E340                        asl     #1,d0           ;
050A  64F6                        bcc.s   WB3             ;
                                                          ;
050C  D5FC  0000  0002            adda.l  #2,a2           ;
0512  B5C9                        cmpa.l  a1,a2           ;
0514  6DDA                        blt.s   WB1             ;
                                                          ;
0516  4E75                WBerr   rts                     ;return
```

```
                        ;
                        ; March --
                        ;
                        ;       Enter:  A0.L    = RAM start pointer
                        ;               A1.L    = RAM end pointer
                        ;
0518  2448                March   move.l  a0,a2           ;
051A  4280                        clr.l   d0              ;
                                                          ;
051C  34C0                MR1     move.w  d0,(a2)+        ;
051E  B5C9                        cmpa.l  a1,a2           ;
0520  66FA                        bne.s   MR1             ;
                                                          ;
0522  3400                        move.w  d0,d2           ;
0524  4642                        not.w   d2              ;
0526  3222                MR2     move.w  -(a2),d1        ;
0528  B240                        cmp.w   d0,d1           ;
052A  6616                        bne.s   MRerr           ;
052C  3482                        move.w  d2,(a2)         ;
052E  B5C8                        cmpa.l  a0,a2           ;
0530  66F4                        bne.s   MR2             ;
                                                          ;
0532  3002                        move.w  d2,d0           ;
0534  4642                        not.w   d2              ;
0536  3212                MR3     move.w  (a2),d1         ;
0538  B240                        cmp.w   d0,d1           ;
053A  6606                        bne.s   MRerr           ;
053C  34C2                        move.w  d2,(a2)+        ;
053E  B5C9                        cmpa.l  a1,a2           ;
0540  66F4                        bne.s   MR3             ;
                                                          ;
0542  4E75                MRerr   rts                     ;return
```

```
                        ;
                        ; IncTest --
                        ;
                        ;         Enter:  A0.L    = RAM start pointer
                        ;                 A1.L    = RAM end pointer
                        ;
0544  2448              IncTest move.l  a0,a2           ;
0546  323C  0101                move.w  #$101,d1        ;
                                                        ;
054A  34C1              IT01    move.w  d1,(a2)+        ;
054C  E359                      rol.w   #1,d1           ;
054E  B5C9                      cmpa.l  a1,a2           ;
0550  6DF8                      blt.s   IT01            ;
                                                        ;
0552  2448                      move.l  a0,a2           ;
0554  323C  0101                move.w  #$101,d1        ;
                                                        ;
0558  B25A              IT02    cmp.w   (a2)+,d1        ;
055A  6606                      bne.s   IT99            ;
055C  E359                      rol.w   #1,d1           ;
055E  B5C9                      cmpa.l  a1,a2           ;
0560  6DF6                      blt.s   IT02            ;
                                                        ;
0562  4E75              IT99    rts                     ;return
```

```
                        ;
                        ; RamSize -- Get end of user RAM pointer
                        ;
                        ;       Exit:   A1.L    = RAM end pointer
                        ;
0564 227C 0009 0000     RamSize move.l  #$90000,a1      ;
056A 0C97 0001 2000             cmpi.l  #ROMend,(sp)    ;are we in PROM?
0570 6E1A                       bgt.s   RamSiz9         ;no, return
0572 23FC 000F FFFC             move.l  #$FFFFC,$FFFFC.L;get actual RAM size
0578 000F FFFC
057C 23FC 000B FFFC             move.l  #$BFFFC,$BFFFC.L;*
0582 000B FFFC
0586 2279 000F FFFC             move.l  $FFFFC.L,a1     ;*
                                                        ;
058C 4E75               RamSiz9 rts                     ;return

                        ;
                        ; ZeroRam -- Move 0 to RAM subroutine
                        ;
                        ;       Enter:  A0.L    = RAM start pointer
                        ;               A1.L    = RAM end pointer
                        ;
058E 4298               ZeroRam clr.l   (A0)+           ;
0590 B1C9                       cmpa.l  a1,a0           ;
0592 6FFA                       ble.s   ZeroRam         ;
0594 4E75                       rts                     ;return

                        ;
                        ; Flash -- Flash display screen subroutine
                        ;
0596 48E7 80C2          Flash   movem.l a0-a1/a6/d0,-(sp);save registers
059A 227C 0008 DFD0             move.l  #DSPend-$30,a1  ;get pointer to end of display screen
05A0 2C7C 0003 0F00             move.l  #VIAbase,a6     ;get pointer to VIA I/O locations
05A6 1D7C 000F 0075             move.b  #$0F,$75(a6)    ;symmetrical wave shape
05AC 1D7C 0040 0071             move.b  #$40,$71(a6)    ;output a high pitch error tone
05B2 207C 0008 0000     FL1     move.l  #DSPbase,a0     ;get pointer to start of display screen
05B8 4658               FL2     not.w   (a0)+           ;
05BA B1C9                       cmpa.l  a1,a0           ;
05BC 6DFA                       blt.s   FL2             ;
05BE 207C 0008 0000             move.l  #DSPbase,a0     ;get pointer to start of display screen
05C4 4658               FL3     not.w   (a0)+           ;
05C6 B1C9                       cmpa.l  a1,a0           ;
05C8 6DFA                       blt.s   FL3             ;
05CA 303C FFFE                  move.w  #$FFFE,d0       ;short delay
05CE 51C8 FFFE          FL4     dbra    d0,FL4          ;*
05D2 1D7C 00FF 0075             move.b  #$FF,$75(a6)    ;turn off initial tone
05D8 303C FFFE                  move.w  #$FFFE,d0       ;short delay
05DC 51C8 FFFE          FL5     dbra    d0,FL5          ;*
05E0 4CDF 4301                  movem.l (sp)+,a0-a1/a6/d0;restore registers
05E4 4E75                       rts                     ;return

```

```
                          855* ;
                          856* ; INTlvl7  -- process level 7 interrupt (ignore interrupt)
                          857* ;
05E6  4E73                858* INTlvl7 rte                    ;return from interrupt
                          859*
                          860* ;
                          861* ; INTkybd -- process KEYBOARD interrupt (ignore interrupt)
                          862* ;
05E8  0039  0002  0003    863* INTkybd ori.b   #$02,$30F05.L   ;lvl 6 (KYBD) - turn off recv interrupt
05EE  0F05
05F0  0239  00F3  0003    864*         andi.b  #$F3,$30F05.L   ;lvl 6 (KYBD) - turn off xmit interrupt
05F6  0F05
05F8  4E73                865*         rte                    ;return from interrupt
                          866*
                          867* ;
                          868* ; INTtimr -- process TIMER interrupt (ignore interrupt)
                          869* ;
05FA  13FC  007F  0003    870* INTtimr move.b  #$7F,$30F7D.L   ;lvl 5 (TIMER) - turn off VIA interrupt
0600  0F7D
0602  4E73                871*         rte                    ;return from interrupt
                          872*
                          873* ;
                          874* ; INTdc0  -- process DATACOMM0 interrupt (ignore interrupt)
                          875* ;
0604  0039  0002  0003    876* INTdc0  ori.b   #$02,$30F25.L   ;lvl 4 (DC0) - turn off recv interrupt
060A  0F25
060C  0239  00F3  0003    877*         andi.b  #$F3,$30F25.L   ;lvl 4 (DC0) - turn off xmit interrupt
0612  0F25
0614  4E73                878*         rte                    ;return from interrupt
                          879*
                          880* ;
                          881* ; INTomni -- process OMNINET interrupt (ignore interrupt)
                          882* ;
0616  4A39  0003  0FC1    883* INTomni tst.b   $30FC1.L       ;lvl 3 (OMNINET) - reset interrupt
061C  4E73                884*         rte                    ;return from interrupt
                          885*
                          886* ;
                          887* ; INTdc1  -- process DATACOMM1 interrupt (ignore interrupt)
                          888* ;
061E  0039  0002  0003    889* INTdc1  ori.b   #$02,$30F45.L   ;lvl 2 (DC1) - turn off recv interrupt
0624  0F45
0626  0239  00F3  0003    890*         andi.b  #$F3,$30F45.L   ;lvl 2 (DC1) - turn off xmit interrupt
062C  0F45
062E  4E73                891*         rte                    ;return from interrupt
                          892*
                          ;
                          ; INTslot -- process SLOT interrupt (ignore interrupt)
                          ;
0630 48E7 8080            INTslot movem.l D0/A0,-(SP)     ;save registers
0634 41F9 0003 0F7F               lea     $30F7F.L,A0     ;get pointer to port A ORA
063A 1010                         move.b  (A0),D0         ;read port A w/o handshake
063C 0840 0007                    bchg    #7,D0           ;toggle IOX
0640 1080                         move.b  D0,(A0)         ;write new IOX
0642 4CDF 0101                    movem.l (SP)+,D0/A0     ;restore registers
0646 4E73                         rte                     ;return from interrupt
                          ;
                          ; SlotAdr -- compute slot address given slot number
                          ;
                          ;       Enter:  D6.B - Slot number
                          ;
                          ;       Exit:   A1.L - I/O port address
                          ;
     00030100             SlotPtr equ     $30100          ;address of slot 0 (non-existant)
                                                          ;
0648 2F06                 SlotAdr move.l  d6,-(sp)        ;save register
064A 4886                         ext.w   d6              ;compute disk port address for slot
064C EB4E                         lsl.w   #5,d6           ;*
064E 227C 0003 0100               move.l  #SlotPtr,a1     ;*
0654 D2C6                         adda.w  d6,a1           ;*
0656 2C1F                         move.l  (sp)+,d6        ;restore register
0658 4E75                         rts                     ;return
```

```
065A  0D 0D              922* msg1     data.b  DSCcr,DSCcr
065C  436F727675732043   923*          data.b  'Corvus CONCEPT Initialization  ('
0664  4F4E434550542049
066C  6E697469616C697A
0674  6174696F6E202028
067C  30 2E 36 29        924*          data.b  PROMvers+$30,'.',PROMlevl+$30,')'
0680  0D 0D 00           925* msg2     data.b  DSCcr,DSCcr,0
0683  426F6F7420657272   926* msg3     data.b  'Boot error ....',DSCcr,DSCcr,0
068B  6F72202E2E2E2E0D
0693  0D 00
0695  0D 0D 4D41435342   927* msg4     data.b  DSCcr,DSCcr,'MACSBUG I/O on DataComm 0',DSCcr,0
069C  554720492F4F206F
06A4  6E20446174614 36F
06AC  6D6D20300D 00
06B2  53656C6563742062   928* msg10    data.b  'Select boot device (D,F,L,O): ',0
06BA  6F6F742064657669
06C2  63652028442C462C
06CA  4C2C4F293A2000
06D1  4F4D4E494E455400   929* msg11    data.b  'OMNINET',0
06D9  4C6F63616C00       930* msg12    data.b  'Local',0
06DF  466C6F707079 00    931* msg13    data.b  'Floppy',0
06E6  206469736B20626F   932* msg19    data.b  ' disk boot',DSCcr,0
06EE  6F740D 00
06F2  53797374656D2074   933* msg30    data.b  'System test ',0
06FA  6573742000
06FF  206661696C65640D   934* msg31    data.b  ' failed',DSCcr,0
0707  00
0708  416C6C2073797374   935* msg32    data.b  'All system tests passed',DSCcr,0
0710  656D207465737473
0718  207061737365640D
0720  00
0721  00                 936*          data.b  0
                         937*
```

```
                                include 'CC.PROM.KB'    ;keyboard driver
                        ;
                        ; File: CC.PROM.KB
                        ; Date: 29-Oct-82
                        ; By:   Keith Ball
                        ;
                        ; KEYBOARD DRIVER FOR PROM (kb)

                        ;
                        ; EQUATES FOR ALL KEYBOARD SOFTWARE
                        ;
                        ; KEYBOARD DATA AREA DEFINITIONS
                        ;
00000000                KBBflgs EQU     0               ;FLAG JUST HI ORDER BYTE
00000002                KBBfrnt EQU     KBBflgs+2       ;FRONT PTR SAVE
00000006                KBBrear EQU     KBBfrnt+4       ;REAR PTR SAVE
0000000A                KBBsrsv EQU     KBBrear+4       ;STATUS REG SAVE AREA
0000000C                KBBbufr EQU     KBBsrsv+2       ;KEYBOARD BUFFER
000000F4                KBBlen  EQU     RAMkblen-KBBbufr;NMBR OF BYTES IN BUFFER
                        ;
                        ; FLAG BIT DEFINITIONS
                        ;
00000000                KBFfull EQU     0               ;BUFFER FULL FLAG
00000001                KBFemty EQU     1               ;BUFFER EMPTY FLAG
00000002                KBFclos EQU     2               ;KEY CLOSURE FLAG
00000003                KBFshft EQU     3               ;SHIFT KEY
00000004                KBFcntl EQU     4               ;CONTROL KEY
00000005                KBFlock EQU     5               ;SHIFT LOCK KEY
                        ;
                        ; MISCELLANEOUS EQUATES
                        ;
0000001F                KBmsk40 EQU     $1F             ;MASK TO CLEAR D7-D5 (CONTROL CODE)
                        ;
                        ; TABLE VALUES FOR PROCESSING CHARACTERS
                        ;
0000007F                KBCqual EQU     $7F             ;QUALIFIER VALUES > THEN THIS
000000FE                KBCshft EQU     $FE             ;TABLE VALUE FOR SHIFT
000000FD                KBCcntl EQU     $FD             ;TABLE VALUE FOR CONTROL
000000FC                KBClock EQU     $FC             ;TABLE VALUE FOR SHIFT LOCK
000000FF                KBCnoch EQU     $FF             ;TABLE VALUE FOR NO CHAR CDE
                        ;
                        ; SPECIAL ASCII CHARACTERS
                        ;
00000061                KBClca  EQU     'a'             ;LOWER CASE A
0000007A                KBClcz  EQU     'z'             ;LOWER CASE Z
0000003F                KBCqmrk EQU     '?'             ;QUESTION MARK

```

```
                          ,
                          , ADDRESSES OF KEYBOARD UART'S I/O REGISTERS
                          ,
      00030F01            KBRdata EQU     $30F01           ;DATA INPUT PORT
      00030F03            KBRstat EQU     $30F03           ,STATUS REGISTER
      00030F05            KBRcmnd EQU     $30F05           ,COMMAND REGISTER
      00030F07            KBRcntl EQU     $30F07           ;CONTROL REGISTER
                          ,
                          , COMMAND AND CONTROL REGISTER VALUES
                          ,
      00000002            KBccOff EQU     $02              ;TURN OFF UART (CMD)
      00000017            KBcc600 EQU     $17              ,600 BAUD AND 8 BIT XMIT (CTL)
      00000008            KBccBrk EQU     $08              ,XMIT A BREAK (CMD)
      00000009            KBccGo  EQU     $09              ;TURN ON INTS & UART (CMD)
                          ,
      00000700            KBdsInt EQU     $0700            ,DISABLE 68000 INTERRUPTS

                          ,
                          ; KBinit - initialize (reset) keyboard
                          ,
                          ;      REGISTER A2 IS USED AS POINTER TO COMMAND REGISTER
                          ,      REGISTER A3 IS ADDRESS OF KBRD DATA AREA
                          ,
0722  48E7  80F0          KBinit  MOVEM.L D0/A0-A3,-(SP)          ;save registers
0726  47F8  0300                  LEA     RAMkbbuf.W,A3
072A  45F9  0003  0F05            LEA     KBRcmnd.L,A2
0730  14BC  0002                  MOVE.B  #KBccOff,(A2)           ,TURN OFF KBRD
0734  41EB  0000                  LEA     KBBflgs(A3),A0          ,CLEAR INT HANDLER FLAGS
0738  4290                        CLR.L   (A0)                    ,INCLUDES QUALIFIERS
073A  08D0  0001                  BSET    #KBFemty,(A0)           ,BUFFER IS EMPTY
                                  ,
                                  , INITIALIZE FRONT & REAR POINTERS
                                  ,
073E  41EB  000C                  LEA     KBBbufr(A3),A0
0742  43EB  0002                  LEA     KBBfrnt(A3),A1
0746  22C8                        MOVE.L  A0,(A1)+
0748  2288                        MOVE.L  A0,(A1)
074A  41FA  0034+                 LEA     KBintr,A0               ;SETUP AUTOVECTOR 6
074E  21C8  0078                  MOVE.L  A0,IVlvl6.W             ;WITH ADDR OF INT HANDLER
                                  ,
                                  , TURN ON KEYBOARD UART
                                  ;
0752  1039  0003  0F03            MOVE.B  KBRstat.L,D0            ,RESET UART
0758  1039  0003  0F01            MOVE.B  KBRdata.L,D0            ,CLEAR RECEIVE
075E  13FC  0017  0003            MOVE.B  #KBcc600,KBRcntl.L      ,8 BITS, 600 BAUD XMISSION
0764  0F07
0766  14BC  0008                  MOVE.B  #KBccBrk,(A2)           ,FORCE BREAK OF KBRD
076A  303C  8235                  MOVE.W  #33333,D0               ;DELAY FOR UART TO DO BREAK
076E  51C8  FFFE          KBinit1 DBF     D0,KBinit1              ;NEED MINIMUM OF 33.3 MILLISECS
0772  14BC  0009                  MOVE.B  #KBccGo,(A2)            ,TURN ON UART & INTERRUPTS
0776  46FC  2500                  move.w  #$2500,sr               ,set priority to 6, KYBD intr only
077A  4CDF  0F01                  MOVEM.L (SP)+,D0/A0-A3          ,restore registers
077E  4E75                        RTS
```

```
                    ;
                    ; KBintr - Keyboard interrupt service routine
                    ;
                    ; BEGIN INTERRUPT SERVICE ROUTINE.  THIS IS THE ENTRY POINT.  IT'S ADDRESS
                    ; MUST BE PLACED IN AUTO VECTOR INTERRUPT 6 VECTOR BEFORE KEYBOARD INTERRUPT
                    ; IS TURNED ON.
                    ;
                    ; REGISTER USEAGE:  D0 - KEYCODE
                    ;                   D1 - CHARACTER
                    ;                   A0 - ADDRESS OF FLAG BYTE
                    ;                   A2 - BASE ADDRESS OF KBRD DATA AREA
                    ;
0780  48E7  FFFE    KBintr  MOVEM.L D0-A6,-(SP)             ;SAVE REGISTERS ON STACK
0784  45F8  0300            LEA     RAMkbbuf.W,A2           ;BASE ADDR OF KBRD DATA AREA
0788  617E                  BSR.S   KBgetky                 ;GET KEYCODE FROM UART DATA PORT
                            ;
                            ; IF BIT 7 OF KEYCODE SET THEN CLOSURE ELSE RELEASE
                            ;
078A  41EA  0000            LEA     KBBflgs(A2),A0
078E  0890  0002            BCLR    #KBFclos,(A0)           ;ASSUME RELEASE
0792  0800  0007            BTST    #7,D0                   ;KEYCODE BIT D7 CLEAR?
0796  6708                  BEQ.S   KBintr1                 ;YES
0798  08D0  0002            BSET    #KBFclos,(A0)
079C  0880  0007            BCLR    #7,D0
                            ;
                            ; GET CHARACTER CODE FOR THIS KEYCODE
                            ;
07A0  43FA  010C+   KBintr1 LEA     KBstable,A1             ;ASSUME SHIFT TABLE
07A4  0810  0003            BTST    #KBFshft,(A0)
07A8  6604                  BNE.S   KBintr2                 ;SHIFT BIT SET
07AA  43FA  0162+           LEA     KBrtable,A1             ;ELSE USE REGULAR TABLE
07AE  1231  0000    KBintr2 MOVE.B  0(A1,D0.W),D1           ;INDEX TABLE BY KEYCODE
                            ;
                            ; IF CHAR(D1) = $FF THEN IGNORE AND EXIT
                            ;
07B2  0C01  00FF            CMPI.B  #KBCnoch,D1
07B6  6702                  BEQ.S   KBintr9
07B8  615E                  BSR.S   KBproky                 ;ELSE PROCESS KEYCODE
                            ;
                            ; EXIT INTERRUPT SERVICE ROUTINE
                            ;
07BA  4CDF  7FFF    KBintr9 MOVEM.L (SP)+,D0-A6             ;RESTORE REGISTERS
07BE  4E73                  RTE                             ;EXIT INTERRUPT
```

```
                    ;
                    ; KBgetch - Get a keyboard character
                    ;
                    ; Register useage:  A0 = Front pointer
                    ;                   A1 = address of end of buffer + 1
                    ;                   A2 = updated front pointer
                    ;                   A3 = address of front pointer
                    ;                   A4 = address of flag byte
                    ;                   A5 = address of keyboard data area
                    ;                   A6 = address of Status Register save area
                    ;
                    ;        Exit.   D0.B - Next character in buffer
                    ;
07C0 48E7 00FE      KBgetch MOVEM.L A0-A6,-(SP)             ;save all address registers
07C4 4BF8 0300              LEA     RAMkbbuf.W,A5           ;keyboard data area
07C8 49ED 0000              LEA     KBBflgs(A5),A4          ;address of Flag byte
                            ;
                            ; Wait for a character in the Buffer.
                            ;
07CC 0814 0001      KBgchr1 BTST    #KBFemty,(A4)           ;while (Buffer_empty) do;
07D0 66FA                   BNE.S   KBgchr1                 ;*
                            ;
                            ; have char, check for wrap around before get char
                            ;
07D2 47ED 0002              LEA     KBBfrnt(A5),A3          ;pointer to Front save loc
07D6 2053                   MOVE.L  (A3),A0                 ;Front pointer
07D8 43ED 0100              LEA     KBBbufr+KBBlen(A5),A1   ;end of buffer + 1
07DC 2448                   MOVE.L  A0,A2                   ;
07DE 528A                   ADDQ.L  #1,A2                   ;add one to pointer to get next addr
07E0 B5C8                   CMPA.L  A0,A2                   ;Front=end of buffer + 1 ?
07E2 6604                   BNE.S   KBgchr2                 ;No
07E4 45ED 000C              LEA     KBBbufr(A5),A2          ;yes, then pointer wraps back to beginning
                                                            ;
07E8 4DED 000A      KBgchr2 LEA     KBBsrsv(A5),A6          ;
07EC 40D6                   MOVE.W  SR,(A6)                 ;
07EE 007C 0700              ORI.W   #KBdsInt,SR             ;*** disable interrupts
07F2 1010                   MOVE.B  (A0),D0                 ;get char
07F4 268A                   MOVE.L  A2,(A3)                 ;save new Front value
07F6 B5ED 0006              CMPA.L  KBBrear(A5),A2          ;if Front=Rear then
07FA 6604                   BNE.S   KBgchr3                 ;Buffer_empty := true;
07FC 08D4 0001              BSET    #KBFemty,(A4)           ;
                                                            ;
0800 46D6           KBgchr3 MOVE.W  (A6),SR                 ;*** enable interrupts
0802 4CDF 7F00              MOVEM.L (SP)+,A0-A6             ;restore callers address regs
0806 4E75                   RTS
```

```
                          1133* ;
                          1134* ; KBgetky - GET KEYCODE (IGNORES ERRORS)
                          1135* ;
                          1136* ;     EXIT : (D0) - UART DATA PORT BYTE
                          1137* ;
0808 4280                 1138* KBgetky CLR.L   D0                   ;MAKE SURE HI 3 BYTES ARE 0
                          1139*         ;
                          1140*         ; READ STATUS REGISTER TO CLEAR IRQ BIT
                          1141*         ; ALWAYS READ DATA PORT SO IF OVERRUN THEN FOR NEXT CHAR
                          1142*         ; IT WILL BE CLEARED.
                          1143*         ;
080A 1239 0003 0F03       1144*         MOVE.B  KBRstat.L,D1         ;GET STATUS OF RECEIVE
0810 1039 0003 0F01       1145*         MOVE.B  KBRdata.L,D0         ;READ UART DATA PORT
0816 4E75                 1146*         RTS
                          1147*
```

```
                         ;
                         ; KBproky - PROCESS CHARACTER OR QUALIFIER
                         ;
                         ;        Enter:  D1 = CHARACTER CODE FROM TABLE
                         ;                D0 = KEYCODE
                         ;                A0 = ADDRESS OF FLAGS
                         ;
0818  0C01  007F         KBproky CMPI.B  #KBCqual,D1          ;IS IT A QUALIFIER
081C  623C                       BHI.S   KBpro3               ;YES
                                 ;
                                 ; IGNORE REST OF KEYS IF NOT CLOSURE
                                 ;
081E  0810  0002                 BTST    #KBFclos,(A0)
0822  6738                       BEQ.S   KBpro9
                                 ;
                                 ; TEST FOR CONTROL
                                 ;
0824  0810  0004                 BTST    #KBFcntl,(A0)
0828  670C                       BEQ.S   KBpro1               ;NO,TRY SHIFT LOCK
082A  0C01  003F                 CMPI.B  #KBCqmrk,D1
082E  6306                       BLS.S   KBpro1
0830  0201  001F                 ANDI.B  #KBmsk40,D1          ;CLEAR BITS D7,D6,D5 OF CHAR
0834  601A                       BRA.S   KBpro2               ;PUT CHAR
                                 ;
                                 ; TEST FOR SHIFT LOCK
                                 ;
0836  0810  0005         KBpro1  BTST    #KBFlock,(A0)
083A  6714                       BEQ.S   KBpro2               ;KEY NOT DOWN
083C  0C01  0061                 CMPI.B  #KBClca,D1
0840  650E                       BCS.S   KBpro2               ;NOT WITHIN RANGE
0842  0C01  007A                 CMPI.B  #KBClcz,D1
0846  6208                       BHI.S   KBpro2               ;NOT WITHIN RANGE
0848  43FA  0064+                LEA     KBstable,A1
084C  1231  0000                 MOVE.B  0(A1,D0.W),D1        ;INDEX TABLE BY KEYCODE
                                 ;
                                 ; IF BUFFER NOT FULL PUT CHARACTER
                                 ;
0850  0810  0000         KBpro2  BTST    #KBFfull,(A0)
0854  6606                       BNE.S   KBpro9
0856  6106                       BSR.S   KBputch
0858  6002                       BRA.S   KBpro9
                                 ;
                                 ; PROCESS A QUALIFIER KEY
                                 ;
085A  6128               KBpro3  BSR.S   KBqual
085C  4E75               KBpro9  RTS
```

```
                 ,
                 ; KBputch - PUT ONE CHARACTER IN BUFFER
                 ;
                 ;       Enter:  D1 = BYTE TO PUT IN BUFFER
                 ;               A0 = ADDRESS OF FLAGS
                 ;               A2 = ADDRESS OF KEYBOARD DATA AREA
                 ;
                         ;
                         ; PUT CHARACTER IN CIRCULAR QUEUE AT REAR
                         ;
085E 4BEA 0006   KBputch LEA     KBBrear(A2),A5
0862 2655                MOVE.L  (A5),A3
0864 16C1                MOVE.B  D1,(A3)+               ,UPDATE POINTER ALSO
                         ;
                         ; IF REAR > ENDBUFFER THEN REAR := @BUFFER
                         ;
0866 49EA 0100           LEA     KBBbufr+KBBlen(A2),A4
086A B7CC                CMPA.L  A4,A3
086C 6604                BNE.S   KBput1                 ,NOT BEYOND BUFFER
086E 47EA 006C           LEA     KBBbufr(A2),A3
                         ,
                         ; IF FRONT = REAR THEN BUFFER FULL
                         ;
0872 B7EA 0002   KBput1  CMPA.L  KBBfrnt(A2),A3
0876 6604                BNE.S   KBput2
0878 08D0 0000           BSET    #KBFfull,(A0)
087C 2A8B        KBput2  MOVE.L  A3,(A5)                ,UPDATE REAR IN MEMORY
087E 0890 0001           BCLR    #KBFemty,(A0)          ,SHOW BUFFER NOT EMPTY
0882 4E75                RTS
```

```
                    ;
                    ; KBqual - PROCESS QUALIFIER KEYS
                    ;
                    ;         Enter: D1 = CHARACTER CODE FROM TABLE
                    ;                A0 = ADDRESS OF FLAGS
                    ;
0884 0C01 00FE      KBqual  CMPI.B  #KBCshft,D1         ;IS IT SHIFT?
0888 6604                   BNE.S   KBqual1             ;NO
088A 7403                   MOVEQ   #KBFshft,D2         ;BIT POSITION OF SHIFT
088C 6012                   BRA.S   KBqual3             ;CHANGE FLAG
                                                        ;
088E 0C01 00FD      KBqual1 CMPI.B  #KBCcntl,D1         ;IS IT CONTROL?
0892 6604                   BNE.S   KBqual2             ;NO
0894 7404                   MOVEQ   #KBFcntl,D2         ;BIT POSITION OF CONTROL
0896 6008                   BRA.S   KBqual3             ;CHANGE FLAG
                                                        ;
0898 0C01 00FC      KBqual2 CMPI.B  #KBClock,D1         ;IS IT SHIFT LOCK?
089C 660E                   BNE.S   KBqual9             ;NO,THEN IT'S GARBAGE
089E 7405                   MOVEQ   #KBFlock,D2
                            ;
                            ; IF CLOSURE THEN SET FLAG ELSE CLEAR FLAG
                            ;
08A0 0810 0002      KBqual3 BTST    #KBFclos,(A0)
08A4 6704                   BEQ.S   KBqual8
08A6 05D0                   BSET    D2,(A0)
08A8 6002                   BRA.S   KBqual9
08AA 0590           KBqual8 BCLR    D2,(A0)
08AC 4E75           KBqual9 RTS
```

```
                              ;
                              ; THE SHIFT TABLE
                              ;        TABLE IS INDEXED BY KEYCODE.  EACH BYTE REPRESENTS THE ENTRY FOR
                              ;        THE CORRESPONDING KEYCODE.
                              ;
                              ;           0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
08AE                          KBstable
                                ;        ..   3   9  ..   6   ,   -  cr  ..   1   7  ..   4   8   5   2
08AE  FF 33 39 FF 36 2C         DATA.B  $FF,$33,$39,$FF,$36,$2C,$2D,$0D,$FF,$31,$37,$FF,$34,$38,$35,$32
08B4  2D 0D FF 31 37 FF
08BA  34 38 35 32
                              ; ;         +  ..   {   :  cr   }  bs  ..   )   ?   P   _   :   ~   "  .
                              ; DATA.B  $2B,$FF,$7B,$7C,$0D,$7D,$08,$FF,$29,$3F,$50,$5F,$3A,$7E,$22,$FE
                                ;         +  ..   {  bs  cr   }   :  ..   )   ?   P   _   :   ~   "  .. ,0 6
08BE  2B FF 7B 08 0D 7D         DATA.B  $2B,$FF,$7B,$08,$0D,$7D,$7C,$FF,$29,$3F,$50,$5F,$3A,$7E,$22,$FE ,0 6
08C4  7C FF 29 3F 50 5F
08CA  3A 7E 22 FE
                                ;        ..  ..  ..  ..  ..  ..  ..  ..   $   %   R   T   F   G   V   B
08CE  FF FF FF FF FF FF         DATA.B  $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$24,$25,$52,$54,$46,$47,$56,$42
08D4  FF FF 24 25 52 54
08DA  46 47 56 42
                                ,         @   #   W   E   S   D   X   C esc   !  ..   Q   .   A  ..   Z
08DE  40 23 57 45 53 44         DATA.B  $40,$23,$57,$45,$53,$44,$58,$43,$1B,$21,$FF,$51,$FC,$41,$FE,$5A
08E4  58 43 1B 21 FF 51
08EA  FC 41 FE 5A
                                ,         ^   &   Y   U   H   J   N   M  ..  ..  ..  sp  ..   0  ..   .
08EE  5E 26 59 55 48 4A         DATA.B  $5E,$26,$59,$55,$48,$4A,$4E,$4D,$FD,$FF,$FF,$20,$FF,$30,$FF,$2E
08F4  4E 4D FD FF FF 20
08FA  FF 30 FF 2E
                                ,         *   (   I   O   K   L   <   >  ..  ..  ..  ..  ..  ..  ..  ..
08FE  2A 28 49 4F 4B 4C         DATA.B  $2A,$28,$49,$4F,$4B,$4C,$3C,$3E,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
0904  3C 3E FF FF FF FF
090A  FF FF FF FF
```

```
                        1281* ;
                        1282* ; THE REGULAR TABLE - UNSHIFTED OR LOWER CASE
                        1283* ;       TABLE IS INDEXED BY KEYCODE.  EACH BYTE REPRESENTS THE ENTRY FOR
                        1284* ;       THE CORRESPONDING KEYCODE.
                        1285* ;
                        1286* ;         0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
090E                    1287* KBrtable
                        1288*   ;       ..  3   9   ..  6   ,   -   cr  ..  1   7   ..  4   8   5   2
090E  FF 33 39 FF 36 2C 1289*   DATA.B  $FF,$33,$39,$FF,$36,$2C,$2D,$0D,$FF,$31,$37,$FF,$34,$38,$35,$32
0914  2D 0D FF 31 37 FF
091A  34 38 35 32
                        1290* ; ;       =   ..  [   \   cr  ]   bs  ..  0   /   p   -   ;   '   ;   ..
                        1291* ; DATA.B  $3D,$FF,$5B,$5C,$0D,$5D,$08,$FF,$30,$2F,$70,$2D,$3B,$60,$27,$FE
                        1292*   ;       =   ..  [   bs  cr  ]   \   ..  0   /   p   -   ;   '   ;   ..  ;0.6
091E  3D FF 5B 08 0D 5D 1293*   DATA.B  $3D,$FF,$5B,$08,$0D,$5D,$5C,$FF,$30,$2F,$70,$2D,$3B,$60,$27,$FE ;0.6
0924  5C FF 30 2F 70 2D
092A  3B 60 27 FE
                        1294*   ;       ..  ..  ..  ..  ..  ..  ..  ..  4   5   r   t   f   g   v   b
092E  FF FF FF FF FF FF 1295*   DATA.B  $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$34,$35,$72,$74,$66,$67,$76,$62
0934  FF FF 34 35 72 74
093A  66 67 76 62
                        1296*   ;       2   3   w   e   s   d   x   c  esc  1   ..  q   ..  a   ..  z
093E  32 33 77 65 73 64 1297*   DATA.B  $32,$33,$77,$65,$73,$64,$78,$63,$1B,$31,$FF,$71,$FC,$61,$FE,$7A
0944  78 63 1B 31 FF 71
094A  FC 61 FE 7A
                        1298*   ;       6   7   y   u   h   j   n   m   ..  ..  ..  sp  ..  0   ..  .
094E  36 37 79 75 68 6A 1299*   DATA.B  $36,$37,$79,$75,$68,$6A,$6E,$6D,$FD,$FF,$FF,$20,$FF,$30,$FF,$2E
0954  6E 6D FD FF FF 20
095A  FF 30 FF 2E
                        1300*   ;       8   9   i   o   k   l   ,   .   ..  ..  ..  ..  ..  ..  ..  ..
095E  38 39 69 6F 6B 6C 1301*   DATA.B  $38,$39,$69,$6F,$6B,$6C,$2C,$2E,$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
0964  2C 2E FF FF FF FF
096A  FF FF FF FF
                        1302*
```

```
                 include 'CC.PROM.DS'    ;display driver
          ;
          ; File: CC.PROM.DS
          ; Date: 29-Oct-82
          ;
          ; DISPLAY DRIVER FOR PROM (mb) 05/18/82
          ;
          ; BOTH horizontal and vertical display driver
          ; contains default window records, copies them into memory
          ; contains default character sets
          ; no CRTST code: no window functions

00000060  DSdefOf equ     96      ;default bytes per scan line
0008D55E  DShomeH equ     $8D55E  ;horizontal home location
0008D506  DShomeV equ     $8D506  ;vertical home location
00000006  DScellW equ     6       ;character cell width
0000000A  DScellY equ     10      ;character cell height
000002CF  DSmaxXH equ     719     ;120*DScellW-1
0000022F  DSmaxYH equ     559     ;56*DScellY-1
0000022D  DSmaxXV equ     557     ;93*DScellW-1
000002CF  DSmaxYV equ     719     ;72*DScellY-1
                                  ;
0000000D  DSCcr   equ     $0D     ;carriage return character
0000001B  DSCesc  equ     $1B     ;escape character
00000020  DSCblnk equ     $20     ;blank character
00000061  DSClca  equ     $61     ;lower case "a"
0000007A  DSClcz  equ     $7A     ;lower case "z"
00000020  DSCdiff equ     $20
          ;
          ; Character Set Record Equates
          ;
          ;Stblloc equ    0       ;character set data pointer (not used)
00000004  CSlpch  equ     4       ;scanlines per character
00000006  CSbpch  equ     6       ;bits per character
00000008  CSfrstch equ    8       ;first character code - ascii
0000000A  CSlastch equ    10      ;last character code - ascii
0000000C  CSmask  equ     12      ;mask used in positioning cells
00000010  CSattr1 equ     16      ;attributes
                                  ;  bit 0 = 1 - vertical orientation
00000011  CSattr2 equ     17      ;currently unused
00000012  CSdata  equ     18      ;offset of char data from char record

```

```
                          ;
                          ; Window Record Equates
                          ;
00000000                  WRcharpt equ    0       ;character set pointer
00000004                  WRhomept equ    4       ;home (upper left) pointer
00000008                  WRcuradr equ    8       ;current location pointer
0000000C                  WRhomeof equ    12      ;bit offset of home location
0000000E                  WRbasex  equ    14      ;home x value, relative to root window
00000010                  WRbasey  equ    16      ;home y value, relative to root window
00000012                  WRlngthx equ    18      ;maximum x value, relative to window (bits)
00000014                  WRlngthy equ    20      ;maximum y value, relative to window (bits)
00000016                  WRcursx  equ    22      ;current x value (bits)
00000018                  WRcursy  equ    24      ;current y value (bits)
0000001A                  WRbitofs equ    26      ;bit offset of current address
0000001C                  WRgrorgx equ    28      ;graphics - origin x (bits relative to home loc)
0000001E                  WRgrorgy equ    30      ;graphics - origin y (bits relative to home loc)
00000020                  WRattr1  equ    32      ;attributes
00000021                  WRattr2  equ    33      ;attributes
                                                  ;
00000000                  vert     equ    0       ;   1 = vertical,   0 = horizontal screen
00000001                  graphic  equ    1       ;   1 = graphics,   0 = character mode
00000002                  curson   equ    2       ;   1 = cursor on,  0 = cursor off
00000003                  invcurs  equ    3       ;   1 = inverse,    0 = underline cursor
00000004                  wrapon   equ    4       ;   1 = wrap,       0 = clip at eoln
00000005                  noscroll equ    5       ;   1 = no scroll,  0 = scroll
                                                  ;
00000022                  WRstate  equ    34      ;used for decoding escape sequences
00000023                  WRrcdlen equ    35      ;window description record length
                                                  ;
00000024                  WRlength equ    36      ;actual window record length
```

```
                          ;
                          ; DSinit - Initialize display driver
                          ;
096E  48E7  08E0          DSinit  MOVEM.L D4/A0-A2,-(SP)          ;save registers
0972  31FC  0060  0764            MOVE.W  #DSdefOf,CPscnofs.W     ;set bytes per scan line
0978  41FA  03B0+                 LEA     DSwndH,A0               ;assume horizontal orientation
097C  45FA  03D0+                 LEA     DScsetH,A2              ;*
0980  0839  0003  0003            BTST    #3,IObootsw.L           ;is display horizontal?
0984  0F61
0988  6708                        BOFF.S  DSinit1                 ;yes, go on
098A  41FA  03D4+                 LEA     DSwndV,A0               ;set vertical orientation
098E  45FA  03F4+                 LEA     DScsetV,A2              ;*
0992  227C  0000  0740    DSinit1 MOVE.L  #CPwndrcd,A1            ;get pointer to RAM window record
0998  7823                        MOVEQ   #WRlength-1,D4          ;get window record length       0 6
099A  12D8                DSinit2 MOVE.B  (A0)+,(A1)+             ;copy window record to RAM      0 6
099C  51CC  FFFC                  DBRA    D4,DSinit2              ;*
09A0  207C  0000  0740            MOVE.L  #CPwndrcd,A0            ;get RAM window record pointer
09A6  214A  0000                  MOVE.L  A2,WRcharpt(A0)         ;set character set record pointer
09AA  6100  01EE                  BSR     DScurs                  ;display cursor on screen
09AE  4CDF  0710                  MOVEM.L (SP)+,D4/A0-A2          ;restore registers
09B2  4E75                        RTS                             ;return

                          ;
                          ; DScvtUC - Convert character to upper case
                          ;
                          ;       Enter:  D0.B = ASCII character
                          ;
                          ;       Exit.   D0.B = upper case ASCII character
                          ;
09B4  0C00  0061          DScvtUC CMPI.B  #DSClca,D0              ;is character lower case?
09B8  650A                        BLO.S   DScvtU1                 ;no, return
09BA  0C00  007A                  CMPI.B  #DSClcz,D0              ;*
09BE  6204                        BHI.S   DScvtU1                 ;no, return
09C0  0400  0020                  SUBI.B  #DSCdiff,D0             ;convert character to upper case
09C4  4E75                DScvtU1 RTS                             ;return

                          ;
                          ; DSputst - Display a string
                          ;
                          ;       Enter:  A0.L - Character string pointer
                          ;                       (terminated by 0)
                          ;
09C6  48E7  8080          DSputst movem.l D0/A0,-(SP)             ;save registers
                                                                  ;
09CA  1018                DSpst1  move.b  (a0)+,d0                ;get next character
09CC  6704                        beq.s   DSpst9                  ;finished, return
09CE  6108                        bsr.s   DSputch                 ;output character
09D0  60F8                        bra.s   DSpst1                  ;get next character
                                                                  ;
09D2  4CDF  0101          DSpst9  movem.l (SP)+,D0/A0             ;restore registers
09D4  4E75                        rts                             ;return

```

```
                          ;
                          ; DSputch - Display a character
                          ;
                          ;       Enter:  D0.B - Character to output
                          ;
09D8  48E7  FFFE          DSputch MOVEM.L D0-D7/A0-A6,-(SP)     ;save registers
09DC  0240  007F                  andi.w  #$7F,d0               ;make character 7 bits
09E0  207C  0000  0740            MOVE.L  #CPwndrcd,A0          ;get RAM window record pointer
09E6  2468  0000                  MOVE.L  WRcharpt(A0),A2       ;get character set record pointer
09EA  4283                        CLR.L   D3                    ;
09EC  1628  0022                  MOVE.B  WRstate(A0),D3        ;
09F0  E34B                        LSL.W   #1,D3                 ;convert to state table index
09F2  43FA  0332+                 LEA     DSsTbl,A1             ;
09F6  3631  3000                  MOVE.W  0(A1,D3.W),D3         ;D3 = dist from DSsTbl
09FA  4EF1  3000                  JMP     0(A1,D3.W)            ;go to current state processing
                                                                ;
09FE  5228  0022          DSnxtSt ADDQ.B  #1,WRstate(A0)        ;increment for next state
0A02  6004                        BRA.S   DSexit                ;return
                                                                ;
0A04  4228  0022          DSreset CLR.B   WRstate(A0)           ;reset current state
                                                                ;
0A08  4CDF  7FFF          DSexit  MOVEM.L (SP)+,D0-D7/A0-A6     ;restore registers
0A0C  4E75                        RTS                           ;return

0A0E  B03C  001B          DSst0   CMP.B   #DSCesc,D0            ;is char ESC?
0A12  67EA                        BEQ.S   DSnxtSt               ;yes, go to next state
0A14  B06A  0008                  CMP.W   CSfrstch(A2),D0       ;ascinum < first char?
0A18  6508                        BLO.S   DSctl                 ;yes, it's a control char
0A1A  6148                        BSR.S   DSshwCh               ;display character
0A1C  6100  0102                  BSR     DSincx                ;inccurx
0A20  60E6                        BRA.S   DSexit                ;return

0A22  5140                DSctl   SUBQ.W  #8,D0                 ;commence decoding ctrl char
0A24  6BE2                        BMI.S   DSexit                ;
0A26  0C40  0005                  CMPI.W  #5,D0                 ;ascinum in [8..13]?
0A2A  62DC                        BHI.S   DSexit                ;yes, do cursor ctrl
0A2C  47FA  02DA+                 LEA     DScTbl,A3             ;A3==>jump table for ctrl chars
0A30  E348                        LSL.W   #1,D0                 ;make it word count
0A32  487A  FFD4+                 PEA     DSexit                ;ensure RTS to exit
0A36  3033  0000                  MOVE.W  0(A3,D0),D0           ;D0 is offset from DScTbl
0A3A  4EF3  0000                  JMP     0(A3,D0)              ;jump to proper routine
```

```
0A3E 4241                DSesc   CLR.W   D1                  ;initialise index reg
0A40 47FA 02D2+                  LEA     DSeTbl,A3           ;A3==> beginning of table
                                                             ;
0A44 B073 1000           DSesc1  CMP.W   0(A3,D1),D0         ;does table entry match char?
0A48 670A                        BEQ.S   DSesc2              ;yes, go on
0A4A 5841                        ADDQ.W  #4,D1               ;go to next entry
0A4C 4A73 1000                   TST.W   0(A3,D1)            ;end of table?
0A50 6AF2                        BPL.S   DSesc1              ;no, loop
0A52 60B0                        BRA.S   DSreset             ;return
                                                             ;
0A54 3001                DSesc2  MOVE.W  D1,D0               ;set D0 to table offset
0A56 5440                        ADDQ.W  #2,D0               ;
0A58 487A FFAA+                  PEA     DSreset             ;ensure RTS to reset state
0A5C 3033 0000                   MOVE.W  0(A3,D0),D0         ;D0 is offset from DSeTbl
0A60 4EF3 0000                   JMP     0(A3,D0)            ;jump to proper routine

```

```
                         ;
                         ; DSshwCh - Display character
                         ;
                         ;        Enter.  A0.L = window record pointer
                         ;                A2.L = character set record pointer
                         ;                D0.W = ASCII character
                         ;
                         ;        Note.   Character set must be in bytes, not words
                         ;
0A64  6100  FF4E           DSshwCh BSR      DScvtUC              ;convert character to upper case
0A68  B06A  0008                   CMP.W    CSfrstch(A2),D0      ;is character in character set?
0A6C  6D06                         BLT.S    DSshow1              ;no, output space
0A6E  B06A  000A                   CMP.W    CSlastch(A2),D0      ;*
0A72  6F04                         BLE.S    DSshow2              ;yes, output character
0A74  303C  0020           DSshow1 MOVE.W   #DSCblnk,D0          ;no, output space
                                                                 ;
0A78  906A  0008           DSshow2 SUB.W    CSfrstch(A2),D0      ;get relative character position
0A7C  47FA  0318+                  LEA      DScsetV+CSdata,a3    ;get pointer to character data
0A80  C0FC  0006                   MULU     #DScellW,D0          ;*
0A84  D7C0                         ADDA.L   D0,A3                ;*
0A86  2868  0008                   MOVE.L   WRcuradr(A0),A4      ;get current character address
0A8A  322A  0004                   MOVE.W   CSlpch(A2),D1        ;get number of scan lines for character
0A8E  5341                         SUBQ.W   #1,D1                ;get count for DBRA
0A90  3638  0764                   MOVE.W   CPscnofs.W,D3        ;get scan line length
0A94  3A28  001A                   MOVE.W   WRbitofs(A0),D5      ;get bit offset of character in cell
0A98  2C2A  000C                   MOVE.L   CSmask(A2),D6        ;get character mask
0A9C  0828  0000  0021             BTST     #vert,WRattr2(A0)    ;is this vertical orientation?
0AA2  671C                         BOFF.S   DSshow6              ;no, output horizontal character
                                   ;
                                   ; output vertical orientation character
                                   ;
0AA4  2006                         MOVE.L   D6,D0                ;
0AA6  4680                         NOT.L    D0                   ;D0 = inverted mask
0AA8  EABE                         ROR.L    D5,D6                ;D6 = positioned mask
0AAA  141B                 DSshow3 MOVE.B   (A3)+,D2             ;D2 = char data
0AAC  E14A                         LSL.W    #8,D2                ;
0AAE  4842                         SWAP     D2                   ;get char in high word
0AB0  C480                         AND.L    D0,D2                ;clear rest of source char
0AB2  EAAA                         LSR.L    D5,D2                ;position source char
0AB4  CD94                         AND.L    D6,(A4)              ;clear dest char area
0AB6  8594                         OR.L     D2,(A4)              ;move in character
0AB8  98C3                         SUBA.W   D3,A4                ;
0ABA  51C9  FFEE                   DBRA     D1,DSshow3           ;repeat for D1:=CSlpch-1 to 0
0ABE  605A                         BRA.S    DSshow9              ;return
```

```
                          ;
                          ; output horizontal orientation character
                          ;
0AC0  4A64          DSshow6 TST.W   -(A4)           ;A4==>long word with cell
0AC2  EBBE                  ROL.L   D5,D6           ;D6 = positioned mask
0AC4  2806                  MOVE.L  D6,D4           ;
0AC6  4684                  NOT.L   D4              ;D4 = inverted mask
0AC8  7007                  moveq   #7,d0           ;use 8 bits of character data
                                                    ;
0ACA  4282          DSshow7 clr.l   d2              ;clear current scan line of character
0ACC  4A40                  tst.w   d0              ;have we used 8 bits of character data?
0ACE  6D3C                  blt.s   DSshw76         ;yes, pad with space
0AD0  012B 0000             btst    d0,0(a3)        ;construct next horizontal character
0AD4  6704                  boff.s  DSshw71         ;*  from vertical character data
0AD6  08C2 0000             bset    #0,d2           ;*
0ADA  012B 0001     DSshw71 btst    d0,1(a3)        ;*
0ADE  6704                  boff.s  DSshw72         ;*
0AE0  08C2 0001             bset    #1,d2           ;*
0AE4  012B 0002     DSshw72 btst    d0,2(a3)        ;*
0AE8  6704                  boff.s  DSshw73         ;*
0AEA  08C2 0002             bset    #2,d2           ;*
0AEE  012B 0003     DSshw73 btst    d0,3(a3)        ;*
0AF2  6704                  boff.s  DSshw74         ;*
0AF4  08C2 0003             bset    #3,d2           ;*
0AF8  012B 0004     DSshw74 btst    d0,4(a3)        ;*
0AFC  6704                  boff.s  DSshw75         ;*
0AFE  08C2 0004             bset    #4,d2           ;*
0B02  012B 0005     DSshw75 btst    d0,5(a3)        ;*
0B06  6704                  boff.s  DSshw76         ;*
0B08  08C2 0005             bset    #5,d2           ;*
0B0C  5340          DSshw76 subq    #1,d0           ;indicate another bit used
0B0E  EBAA                  LSL.L   D5,D2           ;shift char into position
0B10  CD94                  AND.L   D6,(A4)         ;
0B12  8594                  OR.L    D2,(A4)         ;
0B14  98C3                  SUBA.W  D3,A4           ;
0B16  51C9 FFB2             DBRA    D1,DSshow7      ;
0B1A  4E75          DSshow9 RTS                     ;return
```

```
                          ;
                          ; DScrsR -- cursor right
                          ;
0B1C  6100  007E          DScrsR  BSR     DScurs0              ;remove cursor
0B20  3228  0016          DSincx  MOVE.W  WRcursx(A0),D1       ;get current cursor X position
0B24  5C41                        ADDQ.W  #DScellW,D1          ;increment 1 character space   0.6
0B26  3141  0016                  MOVE.W  D1,WRcursx(A0)       ;save new cursor X position    0.6
0B2A  B268  0012                  CMP.W   WRlngthx(A0),D1      ;at end of line?               0.6
0B2E  6C08                        BGE.S   DSrtrni              ;yes, do carriage return       0.6
0B30  6068                        BRA.S   DScurs               ;show cursor                   0.6

                          ;
                          ; DScrsU -- cursor up
                          ;
0B32  6168                DScrsU  BSR.S   DScurs0              ;remove cursor
0B34  6048                        BRA.S   DSdecy               ;decrement cursor Y position

                          ;
                          ; DSrtrn -- return
                          ;
0B36  6164                DSrtrn  BSR.S   DScurs0              ;remove cursor
0B38  4268  0016          DSrtrni CLR.W   WRcursx(A0)          ;zero current cursor X position
0B3C  0838  0001  0766            BTST    #1,CPdspflg.w        ;auto line feed?
0B42  6704                        BOFF.S  DSincy               ;no, increment cursor Y position
0B44  6054                        BRA.S   DScurs               ;show cursor

                          ;
                          ; DScrsD -- cursor down
                          ;
0B46  6154                DScrsD  BSR.S   DScurs0              ;remove cursor
0B48  3228  0018          DSincy  MOVE.W  WRcursy(A0),D1       ;get current cursor Y position
0B4C  0641  000A                  ADDI.W  #DScellY,D1          ;increment 1 character space   0.6
0B50  3141  0018                  MOVE.W  D1,WRcursy(A0)       ;save new cursor Y position    0.6
0B54  B268  0014                  CMP.W   WRlngthy(A0),D1      ;at bottom of screen?          0.6
0B58  6F40                        BLE.S   DScurs               ;on bottom line?               0.6
0B5A  6000  0084                  BRA     DSclAL               ;yes, wrap to home position    0.6

                          ;
                          ; DScrsL -- cursor left
                          ;
0B5E  613C                DScrsL  BSR.S   DScurs0              ;remove cursor
0B60  4A68  0016          DSdecx  TST.W   WRcursx(A0)          ;at beginning of line?
0B64  6712                        BEQ.S   DSwrapx              ;yes, wrap to previous line
0B66  5D68  0016                  SUBQ.W  #DScellW,WRcursx(A0) ;decrement 1 character space
0B6A  602E                        BRA.S   DScurs               ;show cursor

```

```
                          ;
                          ; DScrsH -- cursor home
                          ;
0B6C 612E                 DScrsH  BSR.S   DScurs0                   ;remove cursor
0B6E 4268 0016            DScrsH1 CLR.W   WRcursx(A0)               ;zero current cursor X position
0B72 4268 0018                    CLR.W   WRcursy(A0)               ;zero current cursor Y position
0B76 6022                         BRA.S   DScurs                    ;show cursor

0B78 6112                 DSwrapx BSR.S   DSwrap                    ;
0B7A 3140 0016                    MOVE.W  D0,WRcursx(A0)            ;
0B7E 4A68 0018            DSdecy  TST.W   WRcursy(A0)               ;at top line?
0B82 6716                         BEQ.S   DScurs                    ;yes, show cursor
0B84 0468 000A 0018               SUBI.W  #DScellY,WRcursy(A0)      ;decrement 1 character space
0B8A 600E                         BRA.S   DScurs                    ;show cursor

0B8C 4280                 DSwrap  CLR.L   D0                        ;get current cursor X position
0B8E 3028 0012                    MOVE.W  WRlngthx(A0),D0           ;*
0B92 7406                         MOVEQ   #DScellW,D2               ;get character width
0B94 80C2                         DIVU    D2,D0                     ;
0B96 C0C2                         MULU    D2,D0                     ;
0B98 4E75                         RTS                               ;return

0B9A 6130                 DScurs  BSR.S   DScrsAd                   ;compute cursor address
0B9C 322A 0004            DScurs0 MOVE.W  CSlpch(A2),D1             ;get scan lines per character
0BA0 5341                         SUBQ.W  #1,D1                     ;set loop counter
0BA2 2868 0008                    MOVE.L  WRcuradr(A0),A4           ;get current cursor address
0BA6 3A28 001A                    MOVE.W  WRbitofs(A0),D5           ;get current cursor bit offset
0BAA 2E2A 000C                    MOVE.L  CSmask(A2),D7             ;get character mask
0BAE 0828 0000 0021               BTST    #vert,WRattr2(A0)         ;vertical orientation?
0BB4 6704                         BOFF.S  DScurs1                   ;no
0BB6 EABF                         ROR.L   D5,D7                     ;
0BB8 6004                         BRA.S   DScurs2                   ;
0BBA 4A64                 DScurs1 TST.W   -(A4)                     ;
0BBC EBBF                         ROL.L   D5,D7                     ;
0BBE 4687                 DScurs2 NOT.L   D7                        ;D7 = positioned inverted mask
0BC0 BF94                 DScurs3 EOR.L   D7,(A4)                   ;invert character
0BC2 98F8 0764                    SUBA.W  CPscnofs.W,A4             ;*
0BC4 51C9 FFF8                    DBRA    D1,DScurs3                ;*
0BCA 4E75                         RTS                               ;return

0BCC 4CA8 0060 0016       DScrsAd MOVEM.W WRcursx(A0),D5-D6         ;get current cursor position
0BD2 6100 00F8                    BSR     DSaddr                    ;compute cursor address
0BD6 3147 001A                    MOVE.W  D7,WRbitofs(A0)           ;save cursor bit offset
0BDA 214C 0008                    MOVE.L  A4,WRcuradr(A0)           ;save cursor address
0BDE 4E75                         RTS                               ;return

```

```
                          ;
                          ; DSclAL -- clear screen
                          ;
0BE0  6100  FF8C          DSclAL  BSR     DScrsH1             ;home cursor

                          ;
                          ; DSclES -- clear to end of screen
                          ;
0BE4  6138                DSclES  BSR.S   DSclEL              ;first clear this line
0BE6  0828  0000  0021            BTST    #vert,WRattr2(A0)   ;vertical orientation?
0BEC  6618                        BON.S   DSclES2             ;yes, clear vertical screen
                                                              ; --- clear to end of horizontal screen
0BEE  3C28  0018                  MOVE.W  WRcursy(A0),D6      ;get current cursor Y position
0BF2  0646  000A          DSclES1 ADDI.W  #DScellY,D6         ;increment to next line
0BF6  3028  0014                  MOVE.W  WRlngthy(A0),D0     ;get bottom of screen limit
0BFA  BC40                        CMP.W   D0,D6               ;at bottom of screen?
0BFC  6C1E                        BGE.S   DSclES9             ;yes, return
0BFE  9046                        SUB.W   D6,D0               ;compute number of scan lines to clear
0C00  4243                        CLR.W   D3                  ;set starting X position to 0
0C02  6158                        BSR.S   DSclrH              ;clear to bottom of screen
0C04  6016                        BRA.S   DSclES9             ;return
                                                              ; --- clear to end of vertical screen
0C06  3028  0018          DSclES2 MOVE.W  WRcursy(A0),D0      ;get current cursor Y position
0C0A  0640  000A          DSclES3 ADDI.W  #DScellY,D0         ;increment to next line
0C0E  B068  0014                  CMP.W   WRlngthy(A0),D0     ;at bottom of screen?
0C12  6C08                        BGE.S   DSclES9             ;yes, return
0C14  3C00                        MOVE.W  D0,D6               ;
0C16  4245                        CLR.W   D5                  ;
0C18  6124                        BSR.S   DSclrV              ;clear one vertical line
0C1A  60EE                        BRA.S   DSclES3             ;repeat until all lines cleared
                                                              ;
0C1C  4E75                DSclES9 RTS                         ;return

                          ;
                          ; DSclEL -- clear to end of line
                          ;
0C1E  6100  FF7C          DSclEL  BSR     DScurs0             ;remove cursor
0C22  4CA8  0060  0016            MOVEM.W WRcursx(A0),D5-D6   ;get current cursor X and Y
0C28  0828  0000  0021            BTST    #vert,WRattr2(A0)   ;vertical orientation
0C2E  6704                        BOFF.S  DSclEL1             ;no, clear horizontal line
0C30  610C                        BSR.S   DSclrV              ;clear one vertical line
0C32  6006                        BRA.S   DSclEL2             ;show cursor
                                                              ;
0C34  7009                DSclEL1 MOVEQ   #DScellY-1,D0       ;D0 = #scanlines to clear
0C36  3605                        MOVE.W  D5,D3               ;
0C38  6122                        BSR.S   DSclrH              ;clear one horizontal line
0C3A  6000  FF40          DSclEL2 BRA     DScurs0             ;show cursor
```

```
0C3E  3828  0012      DSclrV   MOVE.W  WRlngthx(A0),D4      ;get length of line
0C42  9845                     SUB.W   D5,D4                ;compute number of scan lines clear
0C44  6100  0086               BSR     DSaddr               ;compute cursor address
0C48  3238  0764               MOVE.W  CPscnofs.W,D1        ;get bytes per scan line
0C4C  2C2A  000C               MOVE.L  CSmask(A2),D6        ;get character mask
0C50  EEBE                     ROR.L   D7,D6                ;align character mask
0C52  CD94            DSclrV1  AND.L   D6,(A4)              ;clear one scan line
0C54  98C1                     SUBA.W  D1,A4                ;compute address of next scan line
0C56  51CC  FFFA               DBRA    D4,DSclrV1           ;repeat to end of line
0C5A  4E75                     RTS                          ;return

0C5C  3828  0012      DSclrH   MOVE.W  WRlngthx(A0),D4      ;D5 = x, D6 = y
0C60  5244                     ADDQ.W  #1,D4                ;
0C62  3A04                     MOVE.W  D4,D5                ;
0C64  9843                     SUB.W   D3,D4                ;
0C66  6100  0064               BSR     DSaddr               ;A4 = addr(x,y), D7 = bitnum
0C6A  9847                     SUB.W   D7,D4                ;
0C6C  4A47                     TST.W   D7                   ;
0C6E  6602                     BNE.S   DSclrH1              ;
0C70  4A5C                     TST.W   (A4)+                ;
0C72  3604            DSclrH1  MOVE.W  D4,D3                ;
0C74  0243  000F               ANDI.W  #$F,D3               ;
0C78  E844                     ASR.W   #4,D4                ;
0C7A  5344                     SUBQ.W  #1,D4                ;
0C7C  72FF                     MOVEQ   #-1,D1               ;
0C7E  3401                     MOVE.W  D1,D2                ;
0C80  EF69                     LSL.W   D7,D1                ;
0C82  E66A                     LSR.W   D3,D2                ;
0C84  2A4C            DSclrH2  MOVE.L  A4,A5                ;
0C86  4A47                     TST.W   D7                   ;
0C88  6702                     BEQ.S   DSclrH3              ;
0C8A  C35D                     AND.W   D1,(A5)+             ;
0C8C  3C04            DSclrH3  MOVE.W  D4,D6                ;
0C8E  6B06                     BMI.S   DSclrH5              ;
0C90  425D            DSclrH4  CLR.W   (A5)+                ;
0C92  51CE  FFFC               DBRA    D6,DSclrH4           ;
0C96  4A43            DSclrH5  TST.W   D3                   ;
0C98  6702                     BEQ.S   DSclrH6              ;
0C9A  C555                     AND.W   D2,(A5)              ;
0C9C  98F8  0764      DSclrH6  SUBA.W  CPscnofs.W,A4        ;
0CA0  51C8  FFE2               DBRA    D0,DSclrH2           ;
0CA4  4E75                     RTS                          ;return
```

```
0CA6  3028  0016        DStab    MOVE.W  WRcursx(A0),D0         ;get current cursor X position
0CAA  C0BC  0000  FFFF           AND.L   #$FFFF,D0              ;clear hi word
0CB0  7430                       MOVEQ   #DScellW*8,D2          ;
0CB2  80C2                       DIVU    D2,D0                  ;find next tab to right
0CB4  5240                       ADDQ.W  #1,D0                  ;
0CB6  C0C2                       MULU    D2,D0                  ;
0CB8  B068  0012                 CMP.W   WRlngthx(A0),D0        ;new x > right?
0CBC  6302                       BLS.S   DStab1                 ;no, change x
0CBE  4E75                       RTS                            ;return

0CC0  6100  FEDA        DStab1   BSR     DScurs0                ;remove cursor
0CC4  3140  0016                 MOVE.W  D0,WRcursx(A0)         ;save new cursor X position
0CC8  6000  FED0                 BRA     DScurs                 ;show cursor


                        ;
                        ; DSaddr -- compute cursor address
                        ;
                        ;        Enter:  D5 = x
                        ;                D6 = y
                        ;
                        ;        Exit:   DSaddr (x,y) in A4, bit offset in D7
                        ;
0CCC  2868  0004        DSaddr   MOVEA.L WRhomept(A0),A4        ;get home pointer for orientation
0CD0  0828  0000  0021           BTST    #vert,WRattr2(A0)      ;vertical orientation?
0CD6  6618                       BNE.S   DSaddrV                ;yes, compute address for vertical
                                                                ;
0CD8  DA68  000C        DSaddrH  ADD.W   WRhomeof(A0),D5        ;
0CDC  3E05                       MOVE.W  D5,D7                  ;
0CDE  0247  000F                 ANDI.W  #$F,D7                 ;
0CE2  E845                       ASR.W   #4,D5                  ;
0CE4  E345                       ASL.W   #1,D5                  ;
0CE6  98C5                       SUBA.W  D5,A4                  ;
0CE8  CCF8  0764                 MULU    CPscnofs.W,D6          ;
0CEC  99C6                       SUBA.L  D6,A4                  ;
0CEE  4E75                       RTS                            ;return
                                                                ;
0CF0  DC68  000C        DSaddrV  ADD.W   WRhomeof(A0),D6        ;
0CF4  3E06                       MOVE.W  D6,D7                  ;
0CF6  0247  000F                 ANDI.W  #$F,D7                 ;
0CFA  E846                       ASR.W   #4,D6                  ;
0CFC  E346                       ASL.W   #1,D6                  ;
0CFE  D8C6                       ADDA.W  D6,A4                  ;
0D00  CAF8  0764                 MULU    CPscnofs.W,D5          ;
0D04  99C5                       SUBA.L  D5,A4                  ;
0D06  4E75                       RTS                            ;return
```

```
                          ;
                          ; jump tables
                          ;
0D08  FE56                DScTbl   DATA.W  DScrsL-DScTbl        ;ctl-H: back space
0D0A  FF9E                         DATA.W  DStab-DScTbl         ;ctl-I: tab
0D0C  FE3E                         DATA.W  DScrsD-DScTbl        ;ctl-J: line feed
0D0E  FE2A                         DATA.W  DScrsU-DScTbl        ;ctl-K: cursor up
0D10  FE14                         DATA.W  DScrsR-DScTbl        ;ctl-L: cursor right
0D12  FE2E                         DATA.W  DSrtrn-DScTbl        ;ctl-M: carriage return

0D14  0048  FE58          DSeTbl   DATA.W  $48,DScrsH-DSeTbl    ;esc-H: home cursor
0D18  004A  FECC                   DATA.W  $4A,DSclAL-DSeTbl    ;esc-J: clear screen
0D1C  004B  FF0A                   DATA.W  $4B,DSclEL-DSeTbl    ;esc-K: clear to end of line
0D20  0059  FED0                   DATA.W  $59,DSclES-DSeTbl    ;esc-Y: clear to end of screen
0D24  FFFF                         DATA.W  -1                   ;end of table

0D26  FCE8                DSsTbl   DATA.W  DSst0-DSsTbl         ;state 0
0D28  FD18                         DATA.W  DSesc-DSsTbl         ;state 1

0D2A  00010D4E+           DSwndH   DATA.L  DScsetH              ;WRcharpt
0D2E  0008D55E                     DATA.L  DShomeH              ;home
0D32  0008D55E                     DATA.L  DShomeH              ;address
0D36  0000  0000  0000             DATA.W  0,0,0                ;WRhomeof,WRbasex,WRbasey
0D3C  02CF  022F                   DATA.W  DSmaxXH,DSmaxYH      ;right,bottom
0D40  0000  0000  0000             DATA.W  0,0,0                ;x,y,WRbitofs
0D46  0000  022F                   DATA.W  0,DSmaxYH            ;WRgrorgx,WRgrorgy
0D4A  00 1C                        DATA.B  0,$1C                ;attr1,attr2
0D4C  00 24                        DATA.B  0,WRlength           ;state, WRrcdlen

0D4E  00010D96+           DScsetH  DATA.L  DScsetV+CSdata       ;character set record pointer
0D52  000A  0006                   DATA.W  DScellY,DScellW      ;CSlpch, CSbpch
0D56  0020  005A                   DATA.W  32,90                ;CSfrstch, CSlastch
0D5A  FFFF  FFC0  0000             DATA.W  $FFFF,$FFC0,0        ;mask, dummy, attribs
                                   ;
                                   ; use vertical character set data
                                   ;

```

```
0D60  00010D84+          DSwndV  DATA.L  DScsetV                 ;WRcharpt
0D64  0008D506                   DATA.L  DShomeV                 ;home
0D68  0008D506                   DATA.L  DShomeV                 ;address
0D6C  0000  0000  0000           DATA.W  0,0,0                   ;WRhomeof,WRbasex,WRbasey
0D72  022D  02CF                 DATA.W  DSmaxXV,DSmaxYV         ;right,bottom
0D76  0000  0000  0000           DATA.W  0,0,0                   ;x,y,WRbitofs
0D7C  0000  02CF                 DATA.W  0,DSmaxYV               ;WRgrorgx,WRgrorgy
0D80  00 1D          .           DATA.B  0,$1D                   ;attr1,attr2
0D82  00 24                      DATA.B  0,WRlength              ;state, WRrcdlen

0D84  00010D96+          DScsetV DATA.L  DScsetV+CSdata          ;character set record pointer
0D88  0006  000A                 DATA.W  DScellW,DScellY         ;CSlpch, CSbpch
0D8C  0020  005A                 DATA.W  32,90                   ;CSfrstch, CSlastch
0D90  003F  FFFF  0100           DATA.W  $003F,$FFFF,256         ;mask, dummy, attribs
                                 ;
                                 ; vertical character set data
                                 ;
0D96  00 00 00 00 00 00          DATA.B  0,0,0,0,0,0             ; blank
0D9C  00 00 FD 00 00 00          DATA.B  0,0,$FD,0,0,0           ; !
0DA2  00 E0 00 E0 00 00          DATA.B  0,$E0,0,$E0,0,0         ; "
0DA8  28 FE 28 FE 28 00          DATA.B  $28,$FE,$28,$FE,$28,0   ; #
0DAE  24 54 FE 54 48 00          DATA.B  $24,$54,$FE,$54,$48,0   ; $
0DB4  C4 C8 10 26 46 00          DATA.B  $C4,$C8,$10,$26,$46,0   ; %
0DBA  6C 92 6A 04 0A 00          DATA.B  $6C,$92,$6A,$04,$0A,0   ; &
0DC0  00 00 20 C0 00 00          DATA.B  0,0,$20,$C0,0,0         ; '
0DC6  00 38 44 82 00 00          DATA.B  0,$38,$44,$82,0,0       ; (
0DCC  00 00 82 44 38 00          DATA.B  0,0,$82,$44,$38,0       ; )
0DD2  08 2A 1C 2A 08 00          DATA.B  $08,$2A,$1C,$2A,$08,0   ; *
0DD8  08 08 3E 08 08 00          DATA.B  $08,$08,$3E,$08,$08,0   ; +
0DDE  00 01 07 00 00 00          DATA.B  0,$01,$07,0,0,0         ; ,
0DE4  10 10 10 10 10 00          DATA.B  $10,$10,$10,$10,$10,0   ; -
0DEA  00 00 02 00 00 00          DATA.B  0,0,$02,0,0,0           ; .
0DF0  04 08 10 20 40 00          DATA.B  $04,$08,$10,$20,$40,0   ; /
0DF6  7C 8A 92 A2 7C 00          DATA.B  $7C,$8A,$92,$A2,$7C,0   ; 0
0DFC  00 42 FE 02 00 00          DATA.B  0,$42,$FE,$02,0,0       ; 1
0E02  46 8A 92 92 62 00          DATA.B  $46,$8A,$92,$92,$62,0   ; 2
0E08  84 82 92 B2 CC 00          DATA.B  $84,$82,$92,$B2,$CC,0   ; 3
0E0E  18 28 48 FE 08 00          DATA.B  $18,$28,$48,$FE,$08,0   ; 4
0E14  E4 A2 A2 A2 9C 00          DATA.B  $E4,$A2,$A2,$A2,$9C,0   ; 5
0E1A  3C 52 92 92 1C 00          DATA.B  $3C,$52,$92,$92,$1C,0   ; 6
0E20  80 8E 90 A0 C0 00          DATA.B  $80,$8E,$90,$A0,$C0,0   ; 7
0E26  6C 92 92 92 6C 00          DATA.B  $6C,$92,$92,$92,$6C,0   ; 8
0E2C  62 92 92 94 78 00          DATA.B  $62,$92,$92,$94,$78,0   ; 9
0E32  00 00 24 00 00 00          DATA.B  0,0,$24,0,0,0           ; :
0E38  00 01 26 00 00 00          DATA.B  0,$01,$26,0,0,0         ; ;
0E3E  00 10 28 44 82 00          DATA.B  0,$10,$28,$44,$82,0     ; <
0E44  00 28 28 28 28 00          DATA.B  0,$28,$28,$28,$28,0     ; =
0E4A  00 82 44 28 10 00          DATA.B  0,$82,$44,$28,$10,0     ; >
0E50  40 80 9A A0 40 00          DATA.B  $40,$80,$9A,$A0,$40,0   ; ?
0E56  7C 82 9A 9A 7A 00          DATA.B  $7C,$82,$9A,$9A,$7A,0   ; @
0E5C  3E 48 88 48 3E 00          DATA.B  $3E,$48,$88,$48,$3E,0   ; A
0E62  FE 92 92 92 6C 00          DATA.B  $FE,$92,$92,$92,$6C,0   ; B
0E68  7C 82 82 82 44 00          DATA.B  $7C,$82,$82,$82,$44,0   ; C
0E6E  FE 82 82 82 7C 00          DATA.B  $FE,$82,$82,$82,$7C,0   ; D
```

```
0E74  FE 92 92 82 82 00 1902*        DATA.B  $FE,$92,$92,$82,$82,0    , E
0E7A  FE 90 90 80 80 00 1903*        DATA.B  $FE,$90,$90,$80,$80,0    , F
0E80  7C 82 8A 8A 4C 00 1904*        DATA.B  $7C,$82,$8A,$8A,$4C,0    , G
0E86  FE 10 10 10 FE 00 1905*        DATA.B  $FE,$10,$10,$10,$FE,0    , H
0E8C  00 82 FE 82 00 00 1906*        DATA.B  0,$82,$FE,$82,0,0        , I
0E92  04 82 82 FC 80 00 1907*        DATA.B  $04,$82,$82,$FC,$80,0    , J
0E98  FE 10 28 44 82 00 1908*        DATA.B  $FE,$10,$28,$44,$82,0    , K
0E9E  FE 02 02 02 02 00 1909*        DATA.B  $FE,$02,$02,$02,$02,0    , L
0EA4  FE 40 30 40 FE 00 1910*        DATA.B  $FE,$40,$30,$40,$FE,0    , M
0EAA  FE 20 10 08 FE 00 1911*        DATA.B  $FE,$20,$10,$08,$FE,0    , N
0EB0  7C 82 82 82 7C 00 1912*        DATA.B  $7C,$82,$82,$82,$7C,0    , O
0EB6  FE 90 90 90 60 00 1913*        DATA.B  $FE,$90,$90,$90,$60,0    , P
0EBC  7C 82 8A 84 7A 00 1914*        DATA.B  $7C,$82,$8A,$84,$7A,0    , Q
0EC2  FE 90 98 94 62 00 1915*        DATA.B  $FE,$90,$98,$94,$62,0    , R
0EC8  64 92 92 92 4C 00 1916*        DATA.B  $64,$92,$92,$92,$4C,0    , S
0ECE  80 80 FE 80 80 00 1917*        DATA.B  $80,$80,$FE,$80,$80,0    , T
0ED4  FC 02 02 02 FC 00 1918*        DATA.B  $FC,$02,$02,$02,$FC,0    , U
0EDA  F8 04 02 04 F8 00 1919*        DATA.B  $F8,$04,$02,$04,$F8,0    , V
0EE0  FC 02 1C 02 FC 00 1920*        DATA.B  $FC,$02,$1C,$02,$FC,0    , W
0EE6  C6 28 10 28 C6 00 1921*        DATA.B  $C6,$28,$10,$28,$C6,0    , X
0EEC  C0 20 1E 20 C0 00 1922*        DATA.B  $C0,$20,$1E,$20,$C0,0    , Y
0EF2  86 8A 92 A2 C2 00 1923*        DATA.B  $86,$8A,$92,$A2,$C2,0    , Z
0EF8  0000              1924*        DATA.W  0
                        1925*
```

```
                                 include 'CC.PROM.LD'    ;local disk driver
                         ;
                         ; File: CC.PROM.LD.TEXT
                         ; Date: 29-Jun-82
                         ; By:   L. Franklin

                         ;
                         ; Lboot -- Local Corvus disk boot processing
                         ;
0EFA  227C  0000  0771   Lboot   movea.l #CPsl1typ,a1       ;get pointer to slot 1 type
0F00  7001                       moveq   #1,d0              ;get initial slot number
                                                            ;
0F02  1231  00FF         Lboot10 move.b  -1(a1,d0),d1       ;get device type
0F06  323C  0001                 cmp.b   #DTlocl,d1         ;is this a local disk interface?
0F0A  670C                       beq.s   Lboot30            ;yes, use it for booting
0F0C  5240                       addq    #1,d0              ;update slot number
0F0E  B07C  0004                 cmp.w   #4,d0              ;have we looked at all slots?
0F12  6FEE                       ble.s   Lboot10            ;no, check next slot
0F14  7EFF                       moveq   #-1,d7             ;set error return code
0F16  6056                       bra.s   Lboot90            ;return (can not find local disk)
                                                            ;
0F18  11C0  0700         Lboot30 move.b  d0,CPbtslot.w      ;set boot slot number
0F1C  4238  0701                 clr.b   CPbtsrvr.w         ;set boot server number
0F20  4DFA  006A+                lea     LDblkIO,a6         ;set boot disk blk i/o subr pointer
0F24  21CE  0714                 move.l  a6,CPblkio.w       ;*
0F28  4DFA  010C+                lea     LDdskIO,a6         ;set boot disk i/o subr pointer
0F2C  21CE  0718                 move.l  a6,CPdskio.w       ;*
                         ;
                         ; fall through to Lboot10 (used by OMNINET boot too)
                         ;

                         ;
                         ; Lboot80 -- Get 4 boot blocks from Corvus disk
                         ;              (code shared by local disk boot and OMNINET disk boot)
                         ;
0F30  207C  0008  E000   Lboot80 movea.l #USRbase,a0     ;get block buffer pointer
0F34  1C38  0701                 move.b  CPbtsrvr.w,d6   ;get boot server number
0F3A  6D32                       blt.s   Lboot90         ;just return if invalid server number
0F3C  E14E                       lsl.w   #8,d6           ;*
0F3E  1C38  0700                 move.b  CPbtslot.w,d6   ;get boot slot number
                                                         ;
0F42  10BC  0014                 move.b  #$14,(a0)       ;set "boot" command
0F46  103C  0007                 move.b  #$07,d0         ;set boot block number
0F4A  323C  0603                 move.w  #$603,d1        ;
0F4E  6120                       bsr.s   LDgetBB         ;get next boot block
0F50  6D1C                       blt.s   Lboot90         ;just return if error
                                                         ;
0F52  323C  0403                 move.w  #$403,d1        ;
0F56  6118                       bsr.s   LDgetBB         ;get next boot block
0F58  6D14                       blt.s   Lboot90         ;just return if error
                                                         ;
0F5A  323C  0203                 move.w  #$203,d1        ;
0F5E  6110                       bsr.s   LDgetBB         ;get next boot block
0F60  6D0C                       blt.s   Lboot90         ;just return if error
                                                         ;
0F62  323C  0003                 move.w  #$3,d1          ;
0F66  6108                       bsr.s   LDgetBB         ;get next boot block
0F68  6D04                       blt.s   Lboot90         ;just return if error
0F6A  D0FC  0004                 adda.w  #$4,a0          ;get pointer to boot code
                                                         ;
0F6E  4E75               Lboot90 rts                     ;return

0F70  1140  0001         LDgetBB move.b  d0,1(a0)        ;set boot block number
0F74  7402                       moveq   #2,d2           ;get number of bytes to send
0F76  7A33                       moveq   #DskWrit,d5     ;get "write" command
0F78  4E96                       jsr     (a6)            ;send write command
0F7A  D0C1                       adda.w  d1,a0           ;
0F7C  343C  0201                 move.w  #513,d2         ;get number of bytes to receive
0F80  7A32                       moveq   #DskRead,d5     ;get "read" command
0F82  4E96                       jsr     (a6)            ;send read command
0F84  6D04                       blt.s   LDgetBX         ;just return if error
0F86  90C1                       suba.w  d1,a0           ;
0F88  5340                       subq    #1,d0           ;update boot block number
0F8A  4E75               LDgetBX rts                     ;return

```

```
                              ;
                              ; LDblkIO - Read or write a local disk block subroutine
                              ;
                              ;       Enter:  A0.L - Buffer address
                              ;               D0.W - Block number
                              ;               D1.W - Drive number
                              ;               D5.W - Read ($32) or Write ($33) command
                              ;               D6.B - Slot number
                              ;
                              ;       Exit.   A0.L - Next free location in buffer
                              ;               D0.W - Updated block number
                              ;               D7.W - IORESULT (disk controller status)
                              ;
                              ;       All other registers are preserved.
                              ;
                              ;       Corvus controller status register [3(a1)]:
                              ;
                              ;       bit 7. controller ready    off - ready          on - not ready
                              ;       bit 6. bus direction       off - host to cntlr  on - cntlr to host
                              ;
0F8C  48E7  E040              LDblkIO movem.l a1/d0-d2,-(sp)  ;Save registers
0F90  6100  F6B6                      bsr     SlotAdr         ;A1 = I/O port address
0F94  3405                            move.w  d5,d2           ;Send a read ($32) or
0F96  6168                            bsr.s   LDsend1         ;  write ($33) block command
0F98  3401                            move.w  d1,d2           ;
0F9A  6150                            bsr.s   LDsend          ;Send drive number
0F9C  3400                            move.w  d0,d2           ;
0F9E  614C                            bsr.s   LDsend          ;Send LSB of block
0FA0  E04A                            lsr.w   #8,d2           ;
0FA2  6148                            bsr.s   LDsend          ;Send MSB of block
0FA4  0C45  0033                      cmpi.w  #DskWrit,d5     ;Are we reading or writing?
0FA8  661C                            bne.s   LDrio1          ;Reading
                                                              ;
                                                              ;Write block processing
                                                              ;
0FAA  343C  01FF                      move.w  #$1FF,d2        ;Block size - 1
0FAE  0829  0007  0003        LDwio1  btst    #7,3(a1)        ;Test controller status
0FB4  66F8                            bne.s   LDwio1          ;Wait until controller ready
0FB6  1358  0001                      move.b  (a0)+,1(a1)     ;Send a byte
0FBA  51CA  FFF2                      dbra    d2,LDwio1       ;Loop until done
0FBE  6156                            bsr.s   LDwait          ;Wait for line to turn
0FC0  1E29  0001                      move.b  1(a1),d7        ;Fetch result code
0FC4  601C                            bra.s   LDrtrn          ;Return

```

```
                        2049*                                      ;
                        2050*                                      ;Read block processing
                        2051*                                      ;
0FC6  614E              2052* LDrio1  bsr.s    LDwait              ;Wait for the line to turn
0FC8  1E29  0001        2053*         move.b   1(a1),d7            ;Fetch result code
                        2054*                                      ;
0FCC  0829  0007  0003  2055* LDrio3  btst     #7,3(a1)            ;Test controller status
0FD2  66F8              2056*         bon.s    LDrio3              ;Wait until controller ready
0FD4  0829  0006  0003  2057*         btst     #6,3(a1)            ;Test bus direction
0FDA  6706              2058*         boff.s   LDrtrn              ;Finished if "host to controller"
0FDC  10E9  0001        2059*         move.b   1(a1),(a0)+         ;Store next byte
0FE0  60EA              2060*         bra.s    LDrio3              ;Go get any more
                        2061*                                      ;
0FE2  4CDF  0207        2062* LDrtrn  movem.l  (sp)+,a1/d0-d2      ;Restore registers
0FE6  5240              2063*         addq.w   #1,d0               ;Update block number
                        2064* ; ----  move.b   d7,CPdiskRC.w       ;Save current disk return code
0FE8  4887              2065*         ext.w    d7                  ;Set return condition code
0FEA  4E75              2066*         rts                          ;Return
                        2067*
```

```
                          ;
                          ; LDsend -- Send a byte to the disk port subroutine
                          ;
                          ;       Enter:  A1.L - I/O port address
                          ;               D2.B - Byte to send
                          ;
                          ;       All registers are preserved.
                          ;
0FEC 0829 0007 0003       LDsend  btst    #7,3(a1)        ;Test controller status
0FF2 66F8                         bon.s   LDsend          ;Wait until controller ready
0FF4 1342 0001                    move.b  d2,1(a1)        ;Send the byte
0FF8 4E75                         rts                     ;Return

                          ;
                          ; LDsend1 -- Send first byte to the disk port subroutine
                          ;
                          ;       Enter:  A1.L - I/O port address
                          ;               D2.B - Byte to send
                          ;
                          ;       All registers are preserved.
                          ;
0FFA 46DF                 LDsend0 move.w  (sp)+,sr        ;enable interrupts
0FFC 4E71                         nop                     ;leave some time for interrupt processing
0FFE 4E71                         nop                     ;*
1000 40E7                 LDsend1 move.w  sr,-(sp)        ;save interrupt level
1002 007C 0700                    ori.w   #$0700,sr       ;disable interrupts
1006 0829 0007 0003               btst    #7,3(a1)        ;test controller status
100C 66EC                         bon.s   LDsend0         ;wait until controller ready
100E 1342 0001                    move.b  d2,1(a1)        ;send first byte
1012 46DF                         move.w  (sp)+,sr        ;enable interrupts
1014 4E75                         rts                     ;return

                          ;
                          ; LDwait -- Wait for the line to turn subroutine
                          ;
                          ;       Enter:  A1.L - I/O port address
                          ;
1016 2F00                 LDwait  move.l  d0,-(sp)        ;save register
1018 7064                         moveq   #100,d0         ;wait a little bit
101A 51C8 FFFE            LDwait1 dbra    d0,LDwait1      ;*
101E 201F                         move.l  (sp)+,d0        ;restore register
1020 6102                         bsr.s   LDwait2         ;check two times in case of glitch
1022 4E71                         nop                     ;*
                                                          ;
1024 0829 0007 0003       LDwait2 btst    #7,3(a1)        ;test controller status
102A 66F8                         bon.s   LDwait2         ;wait until controller ready
102C 0829 0006 0003               btst    #6,3(a1)        ;test bus direction
1032 67F0                         boff.s  LDwait2         ;wait until "controller to host"
1034 4E75                         rts                     ;return
```

```
                         ;
                         ; LDdskIO - Read from/Write to Corvus disk
                         ;
                         ,       Enter:  A0.L - Buffer address
                         ;               D2.W - Count
                         ;               D5.W - Read ($32) or Write ($33) command
                         ;               D6.B - Slot number
                         ;
                         ;       Exit:   D7.W - IORESULT (disk controller status)
                         ;
                         ;       All other registers are preserved.
                         ,
1036  48E7  F0C0         LDdskIO movem.l d0-d3/a0-a1,-(sp);Save registers
103A  6100  F60C                 bsr     SlotAdr         ;A1 = I/O port address
103E  3602                       move.w  d2,d3           ;get count
1040  5343                       subq.w  #1,d3           ,Set DBRA loop length
1042  0C45  0033                 cmpi.w  #DskWrit,d5     ;Are we reading or writing?
1046  6614                       bne.s   LDdsk2          ;Reading
                         ;
                         ; Write Corvus disk processing
                         ,
1048  1418                       move.b  (a0)+,d2        ,get first byte
104A  61B4                       bsr.s   LDsend1         ,send first byte
104C  6006                       bra.s   LDdsk1a         ,send rest of bytes
104E  1418               LDdsk1  move.b  (a0)+,d2        ;get next byte
1050  6100  FF9A                 bsr     LDsend          ,send next byte
1054  51CB  FFF8         LDdsk1a dbra    d3,LDdsk1       ;loop until done
1058  7E00                       moveq   #0,d7           ;force successful result code
105A  601E                       bra.s   LDdsk9          ;finished
                         ,
                         ; Read Corvus disk processing
                         ;
105C  61B8               LDdsk2  bsr.s   LDwait          ;Wait for the line to turn
105E  1E29  0001                 move.b  1(a1),d7        ,Fetch result code
1062  10C7                       move.b  d7,(a0)+        ,Store first byte
                                                         ;
1064  0829  0007  0003   LDdsk3  btst    #7,3(a1)        ;Test controller status
106A  66F8                       bon.s   LDdsk3          ,Wait until controller ready
106C  0829  0006  0003           btst    #6,3(a1)        ;Test bus direction
1072  6706                       boff.s  LDdsk9          ,Finished if "host to controller"
1074  10E9  0001                 move.b  1(a1),(a0)+     ,Store next byte
1078  60EA                       bra.s   LDdsk3          ;Go get any more
                                                         ;
107A  4CDF  030F         LDdsk9  movem.l (sp)+,d0-d3/a0-a1;Restore registers
                         ; ----  move.b  d7,CPdiskRC.w   ;Save current disk return code
107E  4887                       ext.w   d7              ;Set return condition code
1080  4E75                       rts                     ;Return
```

```
                          ;
                          ; LDsync -- Synchronise with Corvus disk controller
                          ;
                          ;       Enter.  D6.B - slot number
                          ;
                          ;       Exit.   D7.W - 0 = no timeout (EQ), -1 = timeout (NE)
                          ;
                          ;       All other registers are preserved.
                          ;
1082  48E7  8040          LDsync  movem.l d0/a1,-(sp)     ;save registers
1086  6100  F5C0                  bsr     SlotAdr         ;get slot address
108A  3E3C  07D0                  move.w  #2000,d7        ;set timeout counter
                                                          ;
108E  137C  00FF  0001    LDsync1 move.b  #$FF,1(a1)      ;send invalid command to controller
1094  303C  0400                  move.w  #1024,d0        ;wait about 1 ms
1098  51C8  FFFE          LDsync2 dbra    d0,LDsync2      ;*
109C  0829  0006  0003            btst    #6,3(a1)        ;test bus direction
10A2  6606                        bon.s   LDsync3         ;go on if "controller to host"
10A4  51CF  FFE8                  dbra    d7,LDsync1      ;send invalid command again
10A6  600C                        bra.s   LDsync5         ;set timeout error and return
                                                          ;
10AA  0829  0007  0003    LDsync3 btst    #7,3(a1)        ;test controller status
10B0  6706                        boff.s  LDsync6         ;go on if controller ready
10B2  51CF  FFF6                  dbra    d7,LDsync3      ;check controller status again
                                                          ;
10B6  7EFF                LDsync5 moveq   #-1,d7          ;indicate controller timeout
10B8  600A                        bra.s   LDsync9         ;return
                                                          ;
10BA  0C29  008F  0001    LDsync6 cmpi.b  #$8F,1(a1)      ;did controller respond with error?
10C0  66CC                        bne.s   LDsync1         ;no, send invalid command again
10C2  7E00                        moveq   #0,d7           ;indicate no controller timeout
                                                          ;
10C4  4CDF  0201          LDsync9 movem.l (sp)+,d0/a1     ;restore registers
10C8  4A47                        tst.w   d7              ;set return condition code
10CA  4E75                        rts                     ;return
```

```
                                include 'CC.PROM.OD'    ;OMNINET disk driver
                        ;
                        ; File: CC.PROM.OD.TEXT
                        ; Date: 29-Jun-82

                        ;
                        ; OMNINET disk driver data area equates
                        ;
00000000                DCmd     EQU     0        ;byte - disk command offset
00000001                DCdrv    EQU     1+DCmd   ;byte - offset for drive number
00000002                DCblklo  EQU     2+DCmd   ,byte - LSB of block number to read or write
00000003                DCblkhi  EQU     3+DCmd   ;byte - MSB      "     "
00000004                DClen    EQU     4+DCmd   ;word - length ot request (in bytes)
                        ;
                        ;        result vector and header used for all setuprecv commands
                        ,
00000006                RHdr     EQU     6        ,
00000006                RHpktRC  EQU     0+RHdr   ;byte - return code from transporter
00000007                RHsor    EQU     1+RHdr   ,byte - the source of the message
00000008                RHpktLN  EQU     2+RHdr   ;word - total length of data portion of packet
0000000A                RHdskLN  EQU     4+RHdr   ;word - length of info returned from drive
0000000C                RHdskRC  EQU     6+RHdr   ;byte - return code from drive
                        ;
                        ;        result vector and header for all sendmsg commands
                        ;
0000000E                SHdr     EQU     14       ,
0000000E                SHpktRC  EQU     0+SHdr   ;byte - return code from transporter
                        ;        EQU     1+SHdr   ;byte - unused
                        ;        EQU     2+SHdr   ;word - unused
00000012                SHtoLN   EQU     4+SHdr   ;word - number of bytes to send to drive
00000014                SHfmLN   EQU     6+SHdr   ;word - number of bytes expected from drive
                                                  ;
00000016                GData    EQU     22       ,word - area to receive "GO" into
                        ;
                        ,        area used for constructing Transporter commands
                        ;
00000018                TCmd     EQU     24       ;
00000018                TCop     EQU     0+TCmd   ,byte - op code
00000019                TCrADhi  EQU     1+TCmd   ;byte - result address HI
0000001A                TCrADlo  EQU     2+TCmd   ;word - result address MED, LO
0000001C                TCsock   EQU     4+TCmd   ,byte - socket number
0000001D                TCdADhi  EQU     5+TCmd   ,byte - data buffer address HI
0000001E                TCdADlo  EQU     6+TCmd   ;word - data buffer address MED, LO
00000020                TCdtaLN  EQU     8+TCmd   ;word - data length
00000022                TChdrLN  EQU     10+TCmd  ;byte - header length
00000023                TCdest   EQU     11+TCmd  ;byte - destination host number
                                                  ,
00000024                ODdw     EQU     36       ,lint - temporary buffer (for 3 byte nmbrs)
00000025                ODdwhi   EQU     37       ;byte - temporary HI
00000026                ODdwlo   EQU     38       ;word - temporary MID, LO
00000028                ODwrAD   EQU     40       ;lint - to save buffer address for CWrites
0000002C                ODvalid  EQU     44       ;word - for marking buffer as valid

```

```
                ;
                ;
                ;
00030FA1        StrAdr  EQU     $30FA1  ;address of Transporter register
00030F7F        RdyAdr  EQU     $30F7F  ;address of VIA register A, used for Omninet ready
0000FFFE        TOintvl EQU     $FFFE   ;timeout interval
                ;
                ; Transporter Return Codes
                ;
000000FF        Waiting EQU     $FF     ;
000000FE        CmdAcpt EQU     $FE     ;
000000C0        Echoed  EQU     $C0     ;echo command was successful
                                        ;
00000080        GaveUp  EQU     $80     ;aborted a send command after MaxRetries tries
00000081        TooLong EQU     $81     ;last message sent was too long for the receiver
00000082        NoSockt EQU     $82     ;sent to an unititialized socket
00000083        HdrErr  EQU     $83     ;sender's header length did not match receiver's
00000084        BadSock EQU     $84     ;illegal socket number
00000085        Inuse   EQU     $85     ;tried to set up a receive on an active socket
00000086        BadDest EQU     $86     ;sent to an illegal host number
                                        ;
00000090        NoTrans EQU     $90     ;could not strobe cmd addr to Transporter
00000091        TimeOut EQU     $91     ;timed out waiting for an Omninet event
00000092        NoBufr  EQU     $92     ;tried a CRRead without a valid write buffer
                ;
                ; Transporter Opcodes
                ;
000000F0        RecvOp  EQU     $F0     ;SETUPRECV opcode
00000040        SendOp  EQU     $40     ;SENDMSG opcode
00000020        InitOp  EQU     $20     ;INIT opcode
00000010        EndOp   EQU     $10     ;ENDRECV opcode
00000008        DebOp   EQU     $08     ;PEEK/POKE opcode
00000002        EchoOp  EQU     $02     ;ECHOCMD opcode
00000001        WhoOp   EQU     $01     ;WHOAMI opcode
                                        ;
000000A0        RestSkt EQU     $A0     ;dest. socket for REST packet
000000B0        CnstSkt EQU     $B0     ;socket for Constellation protocol

```

```
                         ;
                         ; Oboot -- OMNINET disk server boot processing
                         ;
10CC 11FC 0005 0700      Oboot    move.b  #5,CPbtslot.w          ;set boot slot number
10D2 4DFA 005C+                   lea     ODblkIO,a6             ;set boot disk blk i/o subr pointer
10D6 21CE 0714                    move.l  a6,CPblkio.w           ;*
10DA 4DFA 00B4+                   lea     ODdskIO,a6             ;set boot disk i/o subr pointer
10DE 21CE 0718                    move.l  a6,CPdskio.w           ;*
10E2 6000 FE4C                    bra     Lboot80                ;load boot code
                                                                 ;  (Lboot80 is in CC.PROM.LD)

                         ;
                         ; ODcomnd -- send simple command to Transporter
                         ;
                         ;        Enter:  D0.B - Transporter command
                         ;                D1.B - Destination host number (if echo)
                         ;
                         ;        Exit:   D7.B - IORESULT (OMNINET status)
                         ;
10E6 48E7 8060           ODcomnd  movem.l a1-a2/d0,-(sp)             ;save registers
10EA 227C 0008 DFD0               move.l  #CPomnibf,a1               ;get pointer to Data Area
10F0 4869 0006                    pea     RHdr(a1)                   ;get pointer to result record
10F4 235F 0018                    move.l  (sp)+,TCop(a1)             ;set result record pointer
10F8 1340 0018                    move.b  d0,TCop(a1)                ;set Transporter command
10FC 1341 001C                    move.b  d1,TCsock(a1)              ;set destination host number (echo)
1100 137C 00FF 0006               move.b  #Waiting,RHpktRC(A1)       ;set Transporter waiting flag
1106 45E9 0018                    lea     TCmd(A1),A2                ;get command address
110A 6100 00EE                    bsr     StrobIt                    ;strobe command address to Transporte
110E 661A                         bne.s   ODcmd9                     ;Transporter not responding
1110 303C FFFE                    move.w  #TOintvl,D0                ;get timeout interval
                                                                     ;
1114 0C29 00FF 0006      ODcmd1   cmpi.b  #Waiting,RHpktRC(A1)       ;has Transporter responded?
111A 660A                         bne.s   ODcmd2                     ;yes, ready to return
111C 51C8 FFF6                    dbra    D0,ODcmd1                  ;timeout yet?
1120 137C 0091 0006               move.b  #TimeOut,RHpktRC(A1)       ;yes, set timeout error and return
                                                                     ;
1126 1E29 0006           ODcmd2   move.b  RHpktRC(A1),d7             ;get Transporter return code
                                                                     ;
112A 4CDF 0601           ODcmd9   movem.l (sp)+,a1-a2/d0             ;restore registers
                         ; ----   move.b  d7,CPomniRC.w              ;save current OMNINET return code
112E 4E75                         rts                                ;return

```

```
                        2343* ;
                        2344* ; ODblkIO - Read or write an OMNINET disk server block subroutine
                        2345* ;
                        2346* ;       Enter:  A0.L - Buffer address
                        2347* ;               D0.W - Block number
                        2348* ;               D1.W - Drive number
                        2349* ;               D5.W - Read ($32) or Write ($33) command
                        2350* ;               D6.W - Destination host number * 256
                        2351* ;
                        2352* ;       Exit:   A0.L - Next free location in buffer
                        2353* ;               D0.W - Updated block number
                        2354* ;               D7.W - IORESULT (OMNINET/disk controller status)
                        2355* ;
                        2356* ;       All other registers are preserved.
                        2357* ;
1130 48E7 E060          2358* ODblkIO movem.l d0-d2/a1-a2,-(sp);Save registers
1134 227C 0008 DFD0     2359*         move.l  #CPomnibf,a1    ;A1 points to the start of the Data Area
113A 4269 002C          2360*         clr.w   ODvalid(A1)     ;buffer valid = False... see ODdskIO
113E 1345 0000          2361*         move.b  D5,DCmd(A1)     ;Stuff disk command - read or write
1142 1341 0001          2362*         move.b  D1,DCDrv(A1)    ;stuff drive number
1146 1340 0002          2363*         move.b  D0,DCBlkLo(A1)  ;lo order byte of block number
114A E048               2364*         lsr.w   #8,D0           ;
114C 1340 0003          2365*         move.b  D0,DCBlkHi(A1)  ;hi order byte of block number
1150 337C 0200 0004     2366*         move.w  #512,DCLen(A1)  ;set length to 512...
1156 0C45 0033          2367*         cmpi.w  #DskWrit,D5     ;Are we reading or writing?
115A 6614               2368*         bne.s   ODblk2          ;Reading
                        2369*                                 ;
115C 337C 0204 0012     2370* ODblk1  move.w  #516,SHtoLN(A1) ;number of bytes to send to drive
1162 4269 0014          2371*         clr.w   SHfmLN(A1)      ;number of bytes expected back
1166 2348 0028          2372*         move.l  A0,GDwrAD(A1)   ;save address of REST of data
116A 6100 01EA          2373*         bsr     LongCmds        ;Writing
116E 6010               2374*         bra.s   ODblk3          ;return
                        2375*                                 ;
1170 337C 0004 0012     2376* ODblk2  move.w  #4,SHtoLN(A1)   ;number of bytes to send to drive
1176 337C 0200 0014     2377*         move.w  #512,SHfmLN(A1) ;number of bytes expected back
117C 6100 024C          2378*         bsr     ShortCmds       ;
                        2379*                                 ;
1180 4CDF 0607          2380* ODblk3  movem.l (sp)+,d0-d2/a1-a2,Restore registers
1184 D0FC 0200          2381*         adda.w  #512,a0         ;Update buffer pointer
1188 0640 0001          2382*         addi.w  #1,d0           ;Update disk block number
                        2383* ; ----  move.b  d7,CPdiskRC.w   ;Save current disk return code
118C 4887               2384*         ext.w   d7              ;Set return condition code
118E 4E75               2385*         rts                     ;Return
                        2386*
```

```
                          ;
                          ; ODdskIO - Read from/write to Corvus disk
                          ;
                          ;       Enter:  A0.L - Buffer address
                          ;               D2.W - Count
                          ;               D5.W - Read ($32) or Write ($33) command
                          ;               D6.W - Destination host number * 256
                          ;
                          ;       Exit:   D7.W - IORESULT (OMNINET/disk controller status)
                          ;
                          ;       All other registers are preserved.
                          ;
1190  48E7  E0E0          ODdskIO movem.l d0-d2/a0-a2,-(sp);Save registers
1194  227C  0008  DFD0            move.l  #CPomnibf,a1     ;A1 points to the start of the Data Area
119A  0C45  0033                  cmpi.w  #DskWrit,D5      ;do we want to read or write
119E  6616                        bne.s   ODdsk2           ;read
                                                           ;
11A0  3342  0012          ODdsk1  move.w  D2,SHtoLN(A1)    ;
11A4  2358  0000                  move.l  (A0)+,DCmd(A1)   ;move first four bytes of send data to DiskCmd
11A8  2348  0028                  move.l  A0,ODwrAD(A1)    ;save address of REST of data
11AC  4247                        clr.w   D7               ;force successful IOResult
11AE  337C  FFFF  002C            move.w  #-1,ODvalid(A1)  ;mark send buffer as valid...
11B4  603C                        bra.s   ODdsk9           ;return
                                                           ;
11B6  0C69  FFFF  002C    ODdsk2  cmpi.w  #-1,ODvalid(A1)  ;is send buffer valid??
11BC  6706                        beq.s   ODdsk3           ;yes, go on
11BE  3E3C  0092                  move.w  #NoBufr,D7       ;set IOresult to "no buffer" error
11C2  602C                        bra.s   ODdsk6           ;return
                                                           ;
11C4  4269  002C          ODdsk3  clr.w   ODvalid(A1)      ;mark send buffer as invalid
11C8  3342  0014                  move.w  D2,SHfmLN(A1)    ;
11CC  0469  0001  0014            subi.w  #1,SHfmLN(A1)    ;subtract one for the return code
11D2  D1FC  0000  0001            adda.l  #1,A0            ;inc buffer pointer past return code
11D8  0C69  0004  0012            cmpi.w  #4,SHtoLN(A1)    ;are we doing a longcmd?
11DE  6206                        bhi.s   ODdsk4           ;yes
11E0  6100  01E8                  bsr     ShortCmds        ;no
11E4  6004                        bra.s   ODdsk5           ;
11E6  6100  016E          ODdsk4  bsr     LongCmds         ;
11EA  91FC  0000  0001    ODdsk5  suba.l  #1,A0            ;dec buffer pointer past return code
```

```
11F0                  2429* ODdsk6  ;
                      2430*         ; the return code must be loaded explicitly since it comes from
                      2431*         ; the header portion of the results packet....
                      2432*         ;
11F0  1087            2433*         move.b  D7,(A0)         ;stuff return code in read buffer
                      2434*                                 ;
11F2  4CDF  0707      2435* ODdsk9  movem.l (sp)+,d0-d2/a0-a2;Restore registers
                    . 2436* ; ----  move.b  d7,CPdiskRC.w   ;Save current disk return code
11F6  4887            2437*         ext.w   d7              ;Set return condition code
11F8  4E75            2438*         rts                     ;Return
                      2439*
```

```
                  ;
                  ; StrobIt -- Strobe command address to Transporter
                  ;
                  ;       Enter:  A2 = command address
                  ;
                  ;       Exit.   D7 = Transporter strobe status
                  ;
                  ;               EQ = successful
                  ;               NE = Transporter not responding
                  ;
                  ;       All other registers are preserved
                  ;
11FA  48E7  C000    StrobIt movem.l D0-D1,-(sp)     ;save registers
11FE  7E00                  moveq   #0,d7           ,assume no Transporter error
1200  200A                  move.l  A2,D0           ,get command address
1202  E198                  rol.l   #8,D0           ;move command address to msb
                                                    ;
1204  611A                  bsr.s   SBstrob         ;strobe address HI
1206  670C                  beq.s   SBerr           ;
1208  6116                  bsr.s   SBstrob         ;strobe address MED
120A  6708                  beq.s   SBerr           ,
120C  6112                  bsr.s   SBstrob         ;strobe address LO
120E  6704                  beq.s   SBerr           ;
1210  6116                  bsr.s   SBwait          ;wait for Transporter ready
1212  6604                  bne.s   SBexit          ;
                                                    ;
1214  3E3C  0090    SBerr   move.w  #NoTrans,d7     ;no transporter ...
                                                    ;
1218  4CDF  0003    SBexit  movem.l (sp)+,D0-D1     ;restore registers
121C  4A47                  tst.w   d7              ;set return condition code
121E  4E75                  rts                     ;return
                                                    ;
1220  E198          SBstrob rol.l   #8,D0           ;shift address byte to lsb
1222  13C0  0003  0FA1      move.b  D0,StrAdr.L     ;strobe address
1228  323C  FFFE    SBwait  move.w  #TOintvl,D1     ,get timeout interval
122C  0839  0000  0003 SBW1 btst    #0,RdyAdr.L     ,is transporter ready?
1232  0F7F
1234  6600  0006            bon     SBWexit         ;yes, return
1238  51C9  FFF2            dbra    D1,SBW1         ;repeat until transporter ready
123C  4E75          SBWexit rts                     ;return
```

```
                        ;
                        ; SetGo -- set up a receive for the 'GO' packet
                        ;
123E  337C  0002  0020  SetGo   move.w  #2,TCdtaLN(A1)              ;2 bytes of data
1244  4229  0022                clr.b   TChdrLN(A1)                 ;no header
1248  4869  0016                pea     GData(A1)                   ;get address of data area
                        ; ----  move.l  (SP)+,ODdw(A1)              ;
                        ; ----  move.b  ODdwhi(A1),TCdADhi(A1)      ;load data buffer address
                        ; ----  move.w  ODdwlo(A1),TCdADlo(A1)      ;*
124C  235F  001C                move.l  (SP)+,TCdADhi-1(A1)         ;same as above -- TCsock destroyed
1250  6010                      bra.s   SetGo1                      ;
                        ;
                        ; SetRecv -- set up a receive for the disk results and read data
                        ;            returns result in D0
                        ;
1252                    SetRecv
                        ; ----  move.l  A0,ODdw(A1)                 ;load data buffer address
                        ; ----  move.b  ODdwhi(A1),TCdADhi(A1)      ;*
                        ; ----  move.w  ODdwlo(A1),TCdADlo(A1)      ;*
1252  2348  001C                move.l  A0,TCdADhi-1(A1)            ;same as above -- TCsock destroyed
1256  3369  0014  0020          move.w  SHfmLN(A1),TCdtaLN(A1)      ;
125C  137C  0003  0022          move.b  #3,TChdrLN(A1)              ;disk results have a hdr len of 3
                                                                    ;
1262  137C  00FF  0006  SetGo1  move.b  #Waiting,RHpktRC(A1)        ;set result to FF to see it change
                                                                    ;
                                                                    ;prepare the command vector
1268  4869  0006                pea     RHdr(A1)                    ;load result vector address
                        ; ----  move.l  (SP)+,ODdw(A1)              ;*
                        ; ----  move.b  ODdwhi(A1),TCrADhi(A1)      ;*
                        ; ----  move.w  ODdwlo(A1),TCrADlo(A1)      ;*
126C  235F  0018                move.l  (SP)+,TCrADhi-1(A1)         ;same as above -- TCop destroyed
1270  137C  00F0  0018          move.b  #RecvOp,TCop(A1)            ;set up a receive
1276  137C  0080  001C          move.b  #CnstSkt,TCsock(A1)         ;  on socket 80
                                                                    ;
127C  45E9  0018                lea     TCmd(A1),A2                 ;get command address
1280  6100  FF78                bsr     StrobIt                     ;strobe command address to Transporter
1284  6600  00C4                bne     SCerr2                      ;Transporter not responding
1288  303C  FFFE                move.w  #TOintvl,D0                 ;for time out
                                                                    ;
128C  0C29  00FF  0006  SC10    cmpi.b  #Waiting,RHpktRC(A1)        ;
1292  6608                      bne.s   SC12                        ;wait till result changes
1294  51C8  FFF6                dbra    D0,SC10                     ;
1298  6000  0084                bra     SCerr3                      ;timeout error
                                                                    ;
129C  1029  0006        SC12    move.b  RHpktRC(A1),d0              ;get Transporter return code
12A0  0C00  00FE                cmpi.b  #CmdAcpt,d0                 ;was command accepted?
12A4  6D00  009E                blt     SCerr1                      ;no, fatal error
12A8  4240                      clr.w   D0                          ;indicate success
12AA  6000  00A6                bra     SCexit                      ;return
```

```
                         ;
                         ; SndRest -- send the rest of the data (from long command) to the disk server
                         ;            result of call is in D0, 0 = success
                         ;
12AE  1369  0029  001D   SndRest move.b  ODwrAD+1(A1),TCdADhi(A1);load data buffer address
12B4  3369  002A  001E           move.w  ODwrAD+2(A1),TCdADlo(A1);*
12BA  137C  00A0  001C           move.b  #RestSkt,TCSock(A1)     ;
12C0  0469  0004  0012           subi.w  #4,SHtoLN(A1)           ;
12C6  6C04                       bge.s   SC20                    ;
12C8  4269  0012                 clr.w   SHtoLN(A1)              ;result was negative, make it zero
12CC  3369  0012  0020   SC20    move.w  SHtoLN(A1),TCdtaLN(A1)  ;send length - 4 bytes
12D2  4229  0022                 clr.b   TChdrLN(A1)             ;no header for rest packets
12D6  6032                       bra.s   SC40                    ;send it
                         ;
                         ; SndCmds -- send a disk command to the disk server
                         ;            result of call is in D0, 0 = success
                         ;
12D8  4869  0000         SndCmds pea     DCmd(A1)                ;data is the Disk command
                         ; ----  move.l  (SP)+,ODdw(A1)          ;
                         ; ----  move.b  ODdwhi(A1),TCdADhi(A1)  ;load data buffer address
                         ; ----  move.w  ODdwlo(A1),TCdADlo(A1)  ;*
12DC  235F  001C                 move.l  (SP)+,TCdADhi-1(A1)     ;same as above -- TCsock destroyed
12E0  137C  00B0  001C           move.b  #CnstSkt,TCsock(A1)     ;
12E6  0C69  0004  0012           cmpi.w  #4,SHtoLN(A1)           ;are we sending less than 4 bytes
12EC  6C08                       bge.s   SC30                    ;no
12EE  3369  0012  0020           move.w  SHtoLN(A1),TCdtaLN(A1)  ;less
12F4  6006                       bra.s   SC32                    ;
12F6  337C  0004  0020   SC30    move.w  #4,TCdtaLN(A1)          ;disk command is 4 bytes long
12FC  137C  0004  0022   SC32    move.b  #4,TChdrLN(A1)          ;send header is 4 bytes
1302  E05E                       ror.w   #8,d6                   ;set destination host number
1304  1346  0023                 move.b  d6,TCdest(A1)           ;*
1308  E15E                       rol.w   #8,d6                   ;*
                                                                 ;
130A  137C  00FF  000E   SC40    move.b  #Waiting,SHpktRC(A1)    ;set result to FF to see it change
1310  4869  000E                 pea     SHdr(A1)                ;load result vector address
                         ; ----  move.l  (SP)+,ODdw(A1)          ;*
                         ; ----  move.b  ODdwhi(A1),TCrADhi(A1)  ;*
                         ; ----  move.w  ODdwlo(A1),TCrADlo(A1)  ;*
1314  235F  0018                 move.l  (SP)+,TCrADhi-1(A1)     ;same as above -- TCop destroyed
1318  137C  0040  0018           move.b  #SendOp,TCop(A1)        ;sendmsg opcode
131E  45E9  0018         SC50    lea     TCmd(A1),A2             ;get command address
1322  6100  FED6                 bsr     Strobit                 ;strobe command address to Transporter
1326  6622                       bne.s   SCerr2                  ;Transporter not responding
```

```
1328  303C  FFFE         2578*          move.w  #TOintvl,D0             ;for time out
132C  0C29  00FF  000E   2579* SC60     cmpi.b  #Waiting,SHpktRC(A1)    ;
1332  6606               2580*          bne.s   SC70                    ;wait till result changes
1334  51C8  FFF6         2581*          dbra    D0,SC60                 ;
1338  6014               2582*          bra.s   SCerr3                  ;timeout error
                         2583*                                          ;
133A  4240               2584* SC70     clr.w   D0                      ;indicate success
133C  4A29  000E         2585*          tst.b   SHpktRC(A1)             ;did it work?
1340  6D02               2586*          blt.s   SCerr1                  ;no, fatal error
1342  600E               2587*          bra.s   SCexit                  ;return
                         2588*
1344  1029  000E         2589* SCerr1   move.b  SHpktRC(A1),D0          ;get transporter error code
1348  6008               2590*          bra.s   SCexit                  ;return
134A  1007               2591* SCerr2   move.b  D7,D0                   ;no transporter ...
134C  6004               2592*          bra.s   SCexit                  ;return
134E  103C  0091         2593* SCerr3   move.b  #TimeOut,D0             ;time out ...
                         2594*                                          ;
1352  4880               2595* SCexit   ext.w   D0                      ;make return code a word
1354  4E75               2596*          rts                             ;return
                         2597*
```

```
1356                  2599* LongCmds
                      2600* ;
                      2601* ; 1. set up a receive for the GO message
                      2602* ;
1356  6100  FEE6      2603*         bsr     SetGo           ;
135A  6D62            2604*         blt.s   LcmdErr         ;if D0 < 0 then fatal DRW error
                      2605* ;
                      2606* ; 2. send disk command
                      2607* ;
135C  6100  FF7A      2608*         bsr     SndCmds         ;doit
1360  6D5C            2609*         blt.s   LcmdErr         ;if D0 < 0 then fatal DRW error
                      2610* ;
                      2611* ; 3. wait to receive GO
                      2612* ;
1362  1029  0006      2613* Lcmd1   move.b  RHpktRC(A1),d0  ;get Transporter return code
1366  0C00  00FE      2614*         cmpi.b  #CmdAcpt,d0     ;has return code changed?
136A  67F6            2615*         beq.s   Lcmd1           ;no, wait some more
136C  4A00            2616*         tst.b   D0              ;successful receive?
136E  6D4E            2617*         blt.s   LcmdErr         ;no, set error return
                      2618* ;
                      2619* ; 4. validate GO packet
                      2620* ;
1370  0829  0007  0016 2621* Lcmd3  btst    #7,Gdata(A1)    ;
1376  66DE            2622*         bne.s   LongCmds        ;disk server restart
1378  0C69  474F  0016 2623*        cmpi.w  #'GO',Gdata(A1) ;
137E  660A            2624*         bne.s   Lcmd4           ;
1380  1029  0023      2625*         move.b  TCdest(A1),D0   ;
1384  B029  0007      2626*         cmp.b   RHsor(A1),D0    ;did response come from the right place?
1388  6708            2627*         beq.s   Lcmd5           ;
                      2628*                                 ;
138A  6100  FEB2      2629* Lcmd4   bsr     SetGo           ;set up for GO receive again
138E  6D2E            2630*         blt.s   LcmdErr         ;
1390  60D0            2631*         bra.s   Lcmd1           ;
                      2632*
```

```
                        ;
                        ; 5. set up receive for results
                        ;
1392  6100  FEBE        Lcmd5   bsr     SetRecv         ;
1396  6D26                      blt.s   LcmdErr         ;if D0 < 0 then fatal DRW error
                        ;
                        ; 6. send REST
                        ;
1398  6100  FF14                bsr     SndRest         ;
139C  6D20                      blt.s   LcmdErr         ;
                        ;
                        ; 7. wait for results
                        ;
139E  1029  0006        Lcmd6   move.b  RHpktRC(A1),d0  ;get Transporter return code
13A2  0C00  00FE                cmpi.b  #CmdAcpt,d0     ;has return code changed?
13A6  67F6                      beq.s   Lcmd6           ;no, wait some more
13A8  4A00                      tst.b   D0              ;successful receive?
13AA  6D12                      blt.s   LcmdErr         ;no, set error return
                        ;
                        ; 8. validate results
                        ;
13AC  1029  0023        Lcmd7   move.b  TCdest(A1),D0   ;
13B0  B029  0007                cmp.b   RHsor(A1),D0    ;did response come from the right place?
13B4  670C                      beq.s   LcmdOK          ;yes
13B6  6100  FE9A                bsr     SetRecv         ;No, set up receive again....
13BA  6D02                      blt.s   LcmdErr         ;if D0 < 0 then fatal DRW error
13BC  60D4                      bra.s   Lcmd5           ;go back and wait again...
                                                        ;
13BE  1E00              LcmdErr move.b  D0,D7           ;get error return code
13C0  6004                      bra.s   LcmdEx          ;return
                                                        ;
13C2  1E29  000C        LcmdOK  move.b  RHdskRC(A1),D7  ;get disk server return code
                                                        ;
13C6  4887              LcmdEx  ext.w   D7              ;make return code a word
13C8  4E75                      rts                     ;return for ShortCmds or LongCmds
```

```
13CA                     ShortCmds
                         ;
                         ; 1. set up a receive for the results
                         ;
13CA  6100  FE86                 bsr     SetRecv          ;
13CE  6DEE                       blt.s   LcmdErr          ;if D0 < 0 then fatal DRW error
                         ;
                         ; 2. send disk command to disk server
                         ;
13D0  6100  FF06                 bsr     SndCmds          ;doit
13D4  6DE8                       blt.s   LcmdErr          ;if D0 < 0 then fatal DRW error
                         ;
                         ; 3. wait to receive results
                         ;
13D6  2E3C  0004  0000           move.l  #$40000,d7       ;for time out
13DC  1029  0006         Scmd2   move.b  RHpktRC(A1),d0   ;get Transporter return code
13E0  6C0C                       bge.s   Scmd3            ;successful receive, go on
13E2  0C00  00FE                 cmpi.b  #CmdAcpt,d0      ;has return code changed?
13E6  66D6                       bne.s   LcmdErr          ;yes, set error return
13E8  5387                       subq.l  #1,d7            ;time out?
13EA  66F0                       bne.s   Scmd2            ;no, wait some more
13EC  60D0                       bra.s   LcmdErr          ;set error return
                         ;
                         ; 4. validate results
                         ;
13EE  1E29  0007         Scmd3   move.b  RHsor(A1),D7     ;get source of response
13F2  0C29  00FF  0023           cmpi.b  #$FF,TCdest(a1)  ;is this a broadcast?
13F8  6606                       bne.s   Scmd4            ;no, go on
13FA  1347  000C                 move.b  d7,RHdskRC(A1)   ;save disk server number
13FE  60C2                       bra.s   LcmdOk           ;return
                                                          ;
1400  BE29  0023         Scmd4   cmp.b   TCdest(A1),D7    ;did response come from the right place?
1404  67BC                       beq.s   LcmdOK           ;yes
1406  6100  FE4A                 bsr     SetRecv          ;set up receive again...
140A  6DB2                       blt.s   LcmdErr          ;if D0 < 0 then fatal DRW error
140C  60CE                       bra.s   Scmd2            ;go back and wait again...
```

```
                        2709*         include 'CC.PROM.FD'    ;Corvus floppy driver
                        2710* ;
                        2711* ; File: CC.PROM.FD.TEXT
                        2712* ; Date. 19-Jun-82
                        2713* ; By:   Ravi Luthra
                        2714* ,       Keith Ball
                        2715* ;
                        2716*
                        2717* ,
                        2718* ,       Fboot -- Floppy disk boot processing
                        2719* ;
140E 11C0 0700          2720* Fboot   move.b  d0,CPbtslot.w       ;set boot slot number
1412 11C0 0706          2721*         move.b  d0,CPosslot.w       ;set OS slot number
                        2722* ; ----  clr.b   CPbtsrvr.w          ,set boot server number     (already 0)
1416 487A 0034+         2723*         pea     FDblkIO             ;set boot disk blk i/o subr pointer
141A 21DF 0714          2724*         move.l  (sp)+,CPblkio.w     ;*
141E 487A 0044+         2725*         pea     FDsecIO             ;set boot disk sector i/o subr pointer
1422 21DF 0718          2726*         move.l  (sp)+,CPdskio.w     ;*
                        2727* ; ----  moveq   #0,d0               ;                           (already 0)
                        2728* ; ----  move.b  d0,CPossrvr.w       ;set OS server number       (already 0)
                        2729* , ----  move.w  d0,CPosblk+1.w      ;set OS volume block number (already 0)
                        2730* , ----  move.b  d0,CPosdrv.w        ;set OS volume drive number (already 0)
1426 6100 006A          2731*         bsr     FDI8sssd            ;set up floppy constants
142A 6100 00AE          2732*         bsr     FDinit              ;initialize floppy drive
142E 6D1A               2733*         blt.s   Fboot90             ;just return if error
                        2734*                                     ;
1430 207C 0008 E000     2735* Fboot1  move.l  #USRbase,a0         ;get block buffer pointer
1436 7000               2736*         moveq   #0,d0               ;
1438 3200               2737*         move.w  d0,d1               ;
143A 7A32               2738*         moveq   #DskRead,d5         ;get read block function code
                        2739*                                     ;
143C 610E               2740*         bsr.s   FDblkIO             ;read block 1 of boot code
143E 6D0A               2741*         blt.s   Fboot90             ;just return if error
1440 610A               2742*         bsr.s   FDblkIO             ;read block 2 of boot code
1442 6D06               2743*         blt.s   Fboot90             ;just return if error
1444 207C 0008 E000     2744*         move.l  #USRbase,a0         ,get block buffer pointer
                        2745*                                     ;
144A 4E75               2746* Fboot90 rts                         ;return
                        2747*
```

```
                     ;
                     ; PHILOSOPHY:  The user views floppy as a set of 512 byte blocks.
                     ;              The driver then translates this block to track address, sector
                     ;              address, side.
                     ;              It then makes the necessary number of request to read sectors.
                     ;              Partial sectors are not read or written, the excess is ignored.
                     ;              Sector length of an Apple floppy is 256 bytes.
                     ;
                     ; RESTRICTION: Bytes per sector must be exact divisor of 512 (block size).
                     ;              The block address must be less than (2**15)/bytes per sector,
                     ;              so that when sector is formed, it fits in the D3.W.
                     ;
                     ; RESULTS OF SOME COMMANDS:
                     ;
                     ;  1) TRACK REG is incremented by 1 during STEPIN even
                     ;     though there is a seek error because the TRACK adrs
                     ;     requested exceeds the maximum track addres allowed
                     ;
                     ;  2) STEPOUT. the track register is not decremented below 0
                     ;     after TRK00. The seek error bit is set.
                     ;

                     ;
                     ; FDblkIO - Read/Write a Corvus floppy disk block subroutine
                     ;
                     ;       Enter:  A0.L - Buffer address
                     ;               D0.W - Block number
                     ;               D1.W - Drive number
                     ;               D5.W - Read ($32) or Write ($33) command
                     ;
                     ;       Exit:   A0.L - Next free location in buffer
                     ;               D0.W - Updated block number
                     ;               D7.W - IORESULT
                     ;
                     ;       All other registers are preserved.
                     ;
144C  48E7  FE7E     FDblkIO MOVEM.L D0-D6/A1-A6,-(SP)       ;
1450  343C  0200             MOVE.W  #BLKSZ,D2               ;BLOCK SIZE IN BYTES
1454  3600                   MOVE.W  D0,D3                   ;
1456  6100  00D0             BSR     FDrdwr                  ;
145A  4CDF  7E7F             MOVEM.L (SP)+,D0-D6/A1-A6       ;restore registers
145E  5240                   ADDQ.W  #1,D0                   ;INC BASE BLOCK
1460  4A07                   TST.B   D7                      ;set return condition code
1462  4E75                   RTS                             ;return

```

```
                    ;
                    ; FDsctIO - Read/Write a Corvus floppy disk sector
                    ;
                    ;       Enter   A0.L - Buffer address
                    ;               D1.W - Bytes per sector (128 for single density)
                    ;               D3.W - Track number
                    ;               D4.W - Sector number
                    ;               D5.W - Read ($32) or Write ($33) command
                    ;
                    ;       Exit    D7.W - IORESULT
                    ;
                    ;       All other registers are preserved.
                    ;
                    ;DsecIO MOVEM.L D0-D6/A0-A6,-(SP)      ;save registers
                    ;       BSR     FDgetadr               ;set address registers
                    ;                                      ;A1 = ptr to device description info
                    ;                                      ;A2 = ptr to slot controller registers
                    ;                                      ;A3 = ptr to slot static RAM
                    ;       CLR.L   D0                     ;
                    ;       BSR     FDIcmd1                ;turn on motor and setup controller
                    ;       BSR     FDseek                 ;get to track
                    ;       BNE.S   FDsio9                 ;if error, return
                    ;       CMPI.W  #DskWrit,D5            ;only do write if cmd is a write
                    ;       BNE.S   FDsio2                 ;else do a read
                    ;       BSR     FDsecW                 ;
                    ;       BRA.S   FDsio9                 ;
                    ;                                      ;

1464  0C45  0033     FDsecIO CMPI.W  #DskWrit,D5           ;make sure cmd is a read cmd
1468  6604                   BNE.S   FDsio1                ;it is
146A  6000  0378              BRA     FDEopcd               ;it isn't, return error
                                                           ;
146E  48E7  FEFE     FDsio1  MOVEM.L D0-D6/A0-A6,-(SP)     ;save registers
1472  6100  016C              BSR     FDgetadr              ;set address registers
                                                           ;A1 = ptr to device description info
                                                           ;A2 = ptr to slot controller registers
                                                           ;A3 = ptr to slot static RAM
1476  4280                    CLR.L   D0                    ;
1478  6100  008A              BSR     FDIcmd1               ;turn on motor and setup controller
147C  6100  025C              BSR     FDseek                ;get to track
1480  6604                    BNE.S   FDsio9                ;if error, return

1482  6100  00D6     FDsio2  BSR     FDsecR                ;read sector specified by D4.W
                                                           ;
1486  6100  0066     FDsio9  BSR     FDmtrof               ;turn off motor
148A  4CDF  7F7F              MOVEM.L (SP)+,D0-D6/A0-A6     ;restore registers
148E  4A07                    TST.B   D7                    ;set return condition code
1490  4E75                    RTS                           ;return
```

```
                    2845* ;
                    2846* ; FLOPPY MAIN EQUATES USED BY THE DRIVERS AND FORMAT CODE GROUPS.
                    2847* ; indices to code in static ram - ram is slot dependent $900 for
                    2848* ; $A00 for slot 2, $B00 for slot 3, and $C00 for slot 4.
                    2849* ;
00000900            2850* BASERAM  equ   CPslIram        ,ADDRESS OF FIRST RAM FOR SLOTS
00000000            2851* SVLCMD   equ   0               ,SAVE OF LOCAL COMMAND
                    2852* ;
00000200            2853* BLKSZ    equ   512             ;OS BLOCK SIZE
                    2854* ;
                    2855* ; SLOT BASE ADDRESSES
                    2856* ;
                    2857* ; The floppy controller is inserted into one of the slots.
                    2858* ; Each slot has two address select decodes coming to it.
                    2859* ; One is called NDEVSEL and the other is called SLOTSEL
                    2860* ;
00030001            2861* NDEV1AD      equ     $30001     ,ADRS OF NDEVO (does not exist)
00000020            2862* DEVADOFST    equ     $20        ;OFFSET OF OTHER NDEVS ADRS
                    2863*                                 ;
00030001            2864* SLOT1AD      equ     $30001     ,ADRS OF slot 0 (does not exist)
00000200            2865* SLTADOFST    equ     $200       ,ADRS OFST FOR OTHER SLOTS
                    2866*                                 ;
00030A01            2867* SLTSTAD      equ     $30A01     ,SLOT STATUS ADRS
                    2868*                                 ;
00000000            2869* NNMI1        equ     0          ,BIT POSITION FOR EACH STATUS BIT
00000001            2870* NNMI2        equ     1          ;
00000002            2871* NNMI3        equ     2          ;
00000003            2872* NNMI4        equ     3          ;
                    2873*                                 ;
00000004            2874* NIRQ1        equ     4          ;
00000005            2875* NIRQ2        equ     5          ;
00000006            2876* NIRQ3        equ     6          ;
00000007            2877* NIRQ4        equ     7          ;
                    2878*
```

```
                  ;
                  ; Bytes per sector
                  ;
00000080          BPS8ISD equ     128     ;Single density 8"
00000100          BPS8IDD equ     256     ;Double density 8"
00000100          BPS5ISD equ     256     ;Apple 5 1/4" floppy
                  ;
                  ; Sectors per track
                  ;
0000001A          SCPT8SD equ     26      ;Single density 8"
0000001A          SCPT8DD equ     26      ;Double density 8"
00000010          SCPT5SD equ     16      ;Apple 5 1/4" floppy
                  ;
                  ; Tracks per side
                  ;
0000004D          TKPS8SD equ     77      ;Single density 8"
0000004D          TKPS8DD equ     77      ;Double density 8"
00000023          TKPS5SD equ     35      ;Apple 5 1/4" floppy
                  ;
                  ; Number of blocks per disk
                  ;
000001F4          NBLK8SD equ     500     ;Single density 8" single sided
000003E9          NBLK8DD equ     1001    ;Double density 8" single sided
00000118          NBLK5SD equ     280     ;Apple 5 1/4" floppy
                  ;
                  ; Error return codes
                  ;
00000000          RGOOD     equ   0       ;disk accss successful
FFFFFFFF          RBDBLK    equ   -1      ;Block requested is out of range
FFFFFFFE          RBDUNT    equ   -2      ;bad unit number or driver not implemented
FFFFFFFD          RBDOPCO   equ   -3      ;Requested unit I/O function is not valid
FFFFFFFC          RHWRERR   equ   -4      ;Hardware error
FFFFFFFB          RLOSTDEV  equ   -5      ;Lost device. i.e. device went offline
                                          ;
FFFFFFF0          RWRPROT   equ   -16     ;the unit is write protected
FFFFFFEF          RSEEKERR  equ   -17     ;SEEK Error
FFFFFFEE          RBUSY     equ   -18     ;device busy
FFFFFFED          RRNF      equ   -19     ;record not found - maybe disk is bad
FFFFFFEC          RNOTRDY   equ   -20     ;device not ready
FFFFFFC0          RERRUNOWN equ   -64     ;error origin unknown
```

```
                  ;
                  , This section conatins the equates for Floppy Disk cont FD1793
                  ; All references are w.r.t NDEV1AD in A2
                  ; Prefix of 'L' means that this reference is to the 'LOCAL'
                  ; logic on the disk controller board
                  ; Prefix of FDC means that this referenc is to the Floppy disk controller
                  ,
                  , Local disk controller board equates
                  ;
00000000          LSTRR     equ   0       ,index to the local Status reg
00000000          LCMDR     equ   0       ;index to the iocal command reg
                                          ;
00000000          LSDRQ     equ   0       ,BIT 0   =1 DRQ
00000001          LSINT     equ   1       ,INTERRUPT REQUEST
00000004          LS1SD2SD  equ   4       ;=0 if 2 sided , =1 if one sided
00000005          LS8INMIN  equ   5       ;=1 if 8 inch , =0 if min
00000006          LSDSKCHG  equ   6       ,=0 if disk changd, 1 if not
00000007          LSFMMFM   equ   7       ;=1 if sigi density =0 if double
                  ;
                  ; Command register equates
                  ;
00000000          LCFLPSD1  equ   0       ;=0 if side 0 , =1 if side 1
00000001          LCDE0     equ   1       ;drive selct bit 0
00000004          LCDE1     equ   4       ;drive select bit 1
00000005          LCMOTOROF equ   5       ;=1 if motor to be turned off
00000006          LC8INMIN  equ   6       ;
00000006          LCFLP8IN  equ   6       ;=1 to select 8 in, =0 for 5 1/4
00000007          LCFMMFM   equ   7       ;=1 to select singl density , 0 for dbl
```

```
                    ;
                    ; Floppy disk controllers equates
                    ;
00000010            FDCAD      equ   $10     ;Floppy disk controller base index
                    ;
                    ; address of the internal registers of FDC
                    ;
00000010            FDCCMDR    equ   FDCAD+0 ;ADRS OF FDC COMMAMD REG
00000010            FDCSTRR    equ   FDCAD+0 ;ADRS OF FDC STATUS REG
00000012            FDCTRKR    equ   FDCAD+2 ;ADRS OF FDC TRACK REG
00000014            FDCSECR    equ   FDCAD+4 ;ADRS OF FDC SECTOR REG
00000016            FDCDATR    equ   FDCAD+6 ;ADRS OF FDC DATA REG
                    ;
                    ; Command code equates
                    ;
00000000            CRESTORE   equ   0       ;0 0 0 0 H V R1 R0
00000010            CSEEK      equ   $10     ;0 0 0 1 h v r1 r0
00000020            CSTEP      equ   $20     ;0 0 1 U h v r1 r0
00000040            CSTEPIN    equ   $40     ;0 1 0 U h v r1 r0
00000060            CSTEPOUT   equ   $60     ;0 1 1 U h v r1 r0
                    ;
                    ; Type II commands
                    ;
00000080            CRDSEC     equ   $80     ;1 0 0 m F2 E F1 0
000000A0            CWRSEC     equ   $A0     ;1 0 1 m F2 E F1 0
                    ;
                    ;TYpe III commands
                    ;
000000C0            CRDAM      equ   $C0     ;1 1 0 0 0  E 0  0
000000E0            CRDTRK     equ   $E0     ;1 1 1 0 0  E 0  0
000000F0            CWRTRK     equ   $F0     ;1 1 1 1 0  E 0  0
000000D0            CFRCINT    equ   $D0     ;1 1 0 1 I1 I2 I3 I4
                    ;
                    ; FLAGS equates -- all flags have prefix of F
                    ;
00000008            FHld       equ   $8      ;if =1 load head in the beginning
                                             ;if =0  unload head in  beginning
00000004            FVerify    equ   $4      ;if =1 verify destination trk else not
00000010            FUpdttrk   equ   $10     ;if =1 update TRK reg after each STEP
00000000            FSTPRT3ms  equ   $0      ;step rate = 3 milliseconds
00000001            FSTPRT6ms  equ   $1      ;step rate = 6 milliseconds
00000002            FSTPRT10ms equ   $2      ;step rate = 10 milliseconds
00000003            FSTPRT15ms equ   $3      ;step rate = 15 milliseconds
                                             ;
00000010            FMPS       equ   $10     ;M=1 if multiple sectors else =0
00000004            FDLY       equ   $4      ;E=1 if internal dly of 15 ms =0 no dly
00000008            FSDCPM     equ   $8      ;F2=0 compare with side 0,=1 with side 1
00000002            FSDCMPEN   equ   $2      ;F1=1 enable side compare, =0 disable cmp
                                             ;
00000001            FINTRDY    equ   $1      ;not ready to ready
00000002            FINTNRDY   equ   $2      ;ready to not ready
00000004            FINTIDXP   equ   $4      ;interrupt on index pulse
00000008            FINTIMM    equ   $8      ;terminate command immediately and intrpt
```

```
                    ;
                    ; Status register equates -- all status reg bits have prefix of S
                    ;
                    ;name        bit position
                    ;
00000000            SBUSY     equ   0       ;S0  busy
                                            ;
00000001            SINDEX    equ   1       ;S1  index pulse encountered
00000001            SDRQ      equ   1       ;S1  data request
                                            ;
00000002            STRK0     equ   2       ;S2  track 00
00000002            SDTOVER   equ   2       ;S2  data over run
00000002            SDTUNDR   equ   2       ;S2  data under run
                                            ;
00000003            SCRCERR   equ   3       ;S3  crc error
                                            ;
00000004            SSEEKERR  equ   4       ;S4  seek error
00000004            SRNF      equ   4       ;S4  record not found
                                            ;
00000005            SHDLDD    equ   5       ;S5  head loaded
00000005            SRECTYP   equ   5       ;S5  record type
00000005            SWRFAULT  equ   5       ;S5  write fault
                                            ;
00000006            SWRPROT   equ   6       ;S6  floppy write protected
                                            ;
00000007            SNOTRDY   equ   7       ;S7  floppy not ready
```

```
                   ;
                   ; FDI8sssd -- Set up constants for Corvus 8" single side single density
                   ;
1492               FDI8sssd
1492  6100  014C           bsr      FDgetadr              ;set address registers
                                                          ;A1 = ptr to device description info
                                                          ;A2 = ptr to slot controller registers
                                                          ;A3 = ptr to slot static RAM
1496  337C  01F4  0734     move.w   #NBLK8SD,CPfdvsz(A1)  ;set device size in blocks
149C  337C  0080  0736     move.w   #BPS8ISD,CPfbps(A1)   ;set bytes per sector
14A2  137C  001A  0738     move.b   #SCPT8SD,CPfspt(A1)   ;set sectors per track
14A8  137C  004D  0739     move.b   #TKPS8SD,CPftps(A1)   ;set tracks per side
14AE  601C                 bra.s    FDI8ss                ;set other values and return

                   ;
                   ; FDI8ssdd -- Set up constants for Corvus 8" single side double density
                   ;
14B0               FDI8ssdd
14B0  6100  012E           bsr      FDgetadr              ;set address registers
                                                          ;A1 = ptr to device description info
                                                          ;A2 = ptr to slot controller registers
                                                          ;A3 = ptr to slot static RAM
14B4  337C  03E9  0734     move.w   #NBLK8DD,CPfdvss(A1)  ;set device size in blocks
14BA  337C  0100  0736     move.w   #BPS8IDD,CPfbps(A1)   ;set bytes per sector
14C0  137C  001A  0738     move.b   #SCPT8DD,CPfspt(A1)   ;set sectors per track
14C6  137C  004D  0739     move.b   #TKPS8DD,CPftps(A1)   ;set tracks per side
14CC  137C  0001  073A FDI8ss move.b #1,CPfspd(A1)       ;set sides per disk
14D2  137C  0001  073B     move.b   #1,CPfofst(A1)        ;set first track offset
14D8  4E75                 rts                            ;return

                   ;
                   ; FDinit -- Initialize Corvus floppy disk drive
                   ;
14DA  6100  0104   FDinit  BSR      FDgetadr              ;set address registers
                                                          ;A1 = ptr to device description info
                                                          ;A2 = ptr to slot controller registers
                                                          ;A3 = ptr to slot static RAM
14DE  4280                 CLR.L    D0                    ;
14E0  7201                 MOVEQ    #1,D1                 ;A FAKE SECTOR LENGTH
14E2  6120                 BSR.S    FD1cmd1               ;turn on motor and setup controller
14E4  6100  0230           BSR      FDrst                 ;restore to track 0
14E8  6104                 BSR.S    FDmtrof               ;turn motor off
14EA  4A07                 TST.B    D7                    ;set return condition code
14EC  4E75                 RTS                            ;return

```

```
                        ;
                        ;  D0.W  -- FREE
                        ;  D1.W  -- FREE
                        ;  D2.W  -- BYTE CNT
                        ;  D3.W  -- BASE BLK  ADRS
                        ;  D4.W  -- FREE
                        ;  D5.W  -- USER CMD
                        ;  D6.W  -- FREE
                        ;
                        ;  A0.L  -- USER BUFFER ADDRESS
                        ;  A1.L  -- DEVICE DESCRIPTION AREA BASE ADDRESS
                        ;  A2.L  -- FLOPPY CONTROLLER BASE ADDRESS
                        ;  A3.L  -- STATIC RAM BASE ADDRESS
                        ;
                        ;
                        ; FDmtrof -- Turn motor off
                        ;
14EE 182B 0000          FDmtrof move.b  SVLCMD(A3),D4          ;get current local command
14F2 08C4 0005                  bset    #LCMOTOROF,D4          ;set motor off flag
14F6 1544 0000                  move.b  D4,LCMDR(A2)           ;move command to command register
14FA 1744 0000                  move.b  D4,SVLCMD(A3)          ;save current local command
14FE 4E75                       rts                            ;return

                        ;
                        ; FDlcmd -- GET THE LOCAL COMMAND FOR THIS DRIVE INTO REGISTER D0.B
                        ;           RETURNS WITH D7 CLEAR LONG
                        ;
1500 6100 010A          FDlcmd  bsr     FDclcTS         ;Calc first side trk sec
                                ;
                                ; form a local command in D0
                                ; Entry used for Read/Write a sector
                                ;
1504 0880 0005          FDlcmd1 bclr    #LCMOTOROF,D0   ;clear motor off bit
1508 08C0 0006                  bset    #LCFLP8IN,d0    ;indicate 8 inch flp
150C 08C0 0007                  bset    #LCFMMFM,d0     ;indicate 8 inch flp
1510 1E29 0708                  move.b  CPosdrv(a1),d7  ;sel drive
1514 E25F                       ror.w   #1,d7           ;note DE0 is B0
1516 E50F                       lsl.b   #2,d7           ;DE1 is B4
1518 E55F                       rol.w   #2,d7           ;
151A 8007                       or.b    d7,d0           ;set into d0
151C 1740 0000                  move.b  d0,SVLCMD(a3)   ;save then command
1520 1540 0000                  move.b  d0,LCMDR(a2)    ;set local command register
1524 4287                       clr.l   d7              ;clear error register
1526 4E75                       rts                     ;return
```

```
1528 6100 00B6        3127* FDrdwr  BSR     FDgetadr          ;set address registers
                      3128*                                   ;A1 = pointer to device description info
                      3129*                                   ;A2 = pointer to slot controller registers
                      3130*                                   ;A3 = pointer to slot static RAM
152C 4A43             3131*         TST.W   D3                ;test base block
152E 6B00 02A8        3132*         BMI     FDEblck           ;jump if first blk rqstd is invalid
1532 B669 0734        3133*         cmp.w   CPfdvsz(A1),d3    ;is it in limit
1536 6C00 02A0        3134*         bge     FDEblck           ;jump final block exceeds max
153A 61C4             3135*         BSR.S   FDlcmd            ;set local command register
                      3136*
                      3137* ;---------------------------------------------------------------
                      3138* ;  D0.W  -- LOCAL COMMAND
                      3139* ;  D1.W  -- BYTES PER SEC
                      3140* ;  D2.W  -- WORD CNT
                      3141* ;  D3.W  -- TRACK ADDRESS
                      3142* ;  D4.W  -- SECTOR ADDRESS
                      3143* ;  D5.W  -- USER COMMAND
                      3144* ;  D6.W  -- FREE
                      3145* ;---------------------------------------------------------------
                      3146* ; READS/WRITES ONLY COMPLETE SECTORS
                      3147* ; For the rest of the code:
                      3148* ;          A0       points to the user buffer address
                      3149* ;          A1       points to the beginning of the device
                      3150* ;                   table entry for this volume in D0 at the entry
                      3151* ;          A2       Contains the NDEVICE address of the slot
                      3152* ;                   specified in the device table for this Volume
                      3153* ;          A3       BASE ADDRESS OF LOCAL STATIC RAM
                      3154* ;---------------------------------------------------------------
                      3155*
153C 6100 019C        3156*         bsr     FDseek            ;seek the desired track
1540 6612             3157*         bne.s   FDrdwr9           ;if error, return
                      3158*                                   ;
1542 9441             3159* FDrdwr1 sub.w   d1,d2             ;
1544 6B0E             3160*         bmi.s   FDrdwr9           ;return if no more sectors to process
1546 6112             3161*         bsr.s   FDsecRW           ;process sector
1548 4A07             3162*         tst.b   d7                ;d7 contains result code
154A 6608             3163*         bne.s   FDrdwr9           ;if error, return
154C 6100 010A        3164*         bsr     FDincTS           ;
1550 6602             3165*         BNE.S   FDrdwr9           ;TIMED OUT ERROR
1552 60EE             3166*         bra.s   FDrdwr1           ;process next sector
                      3167*                                   ;
1554 6100 FF98        3168* FDrdwr9 bsr     FDmtrof           ;turn off motor
1558 4E75             3169*         rts                       ;return
                      3170*
```

```
                     ;
                     ; FDsecRW -- Read or write a sector of data to the floppy
                     ;             It transfer the data to/from the adrs in A0
                     ;             from / to the floppy
                     ;
                     ;       Enter:  D5 - DskRead or DskWrit
                     ;
155A                 FDsecRW                             ;fall thru to FDsecR
                     ;        cmp.w   #DskWrit,D5        ;see if it is a Unit write
                     ;        beq.s   FDscRW1
                     ;        bsr.s   FDsecR
                     ;        bra.s   FDscRW9
                     ;FDscRW1 bsr.s   FDsecW
                     ;FDscRW9 rts

                     ;
                     ; FDsecR -- Read one sector of data
                     ;
                     ;       Enter:  bytes per sec ---)  D1.w
                     ;               sector adrs  -----) D4.w
                     ;               buffer adrs  -----) A0
                     ;               floppy must be poitioned on desired track
                     ;
                     ;       Exit:   OS result code ---)  D7
                     ;               data   to the adrs pointed by A0
                     ;
     00000004        FDrcRd  equ     4                  ;read sector retry count
     00000004        FDrcDOr equ     4                  ;data overrun retry count
                                                        ;
155A 48E7 0600       FDsecR  movem.l d5-d6,-(SP)        ;save
155E 3A3C 0004               move.w  #FDrcRd,d5         ;get read sector retry count
1562 2848                    move.l  a0,a4              ;save user buf ptr
                                                        ;
1564 3C3C 0004       FDsecR1 move.w  #FDrcDOr,d6        ;get data overrun retry count
                                                        ;
1568 204C            FDsecR2 move.l  a4,a0              ;get user buf adrs
156A 6100 0144               BSR     FDwRdy             ;WAIT FOR READY OR TIMED OUT
156E 661C                    BNE.S   FDsecR9            ;TIMED OUT ERR
1570 6100 0022               BSR     FDccRd             ;
1574 0807 0002               btst    #SDTOVER,d7        ;is ther  data overrun
1578 57CE FFEE               dbeq    d6,FDsecR2         ;data over run, try again.
157C 0807 0003               btst    #SCRCERR,d7        ;is ther crc error
1580 57CD FFE2               DBEQ    d5,FDsecR1         ; DO UNTIL (no crc error)
                                                        ;   or (no more retries left)
                                                        ;
1584 6100 0206       FDsecR3 BSR     FDrdSta            ;check read status
1588 6702                    beq.S   FDsecR9            ;
158A 204C                    move.l  a4,a0              ;
                                                        ;
158C 4CDF 0060       FDsecR9 movem.l (sp)+,d5-d6        ;
1590 4A07                    tst.b   d7                 ;d7 contains result code
1592 4E75                    rts                        ;return
```

```
 ,
 , FDsecW -- Write one sector of data
 ;
 ;       Enter:  bytes per sec ---> D1.w
 ,               sector adrs  -----> D4.w
 ,               buffer adrs  -----> A0
 ,               floppy must be poitioned on desired track
 ,
 ,        Exit.  OS result code ---> D7
 ,               data   to the adrs pointed by A0
 ;
 ;FDrcWr  equ     4                  ;write sector retry count
 ,FDrcDOw equ     4                  ;data overrun retry count
 ;                                   ;
 ,FDsecW  movem.l d5-d6,-(SP)        ;save registers
 ;        move.w  #FDrcWr,d5         ,get write retry count
 ,        move.l  a0,a4              ;save user buf ptr
 ,                                   ;
 ,FDsecW1 move.w  #FDrcDOw,d6        ;get data overrun retry count
 ,                                   ;
 ,FDsecW2 move.l  a4,a0              ,get user buf adrs
 ;        BSR     FDwRdy             ;WAIT FOR READY OR TIMED OUT
 ,        BNE.S   FDsecW9            ;TIMED OUT ERR
 ,        BSR     FDccWr             ;
 ;        btst    #SDTOVER,d7        ,data overrun?
 ;        dbeq    d6,FDsecW2         ,yes, try again
 ,        btst    #SCRCERR,d7        ;CRC error?
 ,        dbeq    d5,FDsecW1         ;yes, try again
 ,                                   ;
 ,        BSR     FDwrSta            ,check write status
 ,        beq.S   FDsecW9            ,if no error, return
 ,        move.l  a4,a0              ;
 ,                                   ;
 ,FDsecW9 movem.l (sp)+,d5-d6        ;restore registers
 ,        tst.b   d7                 ;d7 contains result code
 ;        rts                        ;return
```

```
                        ; ***** TYPE II COMMANDS *****

                        ;
                        ; FDccRd -- Read one sector of data
                        ,
                        ;       Enter.  bytes per sec ---)  D1.w
                        ;               sector adrs  -----) D4.w
                        ,               buffer adrs  -----) A0
                        ;               floppy must be positioned on desired track
                        ;
                        ;       Exit:   status   -----)  D7
                        ;               data   to the adrs pointed by A0
                        ;
1594  3F01              FDccRd  move.w  d1,-(SP)                ,save
1596  5341                      subq.w  #1,d1                   ,byte count
1598  40E7                      move.w  sr,-(SP)                ,
159A  007C  0700                ori.w   #$0700,sr               ;disable interrupts
159E  1544  0014                move.b  d4,FDCSECR(A2)          ;
15A2  157C  0080  0010          move.b  #CRDSEC,FDCCMDR(A2)     ;issue command
15A8  3E3C  0019                move.w  #25,d7                  ;wait at least 28 micro-second
15AC  51CF  FFFE        FDccRd1 dbf     d7,FDccRd1              ,
                                                                ,
15B0  082A  0000  0010  FDccRd2 btst    #SBUSY,FDCSTRR(A2)      ,see if the ctlr is busy
15B6  67F8                      beq.S   FDccRd2                 ,jump if not busy
                                ,
                                ; ***** TIME CRITICAL LOOP
                                ;
15B8  1E2A  0000        FDccRd3 move.b  LSTRR(A2),D7            ;read status
15BC  0807  0000                btst    #LSDRQ,d7               ,is DRQ there
15C0  6608                      bne.s   FDccRd5                 ,yes , jump
                                                                ,
15C2  0807  0001        FDccRd4 btst    #LSINT,D7               ;is FDC done
15C6  67F0                      beq.s   FDccRd3                 ,no, jump
15C8  600C                      bra.s   FDccRd6                 ,ctlr terminated too soon
                                                                ;
15CA  10EA  0016        FDccRd5 move.b  FDCDATR(A2),(a0)+       ,get a byte from FDC
15CE  51C9  FFE8                dbf     d1,FDccRd3              ,read the remaining bytes
                                ,
                                ; a complete sector has been read.
                                ;
15D2  6100  0194                bsr      FDnRdy                 ,
                                                                ,
15D6  1E2A  0010        FDccRd6 move.b   FDCSTRR(A2),d7         ;read the status
15DA  46DF                      move.w   (sp)+,sr               ;restore SR
15DC  321F                      move.w   (SP)+,d1
15DE  4E75                      rts

```

```
;
; FDccWr -- Write one sector of data
;
;       Enter:  bytes per sec ---> D1.w
;               sector adrs  -----> D4.w
;               buffer adrs  -----> A0
;               floppy must be poitioned on desired track
;
;       Exit:   status   -----> D7
;               data   to the adrs pointed by A0
;
;FDccWr  move.w  d1,-(SP)                ;save
;        subq.w  #1,d1                   ;byte count
;        move.w  sr,-(SP)                ;
;        ori.w   #$0700,sr               ;disable interrupts
;        BSR.S   FDccWr1                 ;CALL time critical FDccWr PART
;        bsr     FDnRdy                  ;
;        move.b  FDCSTRR(A2),d7          ;read the status
;        move.w  (sp)+,sr                ;restore SR
;        move.w  (SP)+,d1                ;
;        rts                             ;return
;                                        ;
;                                        ;
;FDccWr1 move.b  d4,FDCSECR(A2)          ;
;        move.b  #CWRSEC,FDCCMDR(A2)     ;issue command
;                                        ;
;        move.w  #25,d7                  ;wait at least 28 micro-second
;FDccWr2 dbf     d7,FDccWr2              ;
;                                        ;
;FDccWr3 btst    #SBUSY,FDCSTRR(A2)      ;
;        beq.s   FDccWr3                 ;
;        ;
;        ; ***** TIME CRITICAL LOOP
;        ;
;FDccWr4 move.b  LSTRR(A2),D7            ;FDCSTRR(A2),d7 ;read status
;        btst    #LSDRQ,d7               ;is DRQ there
;        bne.s   FDccWr6                 ;yes , jump
;                                        ;
;FDccWr5 btst    #LSINT,D7               ;is FDC done
;        beq.s   FDccWr4                 ;no , jump
;        rts                             ;terminated too soon
;                                        ;
;FDccWr6 move.b  (a0)+,FDCDATR(A2)       ;move a byte to FDC
;        dbf     d1,FDccWr4              ;write the reamining bytes
;        RTS                             ;return
```

```
                        3359* ;
                        3360* ; FDgetadr -- Get pointers to device description info, controller registers,
                        3361* ;             and static RAM for current slot
                        3362* ;
                        3363* ;       Exit:   A1 = pointer to device description info
                        3364* ;               A2 = pointer to controller registers for slot
                        3365* ;               A3 = pointer to static RAM for slot (CPosslot)
                        3366* ;
15E0                    3367* FDgetadr
15E0  227C  0000  0000  3368*         movea.l #0,A1           ;get pointer to device description info
15E6  247C  0003  0001  3369*         move.l  #NDEV1AD,A2     ;get pointer to controller registers
15EC  1E29  0706        3370*         move.b  CPosslot(a1),d7 ;*
15F0  4887              3371*         ext.w   d7              ;*
15F2  CEFC  0020        3372*         mulu.w  #DEVADOFST,d7   ;*
15F6  D5C7              3373*         adda.l  d7,a2           ;*
15F8  47F8  0900        3374*         lea     CPsl1ram.w,A3   ;get pointer to static RAM
15FC  1E29  0706        3375*         move.b  CPosslot(a1),d7 ;*
1600  4887              3376*         ext.w   d7              ;*
1602  5347              3377*         subq.w  #1,d7           ;*
1604  CEFC  0100        3378*         mulu    #$100,d7        ;*
1608  D7C7              3379*         adda.l  d7,A3           ;*
160A  4E75              3380*         rts                     ;return
                        3381*
```

```
                       ;
                       ; FDclcTS -- calculate the Side, Track address and sector
                       ;            address for the First block requested by the user
                       ;
                       ;       Enter.  A1      - device table address
                       ;               D3      - block address
                       ;
                       ;       Exit:   D0.bit  - side flag
                       ;               D3.w    - track address
                       ;               D4.w    - sector address
                       ;
160C 4280              FDclcTS clr.l   d0                 ;
160E 3229 0736                 move.w  CPfbps(A1),D1      ;get bytes per sector
1612 48C3                      ext.l   d3                 ;clear the upper 16 bits of d3
1614 2E3C 0000 0200            move.l  #BLKSZ,d7          ;
161A 8EC1                      divu    d1,d7              ;
161C C6C7                      mulu    d7,d3              ;absolute sector adrs to d3
161E 2E03                      move.l  d3,d7              ;
1620 4244                      clr.w   d4                 ;make sure that upper byte is 00
1622 1829 0738                 move.b  CPfspt(A1),d4      ;
1626 8EC4                      divu    d4,d7              ;
1628 3607                      move.w  d7,d3              ;absolute track adrs to d3
162A 4847                      swap    d7                 ;
162C 1829 073B                 move.b  CPfofst(a1),d4     ;get first sector offset
1630 4884                      ext.w   d4                 ;*
1632 D847                      add.w   d7,d4              ;get sector address
                       ; ----  bclr    #LCFLPSD1,d0       ;select side 0          (already 0)
1634 1E29 0739                 move.b  CPftps(A1),d7      ;get tracks per side
1638 4887                      ext.w   d7                 ;*
163A B647                      cmp.w   d7,d3              ;is track on side 0?
163C 6D16                      blt.s   FDclcT9            ;yes, return
163E 0C29 0001 073A            cmpi.b  #1,CPfspd(a1)      ;is there a side 1?
1644 670A                      beq.s   FDclcT8            ;no, report error
1646 08C0 0000                 bset    #LCFLPSD1,D0       ;select side 1
164A 9647                      sub.w   d7,d3              ;update track address
164C B647                      cmp.w   d7,d3              ;is track on side 1?
164E 6D04                      blt.s   FDclcT9            ;yes, return
                                                          ;
1650 6000 0186         FDclcT8 bra     FDEblck            ;indicate block number error
                                                          ;
1654 4247              FDclcT9 clr.w   d7                 ;indicate no error
1656 4E75                      rts                        ;return
```

```
                        3427* ;
                        3428* ; FDincTS -- update the sector address by one. If it was the last
                        3429* ;            sector on the track then update the Track adrs by
                        3430* ;            one. If it was the last track then update the side
                        3431* ;            in the Local command reg and D0 and restore track to 0.
                        3432* ;
                        3433* ;       Exit:  NE - error and D7 has error code
                        3434* ;              EQ - successful update D7 = 0
                        3435* ;
1658  5244              3436* FDincTS addq.w  #1,d4           ;increment sector number
165A  1E29  0738        3437*         move.b  CPfspt(A1),d7   ;get last sector number + 1
165E  DE29  073B        3438*         add.b   CPfofst(A1),d7  ;*
1662  B807              3439*         cmp.b   d7,d4           ;are we past last sector?
1664  6C04              3440*         bge.s   FDincT1         ;yes, go to next track
1666  4247              3441*         clr.w   d7              ;show successful
1668  6042              3442*         bra.s   FDincT9         ;return
                        3443*                                 ;
166A  1829  073B        3444* FDincT1 move.b  CPfofst(A1),d4  ;reset sector number
166E  4884              3445*         ext.w   D4              ;*
1670  1E29  0739        3446*         move.b  CPftps(A1),d7   ;get tracks per side
1674  0C29  0005  073C  3447*         cmpi.b  #DTa5,CPftyp(a1);is this an Apple floppy drive?
167A  6724              3448*         beq.s   FDincT5         ;yes, process it
                        3449*         ;
                        3450*         ; Corvus 8" floppy drive
                        3451*         ;
167C  B607              3452*         cmp.b   d7,d3           ;are we past last track on side?
167E  6C10              3453*         bge.s   FDincT2         ;yes, go to next side
1680  5243              3454*         addq.w  #1,d3           ;increment track number
1682  612C              3455*         bsr.s   FDwRdy          ;wait for ready or timeout
1684  6626              3456*         bne.s   FDincT9         ;return if timeout error
1686  6100  00BC        3457*         bsr     FDccSin         ;step in 1 track
168A  6100  012A        3458*         bsr     FDskSta         ;check seek status
168E  601C              3459*         bra.s   FDincT9         ;return
                        3460*                                 ;
1690  4243              3461* FDincT2 clr.w   d3              ;reset track number
1692  08C0  0000        3462*         bset    #LCFLPSD1,D0    ;Select side 1
1696  1540  0080        3463*         move.b  D0,LCMDR(A2)    ;*
169A  6100  007A        3464*         bsr     FDrst           ;restore to track 0
169E  600C              3465*         bra.s   FDincT9         ;return
                        3466*         ;
                        3467*         ; Apple 5" floppy drive
                        3468*         ;
16A0  B607              3469* FDincT5 cmp.b   d7,d3           ;are we past last track on side?
16A2  6C00  0134        3470*         bge     FDEblck         ;yes, report error
16A6  5243              3471*         addq.w  #1,d3           ;increment track number
16A8  6100  072E        3472*         bsr     ADccSin         ;step in 1 track
                        3473*                                 ;
16AC  4A47              3474* FDincT9 tst.w   d7              ;set return condition codes
16AE  4E75              3475*         rts                     ;return
                        3476*
```

```
                       ;
                       ; FDwRdy --  WAIT UNTIL FDC SAYS DRIVE IS READY OR
                       ;            TIME OUT (NOT MORE THAN 1 SECOND)
                       ;
                       ;        Exit:   NE = timed out D7 has error result
                       ;                EQ = ready (D7 = 0)
                       ;
     00000004          FDtmoHi equ      4                      ;SHOULD BE AT LEAST 1 SECOND
     00007FFF          FDtmoLo equ      $7FFF                  ;*
                                                               ;
16B0 4287              FDwRdy  CLR.L    D7                     ;
16B2 48A7 0600                 MOVEM.W  D5-D6,-(SP)            ;SAVE D6 AND D5
16B6 3A3C 7FFF                 MOVE.W   #FDtmoLo,D5            ;
16BA 3C3C 0004                 MOVE.W   #FDtmoHi,D6            ;
                                                               ;
16BE 082A 0007 0010    FDwRdy1 BTST     #SNOTRDY,FDCSTRR(A2)   ;IS FLOPPY READY
16C4 57CD FFF8                 DBEQ     D5,FDwRdy1             ;DO UNTIL (FLOPPY READY) OR (TIME OUT)
16C8 57CE FFF4                 DBEQ     D6,FDwRdy1             ;
16CC 6704                      BEQ.S    FDwRdy9                ;DIDN'T TIME OUT
16CE 3E3C FFEC                 MOVE.W   #RNOTRDY,D7            ;
                                                               ;
16D2 4C9F 0060         FDwRdy9 MOVEM.W  (SP)+,D5-D6            ;
16D6 4A47                      TST.W    D7                     ;SET CONDITION CODES - NE MEANS ERROR
16D8 4E75                      RTS                             ;return

```

```
                        ;
                        ; FDseek --
                        ;
                        ; It is assumed that TRACK REG contains the number of the track
                        ; of the current position of the read write Head.
                        ;
                        ,       Enter:  D3.W - Seek track address
                        ;
      00000004          FDrcSk  equ     4               ;seek retry count
                                                        ;
16DA  61D4              FDseek  BSR.S   FDwRdy          ;WAIT FOR READY OR TIMED OUT
16DC  662A                      BNE.S   FDseek9         ;TIMED OUT ERR
16DE  6100  0074                BSR     FDccSk          ;
16E2  6100  00D2                BSR     FDskSta         ,check seek status
16E4  6720                      BEQ.S   FDseek9         ;there is NO error
16E8  3F05                      move.w  d5,-(sp)        ;save d5
16EA  3A3C  0004                move.w  #FDrcSk,D5      ;
                                                        ;
16EE  61C0              FDseek1 BSR.S   FDwRdy          ;WAIT FOR READY OR TIMED OUT
16F0  6614                      BNE.S   FDseek8         ,TIMED OUT ERR
16F2  6140                      BSR.S   FDccRst         ;
16F4  61BA                      BSR.S   FDwRdy          ;WAIT FOR READY OR TIMED OUT
16F6  660E                      BNE.S   FDseek8         ;TIMED OUT ERR
16F8  615A                      BSR.S   FDccSk          ,
16FA  0807  0004                btst    #sseekerr,d7    ;
16FE  57CD  FFEE                DBEQ    d5,FDseek1      ;try until no seek error or
                                                        ;no more retries
1702  6100  00B2                BSR     FDskSta         ;check seek status
                                                        ,
1706  3A1F              FDseek8 move.w  (sp)+,d5        ;
                                                        ;
1708  4A07              FDseek9 TST.B   D7              ;set return condition code
170A  4E75                      RTS                     ,return
```

```
                          ;
                          ; FDrstW -- Restore the floppy to track 0
                          ;            It exits when it has successfully restored the floppy to
                          ;            track 0 or when the retry count has exhausted.
                          ;            Then it calls the FDswSta routine to analyse status.
                          ;            Note W in FDswSta. It looks at WRprot bit of FDCSTRR.
                          ;
170C  48E7  0202          FDrstW  movem.l a6/d6,-(sp)      ;
1710  4DFA  0098+                 lea     FDswSta,a6       ;
1714  6008                        bra.s   FDrst0           ;

                          ;
                          ; FDrst -- same as FDrstW except no W there.
                          ;           It does not look at Write protect status bit in FDCSTRR.
                          ;
1716  48E7  0202          FDrst   movem.l a6/d6,-(sp)      ;
171A  4DFA  009A+                 lea     FDskSta,a6       ;
                                                           ;
171E  3C3C  0004          FDrst0  move.w  #4,d6            ;
                                                           ;
1722  618C                FDrst1  BSR.S   FDwRdy           ;WAIT FOR READY OR TIMED OUT
1724  6608                        BNE.S   FDrst2           ;TIMED OUT ERR
1726  610C                        bsr.s   FDccRst          ;
1728  4E96                        jsr     (a6)             ;
172A  57CE  FFF6                  dbeq    d6,FDrst1        ;do until (successful) or (tried enough)
                                                           ;
172E  4CDF  4040          FDrst2  movem.l (sp)+,a6/d6      ;
1732  4E75                        rts                      ;return

```

```
                          ;
                          ; ***** TYPE I COMMANDS *****
                          ;
      00000007            FDCrst   equ    CRESTORE+FSTPRT15ms+FVERIFY
      00000035            FDCstp   equ    CSTEP+FSTPRT6ms+FVERIFY+FUpdttrk
      00000055            FDCstpIn equ    CSTEPIN+FSTPRT6ms+FVERIFY+FUpdttrk
      00000075            FDCstpOt equ    CSTEPOUT+FSTPRT6ms+FVERIFY+FUpdttrk
      00000015            FDCseek  equ    CSEEK+FSTPRT6ms+FVERIFY
                          ;
                          ; FDccRst -- bring the floppy back to track 00
                          ,            Then set the FDCTRKR = 0
                          ;            Stepping pulses are given at the rate specified in cmd
                          ;
1734 157C 0007 0010       FDccRst move.b  #FDCrst,FDCCMDR(A2)      ;issue command
173A 602C                         bra.s   FDnRdy                   ;wait for not ready
                          ;
                          ; FDccStp -- FDCTRKR+/-1 --> FDCTRKR
                          ,
173C 157C 0035 0010       FDccStp move.b  #FDCstp,FDCCMDR(A2)      ;issue command
1742 6024                         bra.s   FDnRdy                   ,wait for not ready
                          ;
                          ; FDccSin -- FDCTRKR+1 --> FDCTRKR
                          ;
1744 157C 0055 0010       FDccSin move.b  #FDCstpIn,FDCCMDR(A2)    ;issue command
174A 601C                         bra.s   FDnRdy                   ;wait for not ready
                          ;
                          ; FDccSot -- FDCTRKR-1 --> FDCTRKR
                          ;
174C 157C 0075 0010       FDccSot move.b  #FDCstpOt,FDCCMDR(A2)    ,issue command
1752 6014                         bra.s   FDnRdy                   ;wait for not ready
                          ;
                          , FDccSk -- issue a seek command.  If there is a seek error,
                          ;           flip the density flag in D0 and try again
                          ;
1754 3F05                 FDccSk  move.w  d5,-(sp)                 ,save d5
1756 1543 0016                    move.b  D3,FDCDATR(A2)           ;load the desired TRACK adrs
175A 7A02                         moveq   #2,d5                    ,
                                                                   ;
175C 157C 0015 0010       FDccSk1 move.b  #FDCseek,FDCCMDR(A2)     ,issue command
1762 6104                         bsr.s   FDnRdy                   ;wait for not ready
                          ; ----  btst    #sseekerr,d7             ;
                          ; ----  beq     FDccSk2                  ,no seek error
                          ; ----  bchg    #LSFMMFM,d7              ;flip the density bit
                          ; ----  dbf     d5,FDccSk1               , $$$ WE MAY HAVE TO FLIP $$$
                                                                   ,
1764 3A1F                 FDccSk2 move.w  (sp)+,d5                 ;get back d5
1766 4E75                         rts                              ,return
```

```
                         ;
                         ; FDnRdy -- WAIT UNTIL FDC SAYS DRIVE IS NOT BUSY OR TIME OUT
                         ;
                         ;       Exit.   D7 = controller status register (FDCSTRR)
                         ;
1768  48A7  0600         FDnRdy  movem.w d5-D6,-(SP)          ;
176C  3C3C  7FFF                 move.w  #FDtmoLo,d6          ;
1770  3A3C  0002                 move.w  #2,d5                ; add 3*$8000 iterations
                                                              ;
1774  082A  0001  0000   FDnRdy1 btst    #LSINT,LSTRR(a2)     ;is it busy
177A  56CE  FFF8                 dbNE    d6,FDnRdy1           ;DO UNTIL (not busy) or (no more retries)
177E  56CD  FFF4                 dbNE    d5,FDnRdy1           ;DO UNTIL (not busy) or (no more retries)
                                                              ;
1782  4C9F  0060         FDnRdy2 movem.w (SP)+,d5-d6          ;
1786  1E2A  0010                 MOVE.B  FDCSTRR(A2),D7       ;GET CONTROLLER STATUS
178A  4E75                       RTS                          ;return
```

```
                     ;
                     ; Check status subroutines
                     ;
                     ;          Exit:    D7 - IORESULT code
                     ;
178C  1E2A  0010     FDrdSta move.b  FDCSTRR(A2),d7  ;read the status
                     ;          bra.s   FDrwSta          ;
                     ;                                   ,
                     ;FDwrSta move.b  FDCSTRR(A2),d7  ,read the status
                     ;          btst    #SWRPROT,d7      ;
                     ;          bon.s   FDEprot          ;
                     ;          btst    #SWRFAULT,d7     ,
                     ,          bon.s   FDEherr          ;
                                                         ,
1790  0807  0003     FDrwSta btst     #SCRCERR,d7      ,
1794  6642                   bon.s    FDEcrc           ,
1796  0807  0004             btst     #SRNF,d7         ;
179A  6660                   bon.s    FDErnf           ,
                                                       ;
179C  0807  0000     FDrwSt1 btst     #SBUSY,d7        ,
17A0  6660                   bon.s    FDEbusy          ,
17A2  0807  0007             btst     #SNOTRDY,d7      ,
17A6  6666                   bon.s    FDEnrdy          ,
17A8  6028                   bra.s    FDokSta          ,no error, return
                                                       ,
17AA  1E2A  0010     FDswSta move.b   FDCSTRR(a2),d7   ,
17AE  0807  0006             btst     #SWRPROT,d7      ,
17B2  663C                   bon.s    FDEprot          ;
17B4  6004                   bra.s    FDskSt1          ;
                                                       ;
17B6  1E2A  0010     FDskSta move.b   FDCSTRR(a2),d7   ,
                                                       ;
17BA  0807  0004     FDskSt1 btst     #SSEEKERR,d7     ;seek error?
17BE  6636                   bon.s    FDEseek          ;SEEK ERROR IN RSLT CODE
17C0  0807  0003             btst     #SCRCERR,d7      ;
17C4  6612                   bon.s    FDEcrc           ,
17C6  0807  0000             btst     #sbusy,d7        ,
17CA  6636                   bon.s    FDEbusy          ,HARDWARE ERROR
17CC  0807  0007             btst     #SNOTRDY,d7      ;
17D0  663C                   bon.s    FDEnrdy          ;
                                                       ,
17D2  4247           FDokSta clr.w    d7               ;indicate no error
                                                       ,
17D4  4A47           FDerSta tst.w    d7               ,set return condition code
17D6  4E75                   rts                       ;return
```

```
17D8                    3683* FDEcrc                              ;error -- CRC
17D8  3E3C  FFFF        3684* FDEblck move.w  #RBDBLK,d7          ;error -- invalid block number
17DC  60F6              3685*         bra.s   FDerSta             ;set condition code and return
                        3686*                                     ;
17DE  3E3C  FFFE        3687* FDEunit move.w  #RBDUNT,d7          ;error -- invalid unit number
17E2  60F0              3688*         bra.s   FDerSta             ;set condition code and return
                        3689*                                     ;
17E4  3E3C  FFFD        3690* FDEopcd move.w  #RBDOPCO,d7         ;error -- invalid op code
17E8  60EA              3691*         bra.s   FDerSta             ;set condition code and return
                        3692*                                     ;
17EA  3E3C  FFFC        3693* FDEherr move.w  #RHWRERR,d7         ;error -- hardware
17EE  60E4              3694*         bra.s   FDerSta             ;set condition code and return
                        3695*                                     ;
17F0  3E3C  FFF0        3696* FDEprot move.w  #RWRPROT,d7         ;error -- write protect
17F4  60DE              3697*         bra.s   FDerSta             ;set condition code and return
                        3698*                                     ;
17F6  3E3C  FFEF        3699* FDEseek move.w  #RSEEKERR,d7        ;error -- seek
17FA  60D8              3700*         bra.s   FDerSta             ;set condition code and return
                        3701*                                     ;
17FC  3E3C  FFED        3702* FDErnf  move.w  #RRNF,d7            ;error -- record (sector) not found
1800  60D2              3703*         bra.s   FDerSta             ;set condition code and return
                        3704*                                     ;
1802  157C  00D8  0010  3705* FDEbusy move.b  #CFRCINT+FINTIMM,FDCCMDR(A2)
1808  3E3C  FFEE        3706*         move.w  #RBUSY,D7           ;error -- busy
180C  60C6              3707*         bra.s   FDerSta             ;set condition code and return
                        3708*                                     ;
180E  3E3C  FFEC        3709* FDEnrdy move.w  #RNOTRDY,d7         ;error -- not ready
1812  60C0              3710*         bra.s   FDerSta             ;set condition code and return
                        3711*
```

```
                                  include 'CC.PROM.AD'    ;Apple floppy driver
                          ;
                          ; File: CC.PROM.AD.TEXT
                          ; Date: 03-Sep-82
                          ; By:   Ravi Luthra
                          ;       Keith Ball
                          ;

                          ;
                          ;       Aboot -- Apple floppy disk boot processing
                          ;
1814  11C0  0700          Aboot   move.b  d0,CPbtslot.w       ;set boot slot number
1818  11C0  0706                  move.b  d0,CPosslot.w       ;set OS slot number
                          ; ----  clr.b   CPbtsrvr.w          ;set boot server number     (already 0)
181C  487A  0038+                 pea     ADblkIO             ;set boot disk blk i/o subr pointer
1820  21DF  0714                  move.l  (sp)+,CPblkio.w     ;"
1824  487A  0058+                 pea     ADsecIO             ;set boot disk sector i/o subr pointer
1828  21DF  0718                  move.l  (sp)+,CPdskio.w     ;"
                          ; ----  moveq   #0,d0               ;                           (already 0)
                          ; ----  move.b  d0,CPossrvr.w       ;set OS server number       (already 0)
                          ; ----  move.w  d0,CPosblk+1.w      ;set OS volume block number (already 0)
                          ; ----  move.b  d0,CPosdrv.w        ;set OS volume drive number (already 0)
182C  6100  0280                  bsr     ADISsssd            ;set up floppy constants
1830  6100  02C2                  bsr     ADinit              ;initialise floppy drive
1834  6D1E                        blt.s   Aboot90             ;just return if error
                                                              ;
1836  207C  0008  E000    Aboot1  move.l  #USRbase,a0         ;get block buffer pointer
183C  7000                        moveq   #0,d0               ;
183E  3200                        move.w  d0,d1               ;
1840  7A32                        moveq   #DskRead,d5         ;get read block function code
                                                              ;
1842  6112                        bsr.s   ADblkIO             ;read block 1 of boot code
1844  6D0E                        blt.s   Aboot90             ;just return if error
1846  610E                        bsr.s   ADblkIO             ;read block 2 of boot code
1848  6D0A                        blt.s   Aboot90             ;just return if error
184A  207C  0008  E000            move.l  #USRbase,a0         ;get block buffer pointer
1850  D0FC  000C                  adda.w  #12,a0              ;get pointer to boot code
                                                              ;
1854  4E75                Aboot90 rts                         ;return

```

```
                 ;
                 ; PHILOSOPHY: The user views floppy as a set of 512 byte blocks.
                 ;             The driver then translates this block to track address, sector
                 ;             address, side.
                 ;             It then makes the necessary number of request to read sectors.
                 ;             Partial sectors are not read or written, the excess is ignored.
                 ;             Sector length of an Apple floppy is 256 bytes.
                 ;
                 ; RESTRICTION. Bytes per sector must be exact divisor of 512 (block size).
                 ;             The block address must be less than (2**15)/bytes per sector,
                 ;             so that when sector is formed, it fits in the D3.W.
                 ;

                 ;
                 ; ADblkIO - Read/Write an Apple floppy disk block subroutine
                 ;
                 ;       Enter:  A0.L - Buffer address
                 ;               D0.W - Block number
                 ;               D1.W - Drive number
                 ;               D5.W - Read ($32) or Write ($33) command
                 ;
                 ;       Exit:   A0.L - Next free location in buffer
                 ;               D0.W - Updated block number
                 ;               D7.W - IORESULT
                 ;
                 ;       All other registers are preserved.
                 ;
1856 48E7 FE7E   ADblkIO MOVEM.L D0-D6/A1-A6,-(sp)     ;save registers
185A 343C 0200           MOVE.W  #BLKSZ,D2             ;BLOCK SIZE IN BYTES
185E 3600                MOVE.W  D0,D3                 ;
1860 0C45 0033           CMPI.W  #DskWrit,D5           ;
1864 6606                BNE.S   ADbio1                ;
1866 6100 FF7C           BSR     FDEopcd               ;only do reads
186A 6008                BRA.S   ADbio9                ;return
                                                       ;
186C 6100 02A2   ADbio1  BSR     ADrdwr                ;
1870 6100 0292          BSR     ADmtrof               ;TURN THE MOTOR OFF
                                                       ;
1874 4CDF 7E7F   ADbio9  MOVEM.L (sp)+,D0-D6/A1-A6     ;
1878 5240                ADDQ.W  #1,D0                 ;INC BASE BLOCK
187A 4A47                TST.W   D7                    ;SET CONDITION CODES *KB 8/24/82*
187C 4E75                RTS                           ;return
```

```
                     ;
                     ; index into static RAM to routines
                     ;
00000014             SRAMln4 equ     ADrd4E-ADrd4B   ;
0000001C             SRAMln5 equ     ADrd5E-ADrd5B   ;
00000020             SRAMln6 equ     ADrd6E-ADrd6B   ;
00000022             SRAMlnw equ     ADwaitE-ADwaitB ;
                                                     ;
                     ;RAMrd4 equ     0               ;ADrd4 routine   (READ)
                     ;RAMrd5 equ     SRAMrd4+SRAMln4 ;ADrd5 routine (READ)
                     ;RAMrd6 equ     SRAMrd5+SRAMln5 ;ADrd61 routine  (READ)
                     ;RAMwt  equ     SRAMrd6+SRAMln6 ;ADwaitB routine (SEEK)
                     ;RAMend equ     SRAMwt+SRAMlnw  ;Must be <= $100
                                                     ;
00000000             SRAMrd4 equ     $00             ;ADrd4 routine   (READ)
00000040             SRAMrd5 equ     $40             ;ADrd5 routine (READ)
00000080             SRAMrd6 equ     $80             ;ADrd61 routine  (READ)
000000C0             SRAMwt  equ     $C0             ;ADwaitB routine (SEEK)
                     ;
                     ; PRENIB16 routine niblises the user data into 6/2 format
                     ; and places in Nbuf1 and Nbuf2 buffers
                     ;
00000D00             NBUF1   equ     CPiobuf         ;MUST BE AT LEAST 258 BYTES
00000E02             NBUF2   equ     NBUF1+258       ;$60 HEX BYTES
                     ;
                     ; AREA FOR THE TEMPORARIES AND OTHER VARIABLES
                     ;
00000E62             APL5VAR equ     NBUF2+$60
                     ;
                     ; 1) for 'ADccSK' subroutine
                     ;
00000000             TRKCNT  equ     0
00000001             PRIOR   equ     1
00000002             TRKN    equ     2
00000003             CURTRK  equ     3
                     ;
                     ; 2) for 'ADccRd'
                     ;
00000004             csumrd  equ     CURTRK+1
                     ;
                     ; 3) FOR 'ADrdad'
                     ;
00000E82             AMBUF   equ     APL5VAR+$20     ;NEED AT MAX. 32 BYTES FOR TEMP
                     ;
                     ; Indices to fields in AMBUF
                     ;
00000000             AMvol   equ     0               ;volume name in adr mark
00000001             AMtrk   equ     AMvol+1         ;track # in adr mark
00000002             AMsec   equ     AMtrk+1         ;sector # in adr mark
00000003             AMchksm equ     AMsec+1         ;check sum in adr mark

```

```
                          3883* ;
                          3884* ; NIBL table
                          3885* ;
                          3886* ; NIBL buffer is used to translate most significant 6 bits of a byte into
                          3887* ; 8 bits of disk data.  PRENIBLE16 routine partitioned 8 bits of user data
                          3888* ; into 6/2 format.  The 6 bit data is left justified, thus every 4th entry
                          3889* ; of the table is used in nibilizing.  Every fourth entry contains the nible
                          3890* ; code for a 6 bit left justified data nibble.  6 bits of data can take a
                          3891* ; value from 0 to 3F.
                          3892* ;
                          3893* ; Interspersed in them is the denible code for the lower 2 bits of data byte
                          3894* ; offset w.r.t. DNIBL2, DNIBL3, DNIBL4.  These values are used to get the
                          3895* ; least significant 2 bits of user data while reading data from floppy.
                          3896* ;
18AA  00                  3897*         DATA.B  0        ;EVEN OUT
18AB  00                  3898* DNIBL2  DATA.B  0
18AC  00                  3899* DNIBL3  DATA.B  0
18AD  00                  3900* DNIBL4  DATA.B  0
                          3901*
18AE  96 02 00 00 97 01 3902* NIBL    data.b  $96,2,0,0,$97,1,0,0,$9A,3,0,0,$9B,0,2,0
18B4  00 00 9A 03 00 00
18BA  9B 00 02 00
18BE  9D 02 02 00 9E 01 3903*         data.b  $9D,2,2,0,$9E,1,2,0,$9F,3,2,0,$A6,0,1,0
18C4  02 00 9F 03 02 00
18CA  A6 00 01 00
18CE  A7 02 01 00 AB 01 3904*         data.b  $A7,2,1,0,$AB,1,1,0,$AC,3,1,0,$AD,0,3,0
18D4  01 00 AC 03 01 00
18DA  AD 00 03 00
18DE  AE 02 03 00 AF 01 3905*         data.b  $AE,2,3,0,$AF,1,3,0,$B2,3,3,0,$B3,0,0,2
18E4  03 00 B2 03 03 00
18EA  B3 00 00 02
18EE  B4 02 00 02 B5 01 3906*         data.b  $B4,2,0,2,$B5,1,0,2,$B6,3,0,2,$B7,0,2,2
18F4  00 02 B6 03 00 02
18FA  B7 00 02 02
18FE  B9 02 02 02 BA 01 3907*         data.b  $B9,2,2,2,$BA,1,2,2,$BB,3,2,2,$BC,0,1,2
1904  02 02 BB 03 02 02
190A  BC 00 01 02
190E  BD 02 01 02 BE 01 3908*         data.b  $BD,2,1,2,$BE,1,1,2,$BF,3,1,2,$CB,0,3,2
1914  01 02 BF 03 01 02
191A  CB 00 03 02
191E  CD 02 03 02 CE 01 3909*         data.b  $CD,2,3,2,$CE,1,3,2,$CF,3,3,2,$D3,0,0,1
1924  03 02 CF 03 03 02
192A  D3 00 00 01
192E  D6 02 00 01 D7 01 3910*         data.b  $D6,2,0,1,$D7,1,0,1,$D9,3,0,1,$DA,0,2,1
1934  00 01 D9 03 00 01
193A  DA 00 02 01
193E  DB 02 02 01 DC 01 3911*         data.b  $DB,2,2,1,$DC,1,2,1,$DD,3,2,1,$DE,0,1,1
1944  02 01 DD 03 02 01
194A  DE 00 01 01
194E  DF 02 01 01 E5 01 3912*         data.b  $DF,2,1,1,$E5,1,1,1,$E6,3,1,1,$E7,0,3,1
1954  01 01 E6 03 01 01
195A  E7 00 03 01
195E  E9 02 03 01 EA 01 3913*         data.b  $E9,2,3,1,$EA,1,3,1,$EB,3,3,1,$EC,0,0,3
1964  03 01 EB 03 03 01
```

```
196A  EC 00 00 03
196E  ED 02 00 03 EE 01 3914*          data.b  $ED,2,0,3,$EE,1,0,3,$EF,3,0,3,$F2,0,2,3
1974  00 03 EF 03 00 03
197A  F2 00 02 03
197E  F3 02 02 03 F4 01 3915*          data.b  $F3,2,2,3,$F4,1,2,3,$F5,3,2,3,$F6,0,1,3
1984  02 03 F5 03 02 03
198A  F6 00 01 03
198E  F7 02 01 03 F9 01 3916*          data.b  $F7,2,1,3,$F9,1,1,3,$FA,3,1,3,$FB,0,3,3
1994  01 03 FA 03 01 03
199A  FB 00 03 03
199E  FC 02 03 03 FD 01 3917*          data.b  $FC,2,3,3,$FD,1,3,3,$FE,3,3,3,$FF,0,2,3
19A4  03 03 FE 03 03 03
19AA  FF 00 02 03
                              3918*
                              3919* ;
                              3920* ; NIBL table
                              3921* ;
19AE  0000  0000  0000  3922* DNIBL   data.w  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
19B4  0000  0000  0000
19BA  0000  0000  0000
19C0  0000  0000  0000
19C6  0000  0000  0000
19CC  0000
19CE  0000  0000  0000  3923*         data.w  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
19D4  0000  0000  0000
19DA  0000  0000  0000
19E0  0000  0000  0000
19E6  0000  0000  0000
19EC  0000
19EE  0000  0000  0000  3924*         data.w  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
19F4  0000  0000  0000
19FA  0000  0000  0000
1A00  0000  0000  0000
1A06  0000  0000  0000
1A0C  0000
1A0E  0000  0000  0000  3925*         data.w  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1A14  0000  0000  0000
1A1A  0000  0000  0000
1A20  0000  0000  0000
1A26  0000  0000  0000
1A2C  0000
1A2E  0000  0000  0000  3926*         data.w  0,0,0,0,0,0,0,0,0,0,0
1A34  0000  0000  0000
1A3A  0000  0000  0000
1A40  0000  0000
                              3927*              ,96 HEX BYTES DISPLACEMENT = 150 DECIMAL
1A44  00 04 98 99 08 0C 3928*         data.b  000,004,$98,$99,008,$0C,$9C,$10
1A4A  9C 10
1A4C  14 18 A0 A1 A2 A3 3929*         data.b  $14,$18,$A0,$A1,$A2,$A3,$A4,$A5
1A52  A4 A5
1A54  1C 20 A8 A9 AA 24 3930*         data.b  $1C,$20,$A8,$A9,$AA,$24,$28,$2C
1A5A  28 2C
1A5C  30 34 B0 B1 38 3C 3931*         data.b  $30,$34,$B0,$B1,$38,$3C,$40,$44
1A62  40 44
```

```
1A64  48 4C B8 50 54 58 3932*        data.b  $48,$4C,$B8,$50,$54,$58,$5C,$60
1A6A  5C 60
1A6C  64 68 C0 C1 C2 C3 3933*        data.b  $64,$68,$C0,$C1,$C2,$C3,$C4,$C5
1A72  C4 C5
1A74  C6 C7 C8 C9 CA 6C 3934*        data.b  $C6,$C7,$C8,$C9,$CA,$6C,$CC,$70
1A7A  CC 70
1A7C  74 78 D0 D1 D2 7C 3935*        data.b  $74,$78,$D0,$D1,$D2,$7C,$D4,$D5
1A82  D4 D5
1A84  80 84 D8 88 8C 90 3936*        data.b  $80,$84,$D8,$88,$8C,$90,$94,$98
1A8A  94 98
1A8C  9C A0 E0 E1 E2 E3 3937*        data.b  $9C,$A0,$E0,$E1,$E2,$E3,$E4,$A4
1A92  E4 A4
1A94  A8 AC E8 B0 B4 B8 3938*        data.b  $A8,$AC,$E8,$B0,$B4,$B8,$BC,$C0
1A9A  BC C0
1A9C  C4 C8 F0 F1 CC D0 3939*        data.b  $C4,$C8,$F0,$F1,$CC,$D0,$D4,$D8
1AA2  D4 D8
1AA4  DC E0 F8 E4 E8 EC 3940*        data.b  $DC,$E0,$F8,$E4,$E8,$EC,$F0,$F4
1AAA  F0 F4
1AAC  F8 FC             3941*        data.b  $F8,$FC
                        3942*                ;it ends on even boundary
```

```
                  ;
                  ; Apple floppy controller equates
                  ;
                  ; These are the index values corresponding to the base address of the slot
                  ; calculated by FDgetadr subroutine
                  ;
                  ; Commands to the floppy are issued by setting or resetting the bits in the
                  ; addressable latch of the type 74LS259.
                  ; These  bits can be set or reset by making a READ/WRITE reference to these
                  ; addresses.  In general, we make a read reference to the addresses assigned
                  ; to these bits (indexed w.r.t Device Select addresses).
                  ; For write operations, these bits are set/reset by making a write reference.
                  ;
                  ; NOTE:
                  ;     The CONCEPT address bit A1 is tied to APPLE slot address bit AD0 and
                  ;     further BIT0 of Apple slot is used to turn on or off a bit in 74LS259.
                  ;     Thus, Apple addresses are 4 * CONCEPT addresses.
                  ;

                  ;
                  ; Address bits of Apple floppy controller to the address bits of CONCEPT
                  ;
00000002          ad0on   equ     2
                  ;
                  ; The following are the equates for the bits of the latch
                  ;
00000000          PHASEOFF  equ   $0+0          ;turn phase 0 OFF
00000002          PHASEON   equ   $0+ad0on      ;turn phase 0 ON
                                                ;
00000000          PHASE0OFF equ   $0+0          ;phase 0 off
00000002          PHASE0ON  equ   $0+ad0on      ;phase 0 on
                                                ;
00000004          PHASE1OFF equ   $1*4+0        ;phase 1 off
00000006          PHASE1ON  equ   $1*4+ad0on    ;phase 1 on
                                                ;
00000008          PHASE2OFF equ   $2*4+0        ;phase 2 off
0000000A          PHASE2ON  equ   $2*4+ad0on    ;phase 2 on
                                                ;
0000000C          PHASE3OFF equ   $3*4+0        ;phase 3 off
0000000E          PHASE3ON  equ   $3*4+ad0on    ;phase 3 on
                                                ;
00000010          MOTOROFF  equ   $4*4+0        ;motor off
00000012          MOTORON   equ   $4*4+ad0on    ;motor on
                                                ;
00000014          DRV0EN    equ   $5*4+0        ;drv 0 enable
00000016          DRV1EN    equ   $5*4+ad0on    ;drv 1 enable

```

```
                   ;
                   ; ADsctIO - Read/Write an Apple floppy disk sector
                   ;
                   ;       Enter:  A0.L - Buffer address
                   ;               D1.W - Bytes per sector (256 for Apple)
                   ;               D3.W - Track number     (range. 0-35)
                   ;               D4.W - Sector number    (range. 0-15)
                   ;               D5.W - Read ($32) or Write ($33) command
                   ;
                   ;       Exit.   D7.W - IORESULT
                   ;
                   ;       All other registers are preserved.
                   ;
187E 0C45 0033     ADsecIO CMPI.W  #DskWrit,D5              ;make sure cmd is a read cmd
1882 6604                  BNE.S   ADscio1                  ;it is
1884 6000 FF5E             BRA     FDEopcd                  ;it isn't, return error

1888 48E7 FEFE     ADscio1 MOVEM.L D0-D6/A0-A6,-(sp)        ;save registers
188C 6100 FD52             BSR     FDgetadr                 ;set address registers
                                                            ;A1 = ptr to device description info
                                                            ;A2 = ptr to slot controller registers
                                                            ;A3 = ptr to slot static RAM
1890 6100 0278             BSR     ADmtron                  ;turn on motor
1894 6100 04D6             BSR     ADseek                   ;get to track specified by D3.W
1898 6604                  BNE.S   ADscio9                  ;if error, return
189A 6100 02B4             BSR     ADsecR                   ;read sector specified by D4.W

189E 6100 0264     ADscio9 BSR     ADmtrof                  ;turn off motor
18A2 4CDF 7F7F             MOVEM.L (sp)+,D0-D6/A0-A6        ;restore registers
18A6 4A47                  TST.W   D7                       ;set condition codes *kb 9/3/82*
18A8 4E75                  RTS                              ;return
```

```
                           ;
                           ; Q6 and Q7 define the operation of controller
                           ;
                           ;                               ,Q7   Q6     OPERATION
      00000018             Q6L      equ     $6*4+0         ;L    L   read disk data
      0000001A             Q6H      equ     $6*4+ad0on     ;L    H   sense write protect
      0000001C             Q7L      equ     $7*4+0         ,H    L   write disk data
      0000001E             Q7H      equ     $7*4+ad0on     ;H    H   write store
                           ;
                           ; To write the disk data.
                           ;        set  Q7 high (=1), Q6 low (=0)
                           ,        set  Q6 to high then to low
                           ;
                           ; To read the disk data.
                           ,        set Q7 low and set Q6 low.
                           ;
                           ;
                           ; GENERAL EQUATES
                           ;
      00000056             LNBUF2  equ     $56      ;length of buffer nbuf2

                           ;
                           , ADI5sssd -- Set up constants for Apple 5" single side single density
                           ,
1AAE                       ADI5sssd
1AAE  6100  FB30                   bsr      FDgetadr               ;set address registers
                                                                   ;A1 = ptr to device description info
                                                                   ;A2 = ptr to slot controller registers
                                                                   ;A3 = ptr to slot static RAM
1AB2  337C  0118  0734             move.w   #NBLK5SD,CPfdvsz(A1)   ;set device size in blocks
1AB8  337C  0100  0736             move.w   #BPS5ISD,CPfbps(A1)    ,set bytes per sector
1ABE  137C  0010  0738             move.b   #SCPT5SD,CPfspt(A1)    ,set sectors per track
1AC4  137C  0023  0739             move.b   #TKPS5SD,CPftps(A1)    ,set tracks per side
1ACA  137C  0001  073A             move.b   #1,CPfspd(A1)          ,set sides per disk
1AD0  4229  073B                   clr.b    CPfofst(A1)            ,set first track offset
1AD4  137C  0005  073C             move.b   #DTa5,CPftyp(A1)       ,set floppy type
1ADA  487A  0008+                  pea      ADilvtb                ,set interleave table pointer
1ADE  235F  0730                   move.l   (sp)+,CPfinlv(A1)      ;*
1AE2  4E75                         rts                             ,return
                           ;
                           , Interleave table for Apple floppy disk drives
                           ;
1AE4  00 02 04 06 08 0A    ADilvtb data.b  $0,$2,$4,$6,$8,$A,$C,$E,$1,$3,$5,$7,$9,$B,$D,$F ,Pascal
1AEA  0C 0E 01 03 05 07
1AF0  09 0B 0D 0F
                           ;------ data.b  $0,$3,$6,$9,$C,$F,$2,$5,$8,$B,$E,$1,$4,$7,$A,$D ,CP/M
                           ,------ data.b  $0,$D,$B,$9,$7,$5,$3,$1,$E,$C,$A,$8,$6,$4,$2,$F ,DOS Basic
```

```
                       ;
                       ; ADinit -- Initialise Apple floppy disk drive
                       ;
1AF4  6100  FAEA       ADinit  BSR     FDgetadr        ,set address registers
                                                       ;A1 = pointer to device description info
                                                       ,A2 = pointer to slot controller registers
                                                       ;A3 = pointer to slot static RAM
                       ,*KB 8/23/82* IN ROM DOESN'T NEED ROUTINES IN STATIC RAM
                       ,       , MOVE WRITE CRITICAL CODE INTO THE STATIC RAM
                       ;       ,
                       ,       MOVEM.L A4-A6,-(sp)     ;save registers
                       ,                               ,
                       ,       LEA     ADrd4B,a4       ;BEGINNING OF CRITICAL AREA
                       ,       LEA     SRAMrd4(A3),a5  ;WHERE IT GOES
                       ;       LEA     ADrd4E,a6       ;AFTER CRITICAL CODE AREA
                       ,       BSR.S   ADmov           ,MOVE CODE
                       ;                               ,
                       ,       LEA     ADrd5B,a4       ,BEGINNING OF CRITICAL AREA
                       ,       LEA  '  SRAMrd5(A3),a5  ,WHERE IT GOES
                       ,       LEA     ADrd5E,a6       ,AFTER CRITICAL CODE AREA
                       ,       BSR.S   ADmov           ,MOVE CODE
                       ;                               ;
                       ,       LEA     ADrd6B,a4       ,BEGINNING OF CRITICAL AREA
                       ,       LEA     SRAMrd6(A3),a5  ,WHERE IT GOES
                       ,       LEA     ADrd6E,a6       ;AFTER CRITICAL CODE AREA
                       ;       BSR.S   ADmov           ;MOVE CODE
                       ,                               ;
                       ,       LEA     ADwaitB,a4      ,BEGINNING OF CRITICAL AREA
                       ,       LEA     SRAMwt(A3),a5   ;WHERE IT GOES
                       ,       LEA     ADwaitE,a6      ,AFTER CRITICAL CODE AREA
                       ,       BSR.S   ADmov           ,MOVE CODE
                       ,                               ,
                       ,       MOVEM.L (sp)+,A4-A6     ,restore registers
                               ;
                               , do restore of drive
                               ,
1AF8  6110                     BSR.S   ADmtron         ,TURN ON MOTOR
1AFA  6100  0288               BSR     ADrst           ;restore to track 0
1AFE  6104                     BSR.S   ADmtrof         ,TURN MOTOR OFF
1B00  4A47                     TST.W   D7              ,set condition codes *kb 9/3/82*
1B02  4E75                     RTS                     ,return

                       ;ADmov   MOVE.W  (A4)+,(A5)+     ;move code to static RAM     *KB 8/23/82*
                       ;       CMPA.L  A4,A6           ,finished moving code?
                       ,       BNE.S   ADmov           ,no, move next word
                       ;       rts                     ;return
```

```
                        ;
                        ;  D0.W  -- FREE
                        ;  D1.W  -- FREE
                        ;  D2.W  -- BYTE CNT
                        ;  D3.W  -- BASE BLK  ADRS
                        ;  D4.W  -- FREE
                        ;  D5.W  -- USER CMD
                        ;  D6.W  -- FREE
                        ;
                        ;  A0.L  -- USER BUFFER ADDRESS
                        ;  A1.L  -- DEVICE DESCRIPTION AREA BASE ADDRESS
                        ;  A2.L  -- FLOPPY CONTROLLER BASE ADDRESS
                        ;  A3.L  -- STATIC RAM BASE ADDRESS
                        ;

                        ;
                        ; ADmtrof -- turn motor off
                        ;
1B04  102A  0010        ADmtrof MOVE.B  MOTOROFF(A2),D0 ;TURN OFF FLOPPY DRIVE MOTOR
1B08  4E75                      RTS                     ;return


                        ;
                        ; ADmtron -- turn motor on
                        ;
1B0A  102A  0012        ADmtron MOVE.B  MOTORON(A2),D0  ;TURN ON FLOPPY DRIVE MOTOR
1B0E  4E75                      rts                     ;return

```

```
1B10  6100  FACE       4115* ADrdwr  BSR     FDgetadr        ;set address registers
                       4116*                                 ;A1 = pointer to device description info
                       4117*                                 ;A2 = pointer to slot controller registers
                       4118*                                 ;A3 = pointer to slot static RAM
1B14  4A43             4119*         TST.W   D3              ;test base block
1B16  6B00  FCC0       4120*         BMI     FDEblck         ;jump if first blk rqstd is invalid
1B1A  B669  0734       4121*         cmp.w   CPtdvsx(A1),d3  ;is it in limit
1B1E  6C00  FCB8       4122*         bge     FDEblck         ;jump final block exceeds max
                       4123*                                 ;
1B22  6100  FAE8       4124*         BSR     FDclcTS         ;compute track and sector
1B26  6616             4125*         BNE.S   ADrdwr9         ;if error, return
1B28  61E0             4126*         BSR.S   ADmtron         ;turn motor on
                       4127*
                       4128* ;-------------------------------------------------------------
                       4129* ;  D0.W  -- FREE
                       4130* ;  D1.W  -- BYTES PER SEC
                       4131* ;  D2.W  -- WORD CNT
                       4132* ;  D3.W  -- TRACK ADDRESS
                       4133* ;  D4.W  -- SECTOR ADDRESS
                       4134* ;  D5.W  -- USER COMMAND
                       4135* ;  D6.W  -- FREE
                       4136* ;-------------------------------------------------------------
                       4137* ; READS/WRITES ONLY COMPLETE SECTORS
                       4138* ; For the rest of the code:
                       4139* ;          A0       points to the user buffer address
                       4140* ;          A1       points to the beginning of the device
                       4141* ;                   table entry for this volume in D0 at the entry
                       4142* ;          A2       Contains the NDEVICE address of the slot
                       4143* ;                   specified in the device table for this Volume
                       4144* ;          A3       BASE ADDRESS OF LOCAL STATIC RAM
                       4145* ;-------------------------------------------------------------
                       4146*
1B2A  6100  0240       4147*         bsr     ADseek          ;seek the desired track
1B2E  660E             4148*         bne.s   ADrdwr9         ;if error, return
                       4149*                                 ;
1B30  9441             4150* ADrdwr1 sub.w   d1,d2           ;
1B32  6B0A             4151*         bmi.s   ADrdwr9         ;jump,no more full sectors to rd
1B34  611A             4152*         bsr.s   ADsecR          ;read a full sector
1B36  6606             4153*         bne.s   ADrdwr9         ;error, exit
1B38  6100  FB1E       4154*         bsr     FDincTS         ;get next sector address
1B3C  67F2             4155*         beq.s   ADrdwr1         ;if ok, read the next sector *KB 8/2/82*
                       4156*                                 ;
1B3E  4E75             4157* ADrdwr9 rts                     ;return
                       4158*
```

```
                    ;
                    ; ADinlv -- get the physical sector number for the logical sector number
                    ;           specified in D4.W
                    ;
                    ;       Enter:  D4.W  =  logical sector number
                    ;       Exit:   D4.W  =  physical sector number
                    ;
1B40 2F08           ADinlv  move.l  A0,-(sp)        ;save register
1B42 2078 0730              move.l  CPfinlv.w,A0    ;get interleave table pointer
1B46 6704                   beq.s   ADinlv9         ;just return if no table pointer
1B48 1830 4000              move.b  0(A0,D4.W),D4   ;get physical sector number
1B4C 205F           ADinlv9 move.l  (sp)+,A0        ;restore register
1B4E 4E75                   rts                     ;return

                    ;
                    ; ADsecR -- find the desired sector on the current track by using ADfsec
                    ;           If the sector is found then call ADccRd to read the sector
                    ;           into the buffer.
                    ;
     00007FFF       ADrcSc  equ     $7FFF           ;retry count    *KB 8/23/82*
                                                    ;
1B50 48E7 FA4E      ADsecR  MOVEM.L D0-D4/D6/A1/A4-A6,-(SP)  ;save registers *KB 8/23/82*
1B54 61EA                   bsr.s   ADinlv          ;get physical sector number

1B56 40E7                   MOVE.W  SR, -(SP)       ;save sr               *KB 8/23/82*
1B58 007C 0700              ORI.W   #$0700, SR      ;turn off interrupts   *KB 8/23/82*

1B5C 3C3C 7FFF              MOVE.W  #ADrcSc,D6      ;retry count
1B60 2848                   MOVEA.L A0,A4           ;SAVE A0 INTO A4
                                                    ;
1B62 204C           ADsecR1 MOVEA.L A4,A0           ;
1B64 6100 012A              BSR     ADfsec          ;find sector
1B68 6608                   BNE.S   ADsecR2         ;sector not found - exit
1B6A 6112                   BSR.S   ADccRd          ;read sector
1B6C 57CE FFF4              DBEQ    D6,ADsecR1      ;do until(sector read) or (no more retries)
1B70 6702                   BEQ.S   ADsecR9         ;
                                                    ;
1B72 204C           ADsecR2 movea.l a4,a0           ;error, so recover a0
                                                    ;
1B74                ADsecR9
1B74 46DF                   MOVE.W  (SP)+,SR        ;restore sr            *KB 8/23/82*
1B76 4CDF 725F              MOVEM.L (SP)+,D0-D4/D6/A1/A4-A6 ;save registers *KB 8/23/82*
1B7A 4A47                   TST.W   D7              ;set return condition code
1B7C 4E75                   RTS                     ;return

```

```
                        ;
                        ; ADccRd -- read 8 bit bytes of disk data, retranslates to data,
                        ;           then perform the Exclusive ors to get user data
                        ;
                        ; First 56 bytes of nbuf2 data are read in.
                        ; Then the nubf1 data is read in three groups
                        ; first and second group of 86 (56H) data, the last group
                        ; of 256-86-86 bytes i.e. left over bytes.
                        ;
                        ; Group1 is de-niblized using deniblizing table DNIBL2
                        ; Group2 is de-niblized using deniblizing table DNIBL3
                        ; Group3 is de-niblized using deniblizing table DNIBL4
                        ;
                        ; disk data format
                        ; +-- prologue --+-- data field ----------+-- cksum --+-- epilogue --+
                        ; :              :                        :           :              :
                        ; :  D5 AA AD    : 342 bytes of disk data : one byte  :   DE AA EB   :
                        ; :              :                        :           :              :
                        ; +--------------+------------------------+-----------+--------------+
                        ;
     00000400           ADrcRd  equ     $400            ;read sector retry count

1B7E  48E7  0208        ADccRd  movem.l d6/a4,-(sp)     ;*KB 8/23/82*
                                ;
                                ; 1) read prologue of disk data field. Try 1024 times.
                                ;    If prologue not found, indicate error
                                ;
1B82  3E3C  0400                move.w  #ADrcRd,d7      ;RETRY COUNT *KB 8/24/82*
                                ;
                                ; SEARCH FOR PROLOGUE
                                ;
1B86  51CF  0006        ADccRd1 dbf     d7,ADccRd2      ;
1B8A  6000  0086                bra     ADccRd8         ;
                                                        ;
1B8E  102A  0018        ADccRd2 move.b  Q6L(a2),d0      ; check for D5 HEX
1B92  6AFA                      bpl.s   ADccRd2         ;jmp if byte has not been asmbld
                                                        ;
1B94  B03C  00D5        ADccRd3 cmp.b   #$D5,d0         ;is a part of prologue
1B98  66EC                      bne.s   ADccRd1         ;no
                                                        ;*KB 8/23/82*  DELAY A LITTLE
1B9A  4240                      clr.w   d0              ;clear bit 8 to 15 of D reg
1B9C  4241                      clr.w   d1              ;clear bit 8 to 15 of D reg
1B9E  4243                      clr.w   d3              ;clear bit 8 to 15 of D reg
```

```
1BA0  102A  0018   ADccRd4 move.b  Q6L(a2),d0        ;check for AA hex
1BA4  6AFA                 bpl.s   ADccRd4           ;jmp if byte has not been asmbld
1BA6  B03C  00AA           cmp.b   #$AA,d0           ;is a part of prologue
1BAA  66E8                 bne.s   ADccRd3           ;no, see if start of prologue
                                                     ;*KB 8/23/82*  DELAY A LITTLE
1BAC  4242                 clr.w   d2                ;clear bit 8 to 15 of D reg
1BAE  49FA  FDFE+          LEA     DNIBL, A4         ;translate table

1BB2  102A  0018   ADccRd5 move.b  Q6L(a2),d0        ;check for AD hex
1BB6  6AFA                 bpl.s   ADccRd5           ;jmp if byte has not been asmbld
1BB8  B03C  00AD           cmp.b   #$AD,d0           ;is a part of prologue
1BBC  66D6                 bne.s   ADccRd3           ;no, see if start of prologue
                           ;
                           ; 2) prologue has been found, read 56H bytes of disk data into NBUF2
                           ;    This DATA is then used to get the LEAST SIGNIFICANT 2 BITS OF A BYTE
                           ;        NOTE THIS DATA IS OF THE FORM.
                           ;        N55   .eor.  0
                           ;        N54   .eor.  N55
                           ;        N53   .eor.  N54
                           ;        N52   .eor.  N53
                           ;        ...............
                           ;        N01   .eor.  N02
                           ;        N00   .eor.  N01
                           ;
1BBE  4BF8  0E02           LEA     NBUF2.W, A5                ;*KB 8/23/82*
1BC2  3C3C  0055           MOVE.W  #LNBUF2-1, D6              ;*KB 8/23/82*
1BC6  6156                 BSR.S   ADrd4                      ;Read into NBUF2,*KB 8/23/82*
                   ;;;     jsr     SRAMrd4(A3)                ;Read NBUF2 ,NOT MOVING TO STATIC RAM
                           ;
                           ; D2 is =  N00
                           ;
1BC8  43F8  0D00           LEA     NBUF1.W, A1                ;*KB 8/23/82*
1BCC  3C38  0736           MOVE.w  CPfbps.w,D6                ;read 256 bytes of data one byte of chk sum
                                                              ;and verify epilogue
1BD0  617C                 BSR.S   ADrd6                      ;*KB 8/23/82*
                   ;;;     jsr     SRAMrd6(A3)                ;DO ADrd6 CODE <===== LOOK !!!!!!
1BD2  6642                 bne.s   ADccRd9                    ;jmp if error

                           ; TRANSLATE the data read from the disk.
                           ;
1BD4  43F8  0D00           LEA.L   NBUF1.W,A1                 ;REPOSITION PTR TO BEGINNING
                           ;
                           ; 3) translate first group of 56H bytes i.e. 86 bytes
                           ;
1BD8  4BF8  0E02           lea.l   NBUF2.W,A5                 ;RD5  GROUP
1BDC  4DFA  FCCD+          lea.l   DNIBL2,a6
1BE0  3C3C  0055           move.w  #LNBUF2-1,d6
1BE4  614C                 BSR.S   ADrd5                      ;*KB 8/23/82*
```

```
                          4299*            ;
                          4300*            ; 4) translate second group of 56H bytes      RD6 GROUP
                          4301*            ;    D2 contains   A55
                          4302*            ;
1BE6  4BF8  0E02          4303*            lea l   NBUF2.W,a5
1BEA  4DFA  FCC0+         4304*            lea.l   DNIBL3,a6
1BEE  3C3C  0055          4305*            move.w  #LNBUF2-1,d6
1BF2  613E                4306*            BSR.S   ADrd5                    ;*KB 8/23/82*
                          4307*            ;
                          4308*            ; 5) translate the third, last group.         RD7 GROUP
                          4309*            ;    D2 contains Aab
                          4310*            ;
1BF4  4BF8  0E02          4311*            lea.l   NBUF2.W,a5
1BF8  4DFA  FCB3+         4312*            lea.l   DNIBL4,a6
1BFC  3C38  0736          4313*            MOVE w  CPfbps.w,D6              ,REMAINING OF 256 BYTES
1C00  DC7C  FF53          4314*            add w   #(-LNBUF2-LNBUF2)-1,d6   ;*
1C04  612C                4315*            BSR S   ADrd5                    ,*KB 8/23/82*
                          4316*            ;
                          4317*            ; NOTE that the last byte of user data was written as CHK SUM
                          4318*            ; so read the last byte and compare with chk sum. EOR should be ZERO
                          4319*            ; D2 contains  Aff
                          4320*            ;
1C06  4247                4321*            clr.w   d7                       ,d7 =0 indicates no error
1C08  1019                4322*            move.b  (a1)+,d0                 ;
1C0A  1234  0000          4323*            move.b  0(a4,d0),d1              ,     dniblize disk data byte
1C0E  B302                4324*            eor.b   d1,d2                    ,
1C10  6704                4325*            beq.s   ADccRd9                  ;no  chk sum error if  zero rslt
                          4326*                                             ,
1C12  3E3C  FFED          4327* ADccRd8 move.w  #RRNF,d7                 ,error code to d7
                          4328*
1C16  4CDF  1040          4329* ADccRd9 movem.l (sp)+,d6/a4              ,restore registers      ,*KB 8/23/82*
1C1A  4A47                4330*            tst w   d7                       ,set return condition code
1C1C  4E75                4331*            rts                              ,return
```

```
                   ;
                   ; ADrd4 -- moved to and executed at SRAMrd4(slot static RAM)
                   ;
1C1E               ADrd4B                              ;start of SRAMrd4 code
1C1E  102A  0018   ADrd4   move.b  Q6L(a2),d0          ;
1C22  6AFA                 bpl.s   ADrd4               ;jmp if byte has not been asmbld
1C24  1234  0000           move.b  0(a4,d0),d1         ;dniblise disk data byte
1C28  1AC1                 move.b  d1,(a5)+            ;store it. it is N(I) EOR N(I-1)
1C2A  B302                 eor.b   d1,d2               ;ULTIMATELY d2 = N00
1C2C  51CE  FFF0           dbf     d6,ADrd4            ;
1C30  4E75                 rts                         ;return
1C32               ADrd4E                              ;end of SRAMrd4 code

                   ;
                   ; ADrd5 -- Translate bytes **not done in time critical section**
                   ;
1C32               ADrd5B                              ;start of SRAMrd5 code
1C32  1019         ADrd5   move.b  (a1)+,d0            ;16
1C34  1234  0000           move.b  0(a4,d0),d1         ;18     dniblise disk data byte
1C38  B302                 eor.b   d1,d2               ;4
1C3A  161D                 move.b  (a5)+,d3            ;12     get an entry of nbuf2
1C3C  1E36  3000           move.b  0(a6,d3),d7         ;18     get low order bits b1,b0 of
1C40  BF02                 eor.b   d7,d2               ;4      a byte and mask them into byte
1C42  10C2                 move.b  d2,(a0)+            ;14     store the byte into user area
1C44  51CE  FFEC           dbf     d6,ADrd5            ;10
                                                       ;TOTAL = 106 CYCLES
1C48  C43C  00FC           and.b   #$FC,d2             ;mask of 2 LSbits. Now d2 contains A55
1C4C  4E75                 rts                         ;return
1C4E               ADrd5E                              ;end of SRAMrd5 code

                   ;
                   ; ADrd6 -- moved to and executed at SRAMrd6(slot static RAM)
                   ;
1C4E               ADrd6B                              ;start of SRAMrd6 code
1C4E               ADrd6
1C4E  102A  0018   ADrd61  move.b  Q6L(a2),d0          ;read 256 bytes of data and one byte of CHK SUM
1C52  6AFA                 bpl.s   ADrd61              ;
1C54  12C0                 move.b  d0,(a1)+            ;move bytes into BUFADRS
1C56  51CE  FFF6           dbf     d6,ADrd61           ;
                           ;
                           ; READ epilogue
                           ;
1C5A  102A  0018   ADrd62  move.b  Q6L(a2),d0          ;
1C5E  6AFA                 bpl.s   ADrd62              ;jmp if byte has not been asmbld
1C60  4247                 clr.w   d7                  ;pre-set d7 =0 to indicate no error
1C62  B03C  00DE           cmp.b   #$DE,d0             ;
1C66  6702                 beq.s   ADrd69              ;jmp if ok
1C68  7EED                 moveq   #RRNF,d7            ;else move error code to d7
                                                       ;
1C6A  4A47         ADrd69  tst.w   d7                  ;set return condition code
1C6C  4E75                 rts                         ;return
1C6E               ADrd6E                              ;end of SRAMrd6 code

```

```
                     ;
                     ; ADwaitB -- moved to and executed at SRAMwt(slot static RAM)
                     ;            delay of units specified in D7
                     ;            Each unit is 100 micro-seconds
                     ;
      0000000A       dly100m equ     010              ;
                                                      ;
1C6E                 ADwaitB                          ;start of SRAMwt code
1C6E  48A7  0600             movem.w d5/d6,-(sp)      ;save registers
1C72  3C3C  000A     ADwait1 move.w  #dly100m,d6      ;each count of d6 = 1100/112 = 10 micro second
1C76  4E71           ADwait2 nop                      ;delay 100 micro-seconds
1C78  4E71                   nop                      ;
1C7A  4E71                   nop                      ;
1C7C  4E71                   nop                      ;
1C7E  4E71                   nop                      ;
1C80  4E71                   nop                      ;
1C82  51CE  FFF2             dbf     d6,ADwait2       ;
1C86  51CF  FFEA             dbf     d7,ADwait1       ;
1C8A  4C9F  0060             movem.w (sp)+,d5/d6      ;restore registers
1C8E  4E75                   rts                      ;return
1C90                 ADwaitE                          ;end of SRAMwt code
```

```
                  ;
                  ; ADfsec -- call ADrdad until it finds the sector specified in D4.B or
                  ;               it has no more retries ieft.
                  ;          Calling routine must disable all the interrupts before making call
                  ;
      00000064    ADfsnrv equ     100    ;number of revolutions until record not found   *KB 8/23/82*

1C90  48E7  0208  ADfsec  MOVEM.L  D6/A4,-(sp)     ,save register  *KB 8/23/82*
1C94  1C38  0738          move.b  CPfspt.w,d6      ,compute retry count
1C98  4886                ext.w   d6               ,*
1C9A  CCFC  0064          mulu    #ADfsnrv,D6      ;*

1C9E  611C        ADfsec1 BSR.S   ADrdad           ,get this sectors adr mark
1CA0  4A47                tst.w   d7               ,ERROR?
1CA2  6608                bne.s   ADfsec2          ,YES, RETRY
                          ;
                          , see if this is the correct track and sector number
                          ;                                     *KB 8/23/82* removed track check
1CA4  4247                clr.w   d7               ,CLR error code to indicate no error
1CA6  B838  0E84          CMP.B   AMBUF+AMsec.W,D4;found sector?
1CAA  6708                BEQ.S   ADfsec9          ,successful ,so exit
1CAC  51CE  FFF0  ADfsec2 DBF     D6,ADfsec1       ,do until(found sector) or (no more retries)
                          ;
                          ; RETRIES FAILED
                          ;
1CB0  3E3C  FFED  ADfsec8 MOVE.W  #RRNF,D7

1CB4  4CDF  1040  ADfsec9 MOVEM.L  (sp)+,D6/A4     ;save register *KB 8/23/82*
1CB8  4A47                tst.w   d7               ;set return condition code
1CBA  4E75                RTS                      ,return
```

```
                          ;
                          ; ADrdad -- read the address field of the Apple floppy
                          ;
                          ; Format of the address field.
                          ;
                          ; +- prologue -+- vol name -+- track -+- sector -+- cksum -+- epilogue -+
                          ; :            :            :         :          :         :            :
                          ; :  D5 AA 96  :   XX  YY   :  XX YY  :  XX YY   :  XX YY  :  DE AA EB  :
                          ; :            :            :         :          :         :            :
                          ; +------------+------------+---------+----------+---------+------------+
                          ;
                          ; The data byte is split into 4 by 4 format, and then is written to disk.
                          ; A data byte d7 d6 d5 d4 d3 d2 d1 d0 is split as follows:
                          ;
                          ; dsk byte1:   1 d7  1 d5 1  d3  1 d1
                          ; dsk byte2:   1 d6  1 d4 1  d2  1 d0
                          ;
                          ; Shifting the first byte left by 1 and .AND. with next byte,
                          ; the original data byte can be reconstructed.
                          ;
                          ;       Calling routine must disable interrupts and save the registers D0-D2/A4.
                          ;       *KB 8/23/82*
                          ;
    00000400              ADrcAd  equ     $400            ;read address retry count

1CBC  3E3C  0400          ADrdad  move.w  #ADrcAd,d7      ;set retry count

1CC0  51CF  0006          ADrdad1 dbf     d7,ADrdad2      ;if no more retries
1CC4  6000  00A0                  bra     ADrdad8         ;then report error

                          ; FIND PROLOGUE
                          ;
1CC8  102A  0018          ADrdad2 move.b  Q6L(a2),d0      ;check for D5 HEX
1CCC  6AFA                        bpl.s   ADrdad2         ;wait until data ready
1CCE  B03C  00D5          ADrdad3 cmp.b   #$D5,d0         ;is a part of prologue
1CD2  66EC                        bne.s   ADrdad1         ;no, try again
1CD4  4242                        clr.w   d2              ;clear chksum

1CD6                      ADrdad3a
1CD6  102A  0018                  move.b  Q6L(a2),d0      ;check for AA hex
1CDA  6AFA                        bpl.s   ADrdad3a        ;wait until data ready
1CDC  B03C  00AA                  cmp.b   #$AA,d0         ;is a part of prologue
1CE0  66EC                        bne.s   ADrdad3         ;no
1CE2  49F8  0E82                  lea     AMBUF.W,a4      ;delay and get adr mark buf addr

1CE6                      ADrdad3b
1CE6  102A  0018                  move.b  Q6L(a2),d0      ;check for 96 hex
1CEA  6AFA                        bpl.s   ADrdad3b        ;wait until data ready
1CEC  B03C  0096                  cmp.b   #$96,d0         ;is a part of prologue
1CF0  66DC                        bne.s   ADrdad3         ;no
                                  ;
                                  ; prologue has been found.  Now read the vol header
                                  ; read 4 bytes: : vol name : track : sector : chksum :
                                  ;
```

```
1CF2  3E3C  0003        4496*         move.w  #4-1,d7          ,number of bytes is 4   I DON'T THINK NEED
                        4497*
                        4498*         , REPEAT CODE 4 TIMES INSTEAD OF LOOP
                        4499*         ;
1CF6                    4500* ADrdad4A                         ,VOL NAME
1CF6  102A  0018        4501*         move.b  Q6L(a2),d0       ,read first nibble
1CFA  6AFA              4502*         bpl.s   ADrdad4A         ;wait until data ready
1CFC  E318              4503*         rol.b   #1,d0            ,rotate left by 1
1CFE                    4504* ADrdad5A
1CFE  122A  0018        4505*         move.b  Q6L(a2),d1       .read second nibble
1D02  6AFA              4506*         bpl.s   ADrdad5A         ,jump if byte has not been asmbld
1D04  C200              4507*         and.b   d0,d1            ;and the two to get actual byte
1D06  18C1              4508*         move.b  d1,(a4)+         ,store it in AMBUF
1D08  B302              4509*         eor.b   d1,d2            ,create chksum
                        4510*
1D0A                    4511* ADrdad4B                         ,TRACK NUMBER
1D0A  102A  0018        4512*         move.b  Q6L(a2),d0       ,read first nibble
1D0E  6AFA              4513*         bpl.s   ADrdad4B         ;wait until data ready
1D10  E318              4514*         rol.b   #1,d0            ,rotate left by 1
1D12                    4515* ADrdad5B
1D12  122A  0018        4516*         move.b  Q6L(a2),d1       ;read second nibble
1D16  6AFA              4517*         bpl.s   ADrdad5B         ,jump if byte has not been asmbld
1D18  C200              4518*         and.b   d0,d1            ,and the two to get actual byte
1D1A  18C1              4519*         move.b  d1,(a4)+         ;store it in AMBUF
1D1C  B302              4520*         eor.b   d1,d2            ;create chksum
                        4521*
1D1E                    4522* ADrdad4C                         ;SECTOR NUMBER
1D1E  102A  0018        4523*         move.b  Q6L(a2),d0       ,read first nibble
1D22  6AFA              4524*         bpl.s   ADrdad4C         ,wait until data ready
1D24  E318              4525*         rol.b   #1,d0            ;rotate left by 1
1D26                    4526* ADrdad5C
1D26  122A  0018        4527*         move.b  Q6L(a2),d1       ,read second nibble
1D2A  6AFA              4528*         bpl.s   ADrdad5C         ,jump if byte has not been asmbld
1D2C  C200              4529*         and.b   d0,d1            ;and the two to get actual byte
1D2E  18C1              4530*         move.b  d1,(a4)+         ;store it in AMBUF
1D30  B302              4531*         eor.b   d1,d2            ,create chksum
                        4532*
1D32                    4533* ADrdad4D                         ,CHECKSUM
1D32  102A  0018        4534*         move.b  Q6L(a2),d0       ;read first nibble
1D36  6AFA              4535*         bpl.s   ADrdad4D         ,wait until data ready
1D38  E318              4536*         rol.b   #1,d0            ;rotate left by 1
1D3A                    4537* ADrdad5D
1D3A  122A  0018        4538*         move.b  Q6L(a2),d1       ;read second nibble
1D3E  6AFA              4539*         bpl.s   ADrdad5D         ,jump if byte has not been asmbld
1D40  C200              4540*         and.b   d0,d1            ;and the two to get actual byte
1D42  18C1              4541*         move.b  d1,(a4)+         ;store it in AMBUF
1D44  B302              4542*         eor.b   d1,d2            ;create chksum
                        4543*
1D46  4A02              4544*         tst.b   d2               ;compare chksum to 0
1D48  661C              4545*         bne.s   ADrdad8          ;return if chksum not = 0
                        4546*         ;
                        4547*         ; VERIFY EPILOGUE  (DE AA EB)
                        4548*         ;
1D4A  102A  0018        4549* ADrdad6 move.b  Q6L(a2),d0       ;get data byte
```

```
1D4E  6AFA            4550*          bpl.s    ADrdad6        ;jump if byte has not been asmbld
1D50  B03C  00DE      4551*          cmp.b    #$DE,d0        ;
1D54  6610            4552*          bne.s    ADrdad8        ;return if error
                      4553*
1D56  4247            4554*          clr.w    d7             ;set d7 =0 to indicate no error
                      4555*
1D58  102A  0018      4556* ADrdad7 move.b   Q6L(a2),d0     ;get data byte
1D5C  6AFA            4557*          bpl.s    ADrdad7        ;jump if byte has not been asmbld
1D5E  B03C  00AA      4558*          cmp.b    #$AA,d0        ;
1D62  6602            4559*          bne.s    ADrdad8        ;return error code if error
1D64  4E75            4560*          rts
                      4561*                                  ;
1D66  3E3C  FFED      4562* ADrdad8 move.w   #RRNF,d7       ;error code to d7
1D6A  4E75            4563*          rts                     ;return
```

```
                         ,
                         ; ADseek -- seek a given track
                         ;
                         ;       Enter:  D3.W - desired track number
                         ;

      00000100           ADrcSk  equ     $100            ,retry count

1D6C  48E7 F208          ADseek  movem.l d0-d3/d6/a4,-(sp)

1D70  3C3C 0100          ADsk1   MOVE.W  #ADrcSk,D6      ,retry count
1D74  4247                       clr.w   d7

1D76  40E7               ADsk2   move.w  sr,-(sp)
1D78  007C 0700                  ori.w   #$0700,sr       ,disable interrupts
1D7C  6100 FF3E                  BSR     ADrdad          ,get track number where head is
1D80  46DF                       move.w  (sp)+,sr        ;restore the interrupt

1D82  4A47                       tst.w   d7              ;ERROR?
1D84  6706                       beq.s   ADsk3           ,If no error then jump

                         ;;;,    bsr.s   ADrst           ,go to track 0
1D86  51CE FFEE                  DBF     D6,ADsk2        ;do until(found sector
1D8A  6020                       bra.s   ADsk9           ,tried enough
                                 ;
                                 ;  See if head is positioned on desired track
                                 ;  issue STEPIN or STEPOUT to get to the correct track
                                 ,
1D8C  49F8 0E62          ADsk3   lea.l   APLSVAR.W,a4

1D90  4240                       clr.w   d0              ;form the current half
1D92  1038 0E83                  move.b  AMBUF+AMtrk.W,d0 ,track and save
1D96  E348                       lsl.w   #1,d0           ,half track into d0
1D98  1940 0003                  move.b  d0,curtrk(a4)   ,half track number to

1D9C  B638 0E83                  CMP.b   AMBUF+AMtrk.W,d3
1DA0  670A                       beq.s   ADsk9           ,exit if equal,found track
1DA2  6D04                       blt.s   ADsk4           ,jump if track is < current track

1DA4  6132                       bsr s   ADccSin         ,step in
1DA6  60C8                       bra.s   ADsk1           ,see if another track

1DA8  612A               ADsk4   bsr.s   ADccSout        ;step out
1DAA  60C4                       bra.s   ADsk1           ,see if another track

1DAC  4CDF 104F          ADsk9   movem.l (sp)+,d0-d3/d6/a4
1DB0  4A47                       tst.w   d7              ;set condition codes
1DB2  4E75                       rts
```

```
                          4614* ;
                          4615* ; ADrst -- restore floppy to track 0
                          4616* ;
1DB4  48E7  9008          4617* ADrst   movem.l d0/d3/a4,-(sp)
1DB8  49F8  0E62          4618*         lea.l   APLSVAR.W,a4
1DBC  197C  0046  0003    4619*         move.B  #2*35,curtrk(a4)
1DC2  4243                4620*         clr.w   d3
1DC4  4240                4621*         clr.w   d0
1DC6  6134                4622*         bsr.s   ADseek1
1DC8  422C  0003          4623*         clr.b   curtrk(a4)      ;*KB 8/23/82*
1DCC  4CDF  1009          4624*         movem.l (sp)+,d0/d3/a4
1DD0  4247                4625*         clr.w   D7              ;force no error return
1DD2  4E75                4626*         rts                     ;return
                          4627* ;
                          4628* ; ADccSout - step out
                          4629* ; ADccSin -- step in
                          4630* ;
1DD4                      4631* ADccSout
1DD4  4287                4632*         CLR.L   D7              ;DO STEP OUT
1DD6  6002                4633*         BRA.S   ADSio1
                          4634*
1DD8  7E01                4635* ADccSin MOVEQ   #1,D7           ;DO STEP IN
                          4636*
1DDA  48E7  1008          4637* ADSio1  movem.l d3/a4,-(sp)
1DDE  4243                4638*         clr.w   d3
1DE0  49F8  0E62          4639*         lea.l   APLSVAR.W,a4
1DE4  162C  0003          4640*         move.b  curtrk(a4),d3
                          4641*
1DE8  4A87                4642*         TST.L   D7              ;should step in
1DEA  6704                4643*         BEQ.S   ADSio2          ;no, step out
1DEC  5403                4644*         addq.b  #1*2,d3         ;point to next track IN
1DEE  6002                4645*         BRA.S   ADSio3
1DF0  5503                4646* ADSio2  SUBQ.B  #1*2,d3         ;point to next track OUT
                          4647*
1DF2  6108                4648* ADSio3  bsr.s   ADseek1
1DF4  4CDF  1008          4649*         movem.l (sp)+,d3/a4
1DF8  4247                4650*         clr.w   D7              ;force no error return
1DFA  4E75                4651*         rts                     ;return
                          4652* ;
                          4653* ; ADseek1 -- seek the track desired by the caller
                          4654* ;
                          4655* ;       Enter:  D3.W - desired track number
                          4656* ;
1DFC  1E2A  0000          4657* ADseek1 move.b  PHASE0OFF(a2),d7        ;Turn all 4 phases off
1E00  1E2A  0004          4658*         move.b  PHASE1OFF(a2),d7
1E04  1E2A  0008          4659*         move.b  PHASE2OFF(a2),d7
1E08  1E2A  000C          4660*         move.b  PHASE3OFF(a2),d7
1E0C  6102                4661*         bsr.s   ADccSk
1E0E  4E75                4662*         rts                             ;return
```

```
                         ;
                         ; ADccSk -- seek the track desired by the caller
                         ;
                         ;        Enter:  D3.W - desired track number
                         ;                CURTRK location holds the current track number.
                         ;
                         ;        Exit.   The final track  -----) curtrk
                         ;                The curtrk-1     -----) PRIOR
                         ;
                         ;        APPLE DOS NAME CONVENTION
                         ;
                         ;        (TRKN)   = Desired track number
                         ;        (CURTRK) = The current track number where floppy heads are positioned
                         ;        (PRIOR)  =  The (CURTRK)-1
                         ;        (TRKCNT) = The number of track the floppy has been moved so far.
                         ;
1E10  48E7  8608         ADccSk  movem.l d0/d5-d6/a4,-(sp)      ;save registers
1E14  4245                       clr.w   d5                     ;clear bit 8 to 15
1E16  4246                       clr.w   d6                     ;clear bit 8 to 15
1E18  4247                       clr.w   d7                     ;clear bit 8 to 15
1E1A  49F8  0E62                 lea     APLSVAR.W,a4           ;
1E1E  B62C  0003                 cmp.b   curtrk(a4),d3          ;is desired trk = current track
1E22  6764                       beq.s   ADccSk9                ;exit if equal

1E24  422C  0000                 clr.b   trkcnt(a4)             ;init track count
1E28  40E7                       move.w  sr,-(sp)               ;save interrupt level
1E2A  007C  0700                 ori.w   #$0700,sr              ;disable interrupts

1E2E  196C  0003  0001   ADccSk1 move.b  curtrk(a4),prior(a4)   ;curtrk to prior
1E34  B62C  0003                 cmp.b   curtrk(a4),d3          ;
1E38  673E                       beq.s   ADccSk8                ;jump if current trk = desired trk
1E3A  6D0E                       blt.s   ADccSk2                ;jump if  desired trk( CURTRK

1E3C  1E03                       move.b  d3,d7                  ; Seek IN
1E3E  9E2C  0003                 sub.b   curtrk(a4),d7          ;trkn - curtrk --) d7
1E42  5307                       subq.b  #1,d7                  ;trkn - curtrk -1 to d7
1E44  522C  0003                 addq.b  #1,curtrk(a4)          ;
1E48  600C                       bra.s   ADccSk3                ;

1E4A  1E2C  0003         ADccSk2 move.b  curtrk(a4),d7          ;curtrk is ) desired track
1E4E  9E03                       sub.b   d3,d7                  ; Seek OUT
1E50  5307                       subq.b  #1,d7                  ;trkn - curtrk -1 to d7
1E52  532C  0003                 subq.b  #1,curtrk(a4)          ;

1E56  BE2C  0000         ADccSk3 cmp.b   trkcnt(a4),d7          ;calculate index to the delay table
1E5A  6D04                       blt.s   ADccSk4                ;jump if d7 is less than trkcnt
1E5C  1E2C  0000                 move.b  trkcnt(a4),d7          ;

1E60  BE3C  0008         ADccSk4 cmp.b   #08,d7                 ; destination .cmp  source
1E64  6C02                       bge.s   ADccSk5                ; leave d0 alone if d7 )= 08
                                                                ;to use as index into turn ON delay OFFdelay
1E66  1007                       move.b  d7,d0                  ; do STEP
1E68  1A2C  0003         ADccSk5 move.b  curtrk(a4),d5          ;
1E6C  1C2C  0001                 move.b  prior(a4),d6           ;
```

```
1E70  611C                 4718*          BSR.S   ADccStp                  ;
1E72  522C  0000           4719*          addq.b  #1,trkcnt(a4)            ,
1E74  60B6                 4720*          bra.s   ADccSk1                  ;seek next track
                           4721*
1E78  1C2C  0003           4722* ADccSk8 move.b  curtrk(a4),d6            ;
1E7C  6146                 4723*          bsr.s   ADclrPh                  ;clear the phase turned on the last time
1E7E  46DF                 4724*          move.w  (sp)+,sr                 ;restore interrupt level
1E80  3E3C  0100           4725*          move.w  #$100,d7                 ;
1E84  6100  FDE8           4726*          BSR     ADwaitB                  ;wait
                           4727* ,,       jsr     SRAMwt(A3)               ,ADwaitB
                           4728*
1E88  4CDF  1061           4729* ADccSk9 movem.l (sp)+,d0/d5-d6/a4        ,restore registers
1E8C  4E75                 4730*          rts                              ,return
```

```
                        ;
                        ; ADccStp -- move the floppy in or out by one phase i.e. Half track
                        ;
                        ;       Enter:  the track num to use for setting a phase   -----) d5
                        ;               the track num to use for clearing a phase  -----) d6
                        ;               the index for ADtblON and ADtblOF to fetch a dealy count --) d0
                        ;
                        ;       Note:   THE BIT 8 TO 15 OF D0,D5,D6 MUST BE ZERO.
                        ;
                        ;       Exit:   SETS one phase and clears a phase
                        ;
                        ; TO MOVE FLOPPY HALF TRACK INWARD.
                        ;       set phase    i
                        ;       clear phase  i-1
                        ;
                        ; TO MOVE FLOPPY HALF TRACK OUTWARD.
                        ;       set phase    i-1
                        ;       clear phase  i
                        ;
1E8E  48E7  0802        ADccStp movem.l d4/a6,-(sp)              ,
1E92  6124                      bsr.s   ADsetPh                  ,turn on phase
1E94  4247                      clr.w   d7                       ;clear bit 8 to 15  of  D7
1E96  4DFA  0038+               lea     ADtblON,a6               ,
1E9A  1E36  0000                move.b  0(a6,d0),d7              ;
1E9E  6100  FDCE                BSR     ADwaitB
                        ;;      jsr     SRAMwt(A3)               ,ADwaitB
1EA2  6120                      bsr.s   ADclrPh                  ;turn off phase
1EA4  4247                      clr.w   d7                       ;clear bit 8 to 15  of  D7
1EA6  4DFA  0030+               lea     ADtblOF,a6               ;
1EAA  1E36  0000                move.b  0(a6,d0),d7              ;
1EAE  6100  FDBE                BSR     ADwaitB
                        ;;      jsr     SRAMwt(A3)               ;ADwaitB
1EB2  4CDF  4010                movem.l (sp)+,d4/a6              ,
1EB6  4E75                      rts                              ;
                                                                 ;
1EB8  0205  0003        ADsetPh andi.b  #03,d5                   ,
1EBC  E50D                      lsl.b   #2,d5                    ;GENERATE APPLE ADRS
1EBE  1E32  5002                move.b  phaseon(a2,d5),d7        ;
1EC2  4E75                      rts                              ;
                                                                 ;
1EC4  0206  0003        ADclrPh andi.b  #03,d6                   ;
1EC8  E50E                      lsl.b   #2,d6                    ;GENERATE APPLE ADRS
1ECA  1E32  600G                move.b  phaseoff(a2,d6),d7       ;
1ECE  4E75                      rts                              ;
                                                                 ;
1ED0  00 2F 27 23       ADtblON DATA.B  1-1,$30-1,$28-1,$24-1    ;
1ED4  1F 1D 1C 1B               DATA.B  $20-1,$1E-1,$1D-1,$1C-1  ,even number of bytes
                                                                 ;
1ED8  6F 2B 25 21       ADtblOF data.b  $70-1,$2C-1,$26-1,$22-1  ;
1EDC  1E 1D 1C 1B               DATA.B  $1F-1,$1E-1,$1D-1,$1C-1  ;even number of bytes
```

```
     000100B4+        4784*         end      setup

ABOOT      011814+  ADRD62     011C5A+  AMTRK     00000001  CPIVEC6  00010084  CSTEP     00000020
ABOOT1     011836+  ADRD69     011C6A+  AMVOL     00000000  CPIVEC7  00010088  CSTEPIN   00000040
ABOOT90    011854+  ADRD6B     011C4E+  APL5VAR   00000E62  CPKBGETC 00010054  CSTEPOUT  00000060
ADOON     00000002  ADRD6E     011C6E+  BADDEST   00000086  CPKBINIT 00010050  CSUMRD    00000004
ADBIO1     01186C+  ADRDAD     011CBC+  BADSOCK   00000084  CPLBLKIO 00010026  CURSON    00000002
ADBIO9     011874+  ADRDAD1    011CC0+  BASERAM   00000900  CPLBOOT  00010022  CURTRK    00000003
ADBLKIO    011856+  ADRDAD2    011CC8+  BLKSZ     00000200  CPLBOOTJ 00010020  CWRSEC    000000A0
ADCCRD     011B7E+  ADRDAD3    011CCE+  BPS5ISD   00000100  CPLDSKIO 0001002A  CWRTRK    000000F0
ADCCRD1    011B86+  ADRDAD3A   011CD6+  BPS8IDD   00000100  CPOBLKIO 00010016  DCBLKHI   00000003
ADCCRD2    011B8E+  ADRDAD3B   011CE6+  BPS8ISD   00000080  CPOBOOT  00010012  DCBLKLO   00000002
ADCCRD3    011B94+  ADRDAD4A   011CF6+  CFRCINT   000000D0  CPOBOOTJ 00010010  DCDRV     00000001
ADCCRD4    011BA0+  ADRDAD4B   011D0A+  CHEND       010188+ CPODSKIO 0001001A  DCLEN     00000004
ADCCRD5    011BB2+  ADRDAD4C   011D1E+  CHEND1      01018C+ CPOMNIBF 0008DFD0  DCMD      00000000
ADCCRD8    011C12+  ADRDAD4D   011D32+  CHERR       010176+ CPOMNIRC 0000070F  DEBOP     00000008
ADCCRD9    011C16+  ADRDAD5A   011CFE+  CHERR1      01017A+ CPOMNRAM 00000880  DEVADOFS  00000020
ADCCSIN    011DD8+  ADRDAD5B   011D12+  CMDACPT   000000FE  CPOSBLK  00000709  DLY100M   0000000A
ADCCSK     011E10+  ADRDAD5C   011D26+  CNSTSKT   000000B0  CPOSDRV  00000708  DNIBL       0119AE+
ADCCSK1    011E2E+  ADRDAD5D   011D3A+  CPABLKIO  00010042  CPOSSLOT 00000706  DNIBL2      0118AB+
ADCCSK2    011E4A+  ADRDAD6    011D4A+  CPAINIT   0001004A  CPOSSRVR 00000707  DNIBL3      0118AC+
ADCCSK3    011E56+  ADRDAD7    011D58+  CPASCTIO  00010046  CPROMLVL 0001000D  DNIBL4      0118AD+
ADCCSK4    011E60+  ADRDAD8    011D66+  CPBLKIO   00000714  CPROMVRS 0001000C  DRV0EN    00000014
ADCCSK5    011E68+  ADRDWR     011B10+  CPBTSLOT  00000700  CPSCNOFS 00000764  DRV1EN    00000016
ADCCSK8    011E78+  ADRDWR1    011B30+  CPBTSRVR  00000701  CPSL1RAM 00000900  DSADDR      010CCC+
ADCCSK9    011E88+  ADRDWR9    011B3E+  CPCKSUM   0001000E  CPSL1TYP 00000771  DSADDRH     010CD8+
ADCCSOUT   011DD4+  ADRST      011DB4+  CPDISKRC  0000070E  CPSL2RAM 00000A00  DSADDRV     010CF0+
ADCCSTP    011E8E+  ADSCIO1    011888+  CPDSCVUC  0001006C  CPSL2TYP 00000772  DSCBLNK   00000020
ADCLRPH    011EC4+  ADSCIO9    01189E+  CPDSINIT  00010060  CPSL3RAM 00000B00  DSCCR     0000000D
ADFSEC     011C90+  ADSECIO    01187E+  CPDSKIO   00000718  CPSL3TYP 00000773  DSCDIFF   00000020
ADFSEC1    011C9E+  ADSECR     011B50+  CPDSPFLG  00000766  CPSL4RAM 00000C00  DSCELLW   00000006
ADFSEC2    011CAC+  ADSECR1    011B62+  CPDSPUTC  00010064  CPSL4TYP 00000774  DSCELLY   0000000A
ADFSEC8    011CB0+  ADSECR2    011B72+  CPDSPUTS  00010068  CPSL5TYP 00000775  DSCESC    0000001B
ADFSEC9    011CB4+  ADSECR9    011B74+  CPEXTCRT  00000F00  CPSTACK  00000F00  DSCLAL      010BE0+
ADFSNRV   00000064  ADSEEK     011D6C+  CPFBLKIO  00010036  CPSYSRST 00010004  DSCLCA    00000061
ADI5SSSD   011AAE+  ADSEEK1    011DFC+  CPFBOOT   00010032  CPSYSST  00000F01  DSCLCZ    0000007A
ADILVTB    011AE4+  ADSETPH    011EB8+  CPFBOOTJ  00010030  CPTPRNBR 0000070D  DSCLEL      010C1E+
ADINIT     011AF4+  ADSIO1     011DDA+  CPFBPS    00000736  CPUNIQID 00010008  DSCLEL1     010C34+
ADINLV     011B40+  ADSIO2     011DF0+  CPFDVSZ   00000734  CPUSERID 00000720  DSCLEL2     010C3A+
ADINLV9    011B4C+  ADSIO3     011DF2+  CPFINIT   0001003E  CPUSERNM 00000726  DSCLES      010BE4+
ADMTROF    011B04+  ADSK1      011D70+  CPFINLV   00000730  CPWNDRCD 00000740  DSCLES1     010BF2+
ADMTRON    011B0A+  ADSK2      011D76+  CPFOFST   0000073B  CRDAM    000000C0  DSCLES2     010C06+
ADRCAD    00000400  ADSK3      011D8C+  CPFSCTIO  0001003A  CRDSEC   00000080  DSCLES3     010C0A+
ADRCRD    00000400  ADSK4      011DA8+  CPFSPD    0000073A  CRDTRK   000000E0  DSCLES9     010C1C+
ADRCSC    00007FFF  ADSK9      011DAC+  CPFSPT    00000738  CRESTORE 00000000  DSCLRH      010C5C+
ADRCSK    00000100  ADTBLOF    011ED8+  CPFTPS    00000739  CSATTR1  00000010  DSCLRH1     010C72+
ADRD4      011C1E+  ADTBLON    011ED0+  CPFTYP    0000073C  CSATTR2  00000011  DSCLRH2     010C84+
ADRD4B     011C1E+  ADWAIT1    011C72+  CPIOBUF   00000D00  CSBPCH   00000006  DSCLRH3     010C8C+
ADRD4E     011C32+  ADWAIT2    011C76+  CPISTACK  00000FFC  CSDATA   00000012  DSCLRH4     010C90+
ADRD5      011C32+  ADWAITB    011C6E+  CPIVEC1   00010070  CSEEK    00000010  DSCLRH5     010C96+
ADRD5B     011C32+  ADWAITE    011C90+  CPIVEC2   00010074  CSFRSTCH 00000008  DSCLRH6     010C9C+
ADRD5E     011C4E+  AMBUF     00000E82  CPIVEC3   00010078  CSLASTCH 0000000A  DSCLRV      010C3E+
ADRD6      011C4E+  AMCHKSM   00000003  CPIVEC4   0001007C  CSLPCH   00000004  DSCLRV1     010C52+
ADRD61     011C4E+  AMSEC     00000002  CPIVEC5   00010080  CSMASK   0000000C  DSCRSAD     010BCC+
DSCRSD      010B46+ DSSHOW9     010B1A+ FDCSTPOT 00000075  FDSEEK1     0116EE+ IOBOOTSW 00030F61
DSCRSH      010B6C+ DSSHW71     010ADA+ FDCSTRR  00000010  FDSEEK8     011706+ IOPBASE  00030000
DSCRSH1     010B6E+ DSSHW72     010AE4+ FDCTRKR  00000012  FDSEEK9     011708+ IT01        01054A+
DSCRSL      010B5E+ DSSHW73     010AEE+ FDEBLCK     0117D8+ FDSIO1      01146E+ IT02        010558+
DSCRSR      010B1C+ DSSHW74     010AF8+ FDEBUSY     011802+ FDSIO2      011482+ IT99        010562+
DSCRSU      010B32+ DSSHW75     010B02+ FDECRC      0117D8+ FDSIO9      011486+ IVLVL1   00000064
DSCSETH     010D4E+ DSSHW76     010B0C+ FDEHERR     0117EA+ FDSKST1     0117BA+ IVLVL2   00000068
DSCSETV     010D84+ DSSHWCH     010A64+ FDENRDY     01180E+ FDSKSTA     0117B6+ IVLVL3   0000006C
DSCTBL      010D08+ DSST0       010A0E+ FDEOPCD     0117E4+ FDSWSTA     0117AA+ IVLVL4   00000070
DSCTL       010A22+ DSSTBL      010D26+ FDEPROT     0117F0+ FDTMOHI  00000004  IVLVL5   00000074
DSCURS      010B9A+ DSTAB       010CA6+ FDERNF      0117FC+ FDTMOLO  00007FFF  IVLVL6   00000078
DSCURS0     010B9C+ DSTAB1      010CC0+ FDERSTA     0117D4+ FDWRDY      0116B0+ IVLVL7   0000007C
DSCURS1     010BBA+ DSWNDH      010D2A+ FDESEEK     0117F6+ FDWRDY1     0116BE+ JUMPTO   00004EF9
DSCURS2     010BBE+ DSWNDV      010D60+ FDEUNIT     0117DE+ FDWRDY9     0116D2+ KBBBUFR  0000000C
DSCURS3     010BC0+ DSWRAP      010B8C+ FDGETADR    0115E0+ FHLD     00000008  KBBFLGS  00000000
DSCVTU1     0109C4+ DSWRAPX     010B78+ FDI8SS      0114CC+ FINTIDXP 00000004  KBBFRNT  00000002
DSCVTUC     0109B4+ DTAS     00000005   FDI8SSDD    0114B0+ FINTIMM  00000008  KBBLEN   000000F4
DSDECX      010B60+ DTC5     00000004   FDI8SSSD    011492+ FINTNRDY 00000002  KBBREAR  00000006
DSDECY      010B7E+ DTC8     00000003   FDINCT1     01166A+ FINTRDY  00000001  KBBSRSV  0000000A
DSDEFOF  00000060   DTLOCL   00000001   FDINCT2     011690+ FL1         0105B2+ KBCC600  00000017
DSESC       010A3E+ DTNDEV   00000000   FDINCT5     0116A0+ FL2         0105B8+ KBCCBRK  00000008
DSESC1      010A44+ DTOMNI   00000002   FDINCT9     0116AC+ FL3         0105C4+ KBCCGO   00000009
DSESC2      010A54+ ECHOED   000000C0   FDINCTS     011658+ FL4         0105CE+ KBCCNTL  000000FD
DSETBL      010D14+ ECHOOP   00000002   FDINIT      0114DA+ FL5         0105DC+ KBCCOFF  00000002
DSEXIT      010A08+ ENDOP    00000010   FDLCMD      011500+ FLASH       010596+ KBCLCA   00000061
DSHOMEH  0008D55E   FBOOT       01140E+ FDLCMD1     011504+ FMPS     00000010  KBCLCZ   0000007A
DSHOMEV  0008D506   FBOOT1      011430+ FDLY     00000004  FSDCMPEN 00000002  KBCLOCK  000000FC
DSINCX      010B20+ FBOOT90     01144A+ FDMTROF     0114EE+ FSDCPM   00000008  KBCNOCH  000000FF
DSINCY      010B48+ FDBLKIO     01144C+ FDNRDY      011768+ FSTPRT10 00000002  KBCQMRK  0000003F
DSINIT      01096E+ FDCAD    00000010   FDNRDY1     011774+ FSTPRT15 00000003  KBCQUAL  0000007F
DSINIT1     010992+ FDCCMDR  00000010   FDNRDY2     011782+ FSTPRT3M 00000000  KBCSHFT  000000FE
DSINIT2     01099A+ FDCCRD      011594+ FDOKSTA     0117D2+ FSTPRT6M 00000001  KBDSINT  00000700
DSKREAD  00000032   FDCCRD1     0115AC+ FDRCDOR  00000004  FUPDTTRK 00000010  KBFCLOS  00000002
DSKWRIT  00000033   FDCCRD2     0115B0+ FDRCRD   00000004  FVERIFY  00000004  KBFCNTL  00000004
DSMAXXH  000002CF   FDCCRD3     0115B8+ FDRCSK   00000004  GAVEUP   00000080  KBFEMTY  00000001
DSMAXXV  0000022D   FDCCRD4     0115C2+ FDRDSTA     01178C+ GDATA    00000016  KBFFULL  00000000
DSMAXYH  0000022F   FDCCRD5     0115CA+ FDRDWR      011528+ GOTOBT      010496+ KBFLOCK  00000005
DSMAXYV  000002CF   FDCCRD6     0115D6+ FDRDWR1     011542+ GOTOBT1     010498+ KBFSHFT  00000003
DSNXTST     0109FE+ FDCCRST     011734+ FDRDWR9     011554+ GOTOBT2     0104A4+ KBGCHR1     0107CC+
DSPBASE  00080000   FDCCSIN     011744+ FDRST       011716+ GOTOBT3     0104B4+ KBGCHR2     0107E8+
DSPEND   0008E000   FDCCSK      011754+ FDRST0      01171E+ GRAPHIC  00000001  KBGCHR3     010800+
DSPLEN   0000E000   FDCCSK1     01175C+ FDRST1      011722+ HDRERR   00000083  KBGETCH     0107C0+
DSPST1      0109CA+ FDCCSK2     011764+ FDRST2      01172E+ INCTEST     010544+ KBGETKY     010808+
DSPST9      0109D2+ FDCCSOT     01174C+ FDRSTW      01170C+ INITOP   00000020  KBINIT      010722+
DSPUTCH     0109D8+ FDCCSTP     01173C+ FDRWST1     01179C+ INTDC0      010604+ KBINIT1     01076E+
DSPUTST     0109C6+ FDCDATR  00000016   FDRWSTA     011790+ INTDC1      01061E+ KBINTR      010780+
DSRESET     010A04+ FDCLCT8     011650+ FDSECIO     011464+ INTKYBD     0105E8+ KBINTR1     0107A0+
DSRTRN      010B36+ FDCLCT9     011654+ FDSECR      01155A+ INTLVL7     0105E6+ KBINTR2     0107AE+
DSRTRN1     010B38+ FDCLCTS     01160C+ FDSECR1     011564+ INTOMNI     010616+ KBINTR9     0107BA+
DSSHOW1     010A74+ FDCRST   00000007   FDSECR2     011568+ INTSLOT     010630+ KBMSK40  0000001F
DSSHOW2     010A78+ FDCSECR  00000014   FDSECR3     011584+ INTTIMR     0105FA+ KBPRO1      010836+
DSSHOW3     010AAA+ FDCSEEK  00000015   FDSECR9     01158C+ INUSE    00000085  KBPRO2      010850+
DSSHOW6     010AC0+ FDCSTP   00000035   FDSECRW     01155A+ INVCURS  00000003  KBPRO3      01085A+
DSSHOW7     010ACA+ FDCSTPIN 00000055   FDSEEK      0116DA+ IOBEEPFQ 00030F71  KBPRO9      01085C+
KBPROKY     010818+ LDSYNC1     01108E+ NIRQ2     00000005  RAMKBBUF  00000300  SBSTROB     011220+
KBPUT1      010872+ LDSYNC2     011098+ NIRQ3     00000006  RAMKBLEN  00000100  SBUSER      010456+
KBPUT2      01087C+ LDSYNC3     0110AA+ NIRQ4     00000007  RAMLEN    00001000  SBUSY     00000000
KBPUTCH     01085E+ LDSYNC5     0110B6+ NNMI1     00000000  RAMMXBUG  00000400  SBW1        01122C+
KBQUAL      010884+ LDSYNC6     0110BA+ NNMI2     00000001  RAMSIZ9     01058C+ SBWAIT      011228+
KBQUAL1     01088E+ LDSYNC9     0110C4+ NNMI3     00000002  RAMSIZE     010564+ SBWEXIT     01123C+
KBQUAL2     010898+ LDWAIT      011016+ NNMI4     00000003  RAMTST1     0101C2+ SC10        01128C+
KBQUAL3     0108A0+ LDWAIT1     01101A+ NOBUFR    00000092  RAMTST2     0101E4+ SC12        01129C+
KBQUAL8     0108AA+ LDWAIT2     011024+ NOSCROLL  00000005  RAMWKSTA  00000700  SC20        0112CC+
KBQUAL9     0108AC+ LDWIO1      010FAE+ NOSOCKT   00000082  RBDBLK    FFFFFFFF  SC30        0112F6+
KBRCMND   00030F05  LNBUF2    00000056  NOTRANS   00000090  RBDOPCO   FFFFFFFD  SC32        0112FC+
KBRCNTL   00030F07  LONGCMDS    011356+ OBOOT       0110CC+ RBDUNT    FFFFFFFE  SC40        01130A+
KBRDATA   00030F01  LS1SD2SD  00000004  ODBLK1      01115C+ RBUSY     FFFFFFEE  SC50        01131E+
KBRSTAT   00030F03  LS8INMIN  00000005  ODBLK2      011170+ RDYADR    00030F7F  SC60        01132C+
KBRTABLE    01090E+ LSDRQ     00000000  ODBLK3      011180+ RECVOP    000000F0  SC70        01133A+
KBSTABLE    0108AE+ LSDSKCHG  00000006  ODBLKIO     011130+ RERRUNOW  FFFFFFC0  SCERR1      011344+
LBOOT       010EFA+ LSFMMFM   00000007  ODCMD1      011114+ RESTSKT   000000A0  SCERR2      01134A+
LBOOT10     010F02+ LSINT     00000001  ODCMD2      011126+ RGOOD     00000000  SCERR3      01134E+
LBOOT30     010F18+ LSTRR     00000000  ODCMD9      01112A+ RHDR      00000006  SCEXIT      011352+
LBOOT80     010F30+ MARCH       010518+ ODCOMND     0110E6+ RHDSKLN   0000000A  SCMD2       0113DC+
LBOOT90     010F6E+ MEMCLR      010214+ ODDSK1      0111A0+ RHDSKRC   0000000C  SCMD3       0113EE+
LC8INMIN  00000006  MEMTEST     010204+ ODDSK2      0111B6+ RHPKTLN   00000008  SCMD4       011400+
LCDE0     00000001  MOTOROFF  00000010  ODDSK3      0111C4+ RHPKTRC   00000006  SCPT5SD   00000010
LCDE1     00000004  MOTORON   00000012  ODDSK4      0111E6+ RHSOR     00000007  SCPT8DD   0000001A
LCFLP8IN  00000006  MR1         01051C+ ODDSK5      0111EA+ RHWRERR   FFFFFFFC  SCPT8SD   0000001A
LCFLPSD1  00000000  MR2         010526+ ODDSK6      0111F0+ RLOSTDEV  FFFFFFFB  SCRCERR   00000003
LCFMMFM   00000007  MR3         010536+ ODDSK9      0111F2+ RNOTRDY   FFFFFFEC  SDRQ      00000001
LCMD1       011362+ MRERR       010542+ ODDSKIO     011190+ ROMBASE   00010000  SDTOVER   00000002
LCMD3       011370+ MSG1        01065A+ ODDW      00000024  ROMEND    00012000  SDTUNDR   00000002
LCMD4       01138A+ MSG10       0106B2+ ODDWHI    00000025  ROMLEN    00002000  SELBOOT     0103C0+
LCMD5       011392+ MSG11       0106D1+ ODDWLO    00000026  ROMTST      0104D4+ SENDOP    00000040
LCMD6       01139E+ MSG12       0106D9+ ODVALID   0000002C  ROMTST1     0101A6+ SETGO       01123E+
LCMD7       0113AC+ MSG13       0106DF+ ODWRAD    00000028  RPTST1      010376+ SETGO1      011262+
LCMDERR     0113BE+ MSG19       0106E6+ OFF       00000000  RPTST2      010396+ SETINTV     010244+
LCMDEX      0113C6+ MSG2        010680+ ON        00000001  RPTST8      01039E+ SETMB       010232+
LCMDOK      0113C2+ MSG3        010683+ PHASE0OF  00000000  RPTST9      0103B6+ SETRECV     011252+
LCMDR     00000000  MSG30       0106F2+ PHASE0ON  00000002  RPTSTAT     010366+ SETUP       0100B4+
LCMOTORO  00000005  MSG31       0106FF+ PHASE1OF  00000004  RRNF      FFFFFFED  SETUP1      0100C2+
LDBLKIO     010F8C+ MSG32       010708+ PHASE1ON  00000006  RSEEKERR  FFFFFFEF  SETUP2      0100C6+
LDDSK1      01104E+ MSG4        010695+ PHASE2OF  00000008  RT1         0104D8+ SETUP3      0100CA+
LDDSK1A     011054+ MSGCPY      010090+ PHASE2ON  0000000A  RT1ERR      0101DA+ SHDLDD    00000005
LDDSK2      01105C+ MXBBASE   00020000  PHASE3OF  0000000C  RT2ERR      0101FA+ SHDR      0000000E
LDDSK3      011064+ MXBEND    00022000  PHASE3ON  0000000E  RWRPROT   FFFFFFF0  SHFMLN    00000014
LDDSK9      01107A+ MXBENTRY  00020008  PHASEOFF  00000000  SBBOOT      010446+ SHORTCMD    0113CA+
LDDSKIO     011036+ MXBINIT   00020004  PHASEON   00000002  SBDEBUG     0103DC+ SHPKTRC   0000000E
LDGETBB     010F70+ MXBLEN    00002000  PRIOR     00000001  SBERR       011214+ SHTOLN    00000012
LDGETBX     010F8A+ NBLK5SD   00000118  PROMLEVL  00000006  SBEXIT      011218+ SINDEX    00000001
LDRIO1      010FC6+ NBLK8DD   000003E9  PROMVERS  00000000  SBFLPY      010414+ SLOT1AD   00030001
LDRIO3      010FCC+ NBLK8SD   000001F4  Q6H       0000001A  SBFLPY1     010422+ SLOTADR     010648+
LDRTRN      010FE2+ NBUF1     00000D00  Q6L       00000018  SBFLPY2     01043C+ SLOTID      010278+
LDSEND      010FEC+ NBUF2     00000E02  Q7H       0000001E  SBFLPY3     010442+ SLOTID1     010286+
LDSEND0     010FFA+ NDEV1AD   00030001  Q7L       0000001C  SBLOCAL     010408+ SLOTID2     0102D4+
LDSEND1     011000+ NIBL        0118AE+ RAMBASE   00000000  SBMSG       010448+ SLOTID3     0102EA+
LDSYNC      011082+ NIRQ1     00000004  RAMEND    00001000  SBOMNI      0103FC+ SLOTID8     010302+
SLOTID9    01030C+  SRAMRD6  00000080  TCDTALN  00000020  USRBASE  0008E000  WRBITOFS 0000001A
SLOTIDA    010340+  SRAMWT   000000C0  TCHDRLN  00000022  VERT     00000000  WRCHARPT 00000000
SLOTIDB    01035A+  SRECTYP  00000005  TCMD     00000018  VIABASE  00030F00  WRCURADR 00000008
SLOTPTR  00030100   SRNF     00000004  TCOP     00000018  WAITING  000000FF  WRCURSX  00000016
SLTADOFS 00000200   SSEEKERR 00000004  TCRADHI  00000019  WALKBIT    0104E6+ WRCURSY  00000018
SLTSTAD  00030A01   STRADR   00030FA1  TCRADLO  0000001A  WB1        0104F0+ WRGRORGX 0000001C
SNDCMDS    0112D8+  STRK0    00000002  TCSOCK   0000001C  WB2        0104F4+ WRGRORGY 0000001E
SNDREST    0112AE+  STROBIT    0111FA+ TIMEOUT  00000091  WB3        010502+ WRHOMEOF 0000000C
SNOTRDY  00000007   SUI1       010252+ TKPS5SD  00000023  WBERR      010516+ WRHOMEPT 00000004
SRAMLN4  00000014   SVLCMD   00000000  TKPS8DD  0000004D  WHOOP    00000001  WRLENGTH 00000024
SRAMLN5  0000001C   SWRFAULT 00000005  TKPS8SD  0000004D  WRAPON   00000004  WRLNGTHX 00000012
SRAMLN6  00000020   SWRPROT  00000006  TOINTVL  0000FFFE  WRATTR1  00000020  WRLNGTHY 00000014
SRAMLNW  00000022   TCDADHI  0000001D  TOOLONG  00000081  WRATTR2  00000021  WRRCDLEN 00000023
SRAMRD4  00000000   TCDADLO  0000001E  TRKCNT   00000000  WRBASEX  0000000E  WRSTATE  00000022
SRAMRD5  00000040   TCDEST   00000023  TRKN     00000002  WRBASEY  00000010  ZERORAM    01058E+

0 errors.  4785 lines.
```

```
ASM68K Cross Reference  0.01d     07 Sep 1982          Date: Nov 1, 1982
Cross-Reference Listing
File ID  CC.PROM.TEXT

ABOOT       618  3724*
ABOOT1     3739*
ABOOT90    3737  3745  3747  3751*
ADBON      3966* 3971  3974  3977  3980  3983  3986  3989  3997  3999
ADBIO1     3785  3789*
ADBIO9     3787  3792*
ADBLKIO     256  3727  3744  3746  3781*
ADCCRD     4193  4228*
ADCCRD1    4237* 4244
ADCCRD2    4237  4240* 4241
ADCCRD3    4243* 4253  4261
ADCCRD4    4250* 4251
ADCCRD5    4258* 4259
ADCCRD8    4238  4327*
ADCCRD9    4286  4325  4329*
ADCCSIN    3472  4604  4635*
ADCCSK     4661  4680*
ADCCSK1    4692* 4720
ADCCSK2    4695  4703*
ADCCSK3    4701  4708*
ADCCSK4    4709  4712*
ADCCSK5    4713  4716*
ADCCSK8    4694  4722*
ADCCSK9    4686  4729*
ADCCSOUT   4607  4631*
ADCCSTP    4718  4751*
ADCLRFH    4723  4758  4772*
ADFSEC     4191  4417*
ADFSEC1    4422* 4431
ADFSEC2    4424  4431*
ADFSEC8    4435*
ADFSEC9    4430  4437*
ADFSNRV    4415* 4420
ADISSSSD   3735  4016*
ADILVTB    4028  4034*
ADINIT      258  3736  4042*
ADINLV     4167* 4182
ADINLV9    4169  4171*
ADMTROF    3790  3825  4077  4105*
ADMTRON    3820  4075  4111* 4126
ADRCAD     4465* 4467
ADRCRD     4226* 4233
ADRCSC     4179* 4187
ADRCSK     4571* 4575
ADRD4      4276  4337* 4338  4342
ADRD4B     3834  4336*
ADRD4E     3834  4344*
ADRD5      4297  4306  4315  4350* 4357
ADRD5B     3835  4349*
ADRD5E     3835  4361*
ADRD6      4284  4367*
ADRD61     4368* 4369  4371
ADRD62     4375* 4376
```

```
ADRD69    4379  4382*
ADRD6B    3836  4366*
ADRD6E    3836  4384*
ADRDAD    4422  4467* 4580
ADRDAD1   4469* 4477
ADRDAD2   4469  4474* 4475
ADRDAD3   4476* 4484  4491
ADRDAD3A  4480* 4482
ADRDAD3B  4487* 4489
ADRDAD4A  4500* 4502
ADRDAD4B  4511* 4513
ADRDAD4C  4522* 4524
ADRDAD4D  4533* 4535
ADRDAD5A  4504* 4506
ADRDAD5B  4515* 4517
ADRDAD5C  4526* 4528
ADRDAD5D  4537* 4539
ADRDAD6   4549* 4550
ADRDAD7   4556* 4557
ADRDAD8   4470  4545  4552  4559  4562*
ADRDWR    3789  4115*
ADRDWR1   4150* 4155
ADRDWR9   4125  4148  4151  4153  4157*
ADRST     4076  4617*
ADSCIO1   3812  3815*
ADSCIO9   3822  3825*
ADSECIO    257  3729  3811*
ADSECR    3823  4152  4181*
ADSECR1   4190* 4194
ADSECR2   4192  4197*
ADSECR9   4195  4199*
ADSEEK    3821  4147  4573*
ADSEEK1   4622  4648  4657*
ADSETPH   4752  4767*
ADSIO1    4633  4637*
ADSIO2    4643  4646*
ADSIO3    4645  4648*
ADSK1     4575* 4605  4608
ADSK2     4578* 4587
ADSK3     4584  4593*
ADSK4     4602  4607*
ADSK9     4588  4601  4610*
ADTBLOF   4760  4780*
ADTBLON   4754  4777*
ADWAIT1   4396* 4404
ADWAIT2   4397* 4403
ADWAITB   3837  4394* 4726  4756  4762
ADWAITE   3837  4407*
AMBUF     3873* 4429  4485  4596  4600
AMCHKSM   3880*
AMSEC     3879* 3880  4429
AMTRK     3878* 3879  4596  4608
AMVOL     3877* 3878
APLSVAR   3858* 3873  4593  4618  4639  4684
BADDEST   2280*
BADSOCK   2278*
BASERAM   2850*
BLKSZ     2786  2853* 3397  3782
```

```
BPS5ISD   2885* 4022
BFS8IDD   2884* 3058
BPS8ISD   2883* 3044
CFRCINT   2983* 3705
CHEND      351   358*
CHEND1     359*  359
CHERR      337   339   341   343   345   347   349   353*
CHERR1     354*  354
CMDACPT   2271* 2527  2614  2648  2688
CNSTSKT   2297* 2514  2555
CPABLKIO   179*
CPAINIT    181*
CPASCTIO   180*
CPBLKIO    113* 1951  2305  2724  3728
CPBTSLOT   100*  680   687  1948  1967  2303  2720  3724
CPBTSRVR   101*  529   689  1949  1964
CPCKSUM    162*  368
CPDISKRC   109*
CPDSCVUC   189*
CPDSINIT   186*
CPDSKIO    114* 1953  2307  2726  3730
CPDSPFLG   131* 1598
CPDSPUTC   187*
CPDSPUTS   188*
CPEXTCRT   148*  311
CPFBLKIO   176*
CPFBOOT    175*
CPFBOOTJ   174*
CPFBPS     122* 3044  3058  3395  4022  4282  4313
CPFDVSZ    121* 3043  3057  3133  4021  4121
CPFINIT    178*
CPFINLV    120* 4029  4168
CPFOFST    126* 3062  3406  3438  3444  4026
CPFSCTIO   177*
CPFSPD     125* 3061  3414  4025
CPFSPT     123* 3045  3059  3402  3437  4023  4418
CPFTPS     124* 3046  3060  3410  3446  4024
CPFTYP     127* 3447  4027
CFIOBUF    146* 3853
CPISTACK   150*  301
CPIVEC1    192*  436
CPIVEC2    193*
CPIVEC3    194*
CPIVEC4    195*
CPIVEC5    196*
CPIVEC6    197*
CPIVEC7    198*
CPKBGETC   184*
CPKBINIT   183*
CPLBLKIO   171*
CPLBOOT    170*
CPLBOOTJ   169*
CPLDSKIO   172*
CPOBLKIO   166*
CPOBOOT    165*
CPOBOOTJ   164*
CPODSKIO   167*
CPOMNIBF   152* 2320  2359  2401
```

```
CPOMNIRC   110*
CPOMNRAM   139*
CPOSBLK    106*
CPOSDRV    105* 3116
CPOSSLOT   103* 2721  3370  3375  3725
CPOSSRVR   104*
CPROMLVL   161*
CPROMVRS   160*
CPSCNOFS   130* 1383  1514  1659  1722  1758  1795  1805
CPSL1RAM   140* 2850  3374
CPSL1TYP   133*  456  ·601  1936
CPSL2RAM   141*
CPSL2TYP   134*
CPSL3RAM   142*
CPSL3TYP   135*
CPSL4RAM   143*
CPSL4TYP   136*
CPSL5TYP   137*
CPSTACK    147*  382   416
CPSYSRST   158*
CPSYSST    149*  355   372   387   400   408   497   516   538   544
CPTPRNBR   108*  509
CPUNIQID   159*
CPUSERID   116*
CPUSERNM   118*
CPWNDRCD   129* 1390  1394  1438
CRDAM     2980*
CRDSEC    2975* 3282
CRDTRK    2981*
CRESTORE  2967* 3572
CSATTR1   1341*
CSATTR2   1343*
CSBPCH    1337*
CSDATA    1344* 1508  1839  1858
CSEEK     2968* 3576
CSFRSTCH  1338* 1457  1501  1507
CSLASTCH  1339* 1503
CSLPCH    1336* 1512  1646
CSMASK    1340* 1516  1650  1723
CSTEP     2969* 3573
CSTEPIN   2970* 3574
CSTEPOUT  2971* 3575
CSUMRD    3869*
CURSON    1368*
CURTRK    3865* 3869  4598  4619  4623  4640  4685  4692  4693  4698  4700  4703  4706  4716  4722
CWRSEC    2976*
CWRTRK    2982*
DCBLKHI   2218* 2365
DCBLKLO   2217* 2363
DCDRV     2216* 2362
DCLEN     2219* 2366
DCMD      2215* 2216  2217  2218  2219  2361  2406  2550
DEBOP     2292*
DEVADOFS  2862* 3372
DLY100M   4392* 4396
DNIBL     3922* 4256
DNIBL2    3898* 4295
DNIBL3    3899* 4304
```

```
DNIBL4    3900* 4312
DRV0EN    3988*
DRV1EN    3989*
DSADDR    1664  1721  1734  1785*
DSADDRH   1789*
DSADDRV   1787  1799*
DSCBLNK   1328* 1505
DSCCR      632   922   922   925   925   926   926   927   927   927   932   934   935  1326*
DSCDIFF   1331* 1411
DSCELLW   1319* 1509  1581  1619  1640  1765  1840  1859
DSCELLY   1320* 1607  1635  1683  1693  1713  1840  1859
DSCESC    1327* 1455
DSCLAL    1611  1673* 1821
DSCLCA    1329* 1407
DSCLCZ    1330* 1409
DSCLEL    1678  1706* 1822
DSCLEL1   1709  1713*
DSCLEL2   1711  1716*
DSCLES    1678* 1823
DSCLES1   1683*
DSCLES2   1680  1692*
DSCLES3   1693* 1699
DSCLES9   1686  1690  1695  1701*
DSCLRH    1689  1715  1730*
DSCLRH1   1737  1739*
DSCLRH2   1747* 1759
DSCLRH3   1749  1751*
DSCLRH4   1753* 1754
DSCLRH5   1752  1755*
DSCLRH6   1756  1758*
DSCLRV    1698  1710  1719*
DSCLRV1   1725* 1727
DSCRSAD   1645  1663*
DSCRSD    1605* 1815
DSCRSH    1626* 1820
DSCRSH1   1627* 1673
DSCRSL    1616* 1813
DSCRSR    1579* 1817
DSCRSU    1590* 1816
DSCSETH   1385  1829  1839*
DSCSETV   1389  1508  1839  1848  1858* 1858
DSCTBL    1467  1813* 1813  1814  1815  1816  1817  1818
DSCTL     1458  1463*
DSCURS    1396  1585  1600  1610  1620  1629  1634  1636  1645* 1775
DSCURS0   1579  1590  1596  1605  1616  1626  1646* 1706  1716  1773
DSCURS1   1652  1655*
DSCURS2   1654  1657*
DSCURS3   1658* 1660
DSCVTU1   1408  1410  1412*
DSCVTUC    270   629  1407* 1500
DSDECX    1617*
DSDECY    1591  1633*
DSDEFOF   1316* 1383
DSESC     1474* 1827
DSESC1    1477* 1481
DSESC2    1478  1484*
DSETBL    1475  1820* 1820  1821  1822  1823
DSEXIT    1448  1452* 1461  1464  1466  1469
```

```
DSHOMEH    1317* 1830  1831
DSHOMEV    1318* 1849  1850
DSINCX     1460  1580*
DSINCY     1599  1606*
DSINIT      267   443  1382*
DSINIT1    1387  1390*
DSINIT2    1392* 1393
DSKREAD     224* 1996  2738  3742
DSKWRIT     225* 1992  2034  2136  2367  2402  2823  3784  3811
DSMAXXH    1321* 1833
DSMAXXV    1323* 1852
DSMAXYH    1322* 1833  1835
DSMAXYV    1324* 1852  1854
DSNXTST    1447* 1456
DSPBASE      79*   81   360   420   838   842
DSPEND       81*   83   152   361   834
DSPLEN       80*   81
DSPST1     1422* 1425
DSPST9     1423  1427*
DSPUTCH     268   550   631   633  1424  1436*
DSPUTST     269   446   448   450   541   547   552   559   585   621   623   627   674  1420*
DSRESET    1450* 1482  1486
DSRTRN     1596* 1818
DSRTRN1    1584  1597*
DSSHOW1    1502  1505*
DSSHOW2    1504  1507*
DSSHOW3    1525* 1533
DSSHOW6    1518  1540*
DSSHOW7    1546* 1572
DSSHOW9    1534  1573*
DSSHW71    1550  1552*
DSSHW72    1553  1555*
DSSHW73    1556  1558*
DSSHW74    1559  1561*
DSSHW75    1562  1564*
DSSHW76    1548  1565  1567*
DSSHWCH    1459  1500*
DSST0      1455* 1826
DSSTBL     1443  1826* 1826  1827
DSTAB      1763* 1814
DSTAB1     1770  1773*
DSWNDH     1384  1829*
DSWNDV     1388  1848*
DSWRAP     1631  1638*
DSWRAPX    1618  1631*
DTAS        214*  488   607  3447  4027
DTC5        213*
DTC8        212*  494   605
DTLOCL      210*  482  1940
DTNDEV      209*
DTOMNI      211*  531
ECHOED      514  2272*
ECHOOP      511  2293*
ENDOP      2291*
FBOOT       615  2720*
FBOOT1     2735*
FBOOT90    2733  2741  2743  2746*
FDBLKIO     253  2723  2740  2742  2785*
```

```
FDCAD     2955* 2959  2960  2961  2962  2963
FDCCMDR   2959* 3282  3582  3587  3592  3597  3607  3705
FDCCRD    3210  3277*
FDCCRD1   3284* 3284
FDCCRD2   3286* 3287
FDCCRD3   3291* 3296  3300
FDCCRD4   3295*
FDCCRD5   3293  3299*
FDCCRD6   3297  3306*
FDCCRST   3524  3561  3582*
FDCCSIN   3457  3592*
FDCCSK    3516  3527  3603*
FDCCSK1   3607*
FDCCSK2   3614*
FDCCSOT   3597*
FDCCSTP   3587*
FDCDATR   2963* 3299  3604
FDCLCT8   3415  3421*
FDCLCT9   3413  3419  3423*
FDCLCTS   3108  3394* 4124
FDCRST    3572* 3582
FDCSECR   2962* 3281
FDCSEEK   3576* 3607
FDCSTP    3573* 3587
FDCSTPIN  3574* 3592
FDCSTPOT  3575* 3597  •
FDCSTRR   2960* 3286  3306  3493  3632  3641  3661  3666
FDCTRKR   2961*
FDEBLCK   3132  3134  3421  3470  3684* 4120  4122
FDEBUSY   3656  3673  3705*
FDECRC    3651  3671  3683*
FDEHERR   3693*
FDENRDY   3658  3675  3709*
FDEOPCD   2825  3690* 3786  3813
FDEPRGT   3663  3696*
FDERNF    3653  3702*
FDERSTA   3679* 3685  3688  3691  3694  3697  3700  3703  3707  3710
FDESEEK   3669  3699*
FDEUNIT   3687*
FDGETADR  2828  3039  3053  3068  3127  3367* 3816  4017  4042  4115
FDI8SS    3047  3061*
FDI8SSDD  3052*
FDI8SSSD  2731  3038*
FDINCT1   3440  3444*
FDINCT2   3453  3461*
FDINCT5   3448  3469*
FDINCT9   3442  3456  3459  3465  3474*
FDINCTS   3164  3436* 4154
FDINIT     255  2732  3068*
FDLCMD    3108* 3135
FDLCMD1   2833  3074  3113*
FDLY      2997*
FDMTROF   2839  3076  3098* 3168
FDNRDY    3304  3583  3588  3593  3598  3608  3623*
FDNRDY1   3627* 3628  3629
FDNRDY2   3631*
FDOKSTA   3659  3677*
FDRCDOR   3199* 3205
```

```
FDRCRD     3198* 3202
FDRCSK     3512* 3520
FDRDSTA    3217  3641*
FDRDWR     2788  3127*
FDRDWR1    3159* 3166
FDRDWR9    3157  3160  3163  3165  3168*
FDRST      3075  3464  3554*
FDRST0     3548  3557*
FDRST1     3559* 3563
FDRST2     3560  3565*
FDRSTW     3546*
FDRWST1    3655*
FDRWSTA    3650*
FDSECIO     254  2725  2823*
FDSECR     2837  3201*
FDSECR1    3205* 3214
FDSECR2    3207* 3212
FDSECR3    3217*
FDSECR9    3209  3218  3221*
FDSECRW    3161  3179*
FDSEEK     2834  3156  3514*
FDSEEK1    3522* 3529
FDSEEK8    3523  3526  3533*
FDSEEK9    3515  3518  3535*
FDSIO1     2824  2827*
FDSIO2     2837*
FDSIO9     2835  2839*
FDSKST1    3664  3668*
FDSKSTA    3458  3517  3531  3555  3666*
FDSWSTA    3547  3661*
FDTMOHI    3485* 3491
FDTMOLO    3486* 3490  3624
FDWRDY     3208  3455  3488* 3514  3522  3525  3559
FDWRDY1    3493* 3494  3495
FDWRDY9    3496  3499*
FHLD       2987*
FINTIDXP   3003*
FINTIMM    3004* 3705
FINTNRDY   3002*
FINTRDY    3001*
FL1         838*
FL2         839*  841
FL3         843*  845
FL4         847*  847
FL5         850*  850
FLASH       356   373   388   401   409   498   517   833*
FMPS       2996*
FSDCMPEN   2999*
FSDCPM     2998*
FSTPRT10   2993*
FSTPRT15   2994* 3572
FSTPRT3M   2991*
FSTPRT6M   2992* 3573  3574  3575  3576
FUPDTTRK   2990* 3573  3574  3575
FVERIFY    2989* 3572  3573  3574  3575  3576
GAVEUP     2274*
GDATA      2239* 2487  2621  2623
GOTOBT      616   619   671*
```

```
GOTOBT1    612   673*
GOTOBT2    671   677*
GOTOBT3    679   682*
GRAPHIC   1367*
HDRERR    2277*
INCTEST    406   785*
INITOP     506  2290*
INTDC0     275   876*
INTDC1     273   889*
INTKYBD    277   863*
INTLVL7    278   858*
INTOMNI    274   883*
INTSLOT    272   897*
INTTIMR    276   870*
INUSE     2279*
INVCURS   1369*
IOBEEPFQ   204*
IOBOOTSW   203*  570  1386
IOPBASE     76*   77
IT01       788*  791
IT02       796*  800
IT99       797   802*
IVLVL1      88*  437
IVLVL2      89*
IVLVL3      90*
IVLVL4      91*
IVLVL5      92*
IVLVL6      93* 1025
IVLVL7      94*
JUMPTO     222*  239   245   251
KBBBUFR    956*  957  1020  1112  1117  1213  1216
KBBFLGS    952*  953  1014  1059  1101
KBBFRNT    953*  954  1021  1110  1220
KBBLEN     957* 1112  1213
KBBREAR    954*  955  1124  1207
KBBSRSV    955*  956  1119
KBCC600    998* 1031
KBCCBRK    999* 1032
KBCCGO    1000* 1035
KBCCNTL    976* 1239
KBCCOFF    997* 1013
KBCLCA     982* 1177
KBCLCZ     983* 1179
KBCLOCK    977* 1244
KBCNOCH    978* 1076
KBCOMRK    984* 1168
KBCQUAL    974* 1156
KBCSHFT    975* 1234
KBDSINT   1002* 1121
KBFCLOS    963* 1060  1063  1161  1250
KBFCNTL    965* 1166  1241
KBFEMTY    962* 1016  1105  1126  1224
KBFFULL    961* 1186  1222
KBFLOCK    966* 1175  1246
KBFSHFT    964* 1069  1236
KBGCHR1   1105* 1106
KBGCHR2   1116  1119*
KBGCHR3   1125  1128*
```

```
KBGETCH    262   628  1099*
KBGETKY   1055  1138*
KBINIT     261   442  1010*
KBINIT1   1034* 1034
KBINTR    1024  1053*
KBINTR1   1062  1068*
KBINTR2   1070  1072*
KBINTR9   1077  1082*
KBMSK40    970* 1170
KBPRO1    1167  1169  1175*
KBFRO2    1171  1176  1178  1180  1186*
KBPRO3    1157  1193*
KBPRO9    1162  1187  1189  1194*
KBPROKY   1078  1156*
KBPUT1    1215  1220*
KBPUT2    1221  1223*
KBPUTCH   1188  1207*
KBQUAL    1193  1234*
KBQUAL1   1235  1239*
KBQUAL2   1240  1244*
KBQUAL3   1237  1242  1250*
KBQUAL8   1251  1254*
KBQUAL9   1245  1253  1255*
KBRCMND    992* 1012
KBRCNTL    993* 1031
KBRDATA    990* 1030  1145
KBRSTAT    991* 1029  1144
KBRTABLE  1071  1287*
KBSTABLE  1068  1181  1264*
LBOOT      596  1936*
LBOOT10   1939* 1944
LBOOT30   1941  1948*
LBOOT80   1963* 2308
LBOOT90   1946  1965  1973  1977  1981  1985  1988*
LC8INMIN  2947*
LCDE0     2944*
LCDE1     2945*
LCFLF8IN  2948* 3114
LCFLPSD1  2943* 3416  3462
LCFMMFM   2949* 3115
LCMD1     2613* 2615  2631
LCMD3     2621*
LCMD4     2624  2629*
LCMD5     2627  2637* 2660
LCMD6     2647* 2649
LCMD7     2655*
LCMDERR   2604  2609  2617  2630  2638  2643  2651  2659  2662* 2676  2681  2689  2692  2705
LCMDEX    2663  2667*
LCMDOK    2657  2665* 2700  2703
LCMDR     2932* 3100  3122  3463
LCMOTORO  2946* 3099  3113
LDBLKIO    247  1950  2024*
LDDSK1    2144* 2146
LDDSK1A   2143  2146*
LDDSK2    2137  2152*
LDDSK3    2156* 2157  2161
LDDSK9    2148  2159  2163*
LDDSKIO    248  1952  2132*
```

```
LDGETBB   1972  1976  1980  1984  1990*
LDGETBX   1998  2001*
LDRIO1    2035  2052*
LDRIO3    2055* 2056  2060
LDRTRN    2046  2058  2062*
LDSEND    2029  2031  2033  2077* 2078  2145
LDSEND0   2090* 2096
LDSEND1   2027  2093* 2142
LDSYNC     480  2178*
LDSYNC1   2182* 2187  2198
LDSYNC2   2184* 2184
LDSYNC3   2186  2190* 2192
LDSYNC5   2188  2194*
LDSYNC6   2191  2197*
LDSYNC9   2195  2201*
LDWAIT    2044  2052  2106* 2152
LDWAIT1   2108* 2108
LDWAIT2   2110  2113* 2114  2116
LDWIO1    2040* 2041  2043
LNBUF2    4011* 4275  4296  4305  4314  4314
LONGCMDS  2373  2425  2599* 2622
LS1SD2SD  2936*
LS8INMIN  2937*
LSDRQ     2934* 3292
LSDSKCHG  2938*
LSFMMFM   2939*
LSINT     2935* 3295  3627
LSTRR     2931* 3291  3627
MARCH      384   397   751*
MEMCLR     407   415*
MEMTEST    398   406*
MOTOROFF  3985* 4105
MOTORON   3986* 4111
MR1        754*  756
MR2        760*  765
MR3        769*  774
MRERR      762   771   776*
MSG1       445   922*
MSG10      626   928*
MSG11      589   929*
MSG12      594   930*
MSG13      599   931*
MSG19      622   932*
MSG2       449   558   925*
MSG3       673   926*
MSG30      546   933*
MSG31      551   934*
MSG32      540   935*
MSG4       584   927*
MSGCPY     282*  447
MX8BASE     70*   72    73    74   428   581
MX8END      72*
MX8ENTRY    74*  586
MX8INIT     73*  427   580
MX8LEN      71*   72
NBLK5SD   2903* 4021
NBLK8DD   2902* 3057
NBLK8SD   2901* 3043
```

```
NBUF1      3853* 3854  4281  4290
NBUF2      3854* 3858  4274  4294  4303  4311
NDEV1AD    2861* 3369
NIBL       3902*
NIRQ1      2874*
NIRQ2      2875*
NIRQ3      2876*
NIRQ4      2877*
NNMI1      2869*
NNMI2      2870*
NNMI3      2871*
NNMI4      2872*
NOBUFR     2284* 2414
NOSCROLL   1371*
NOSGCKT    2276*
NOTRANS    2282* 2467
OBOOT       591  2303*
ODBLK1     2370*
ODBLK2     2368  2376*
ODBLK3     2374  2380*
ODBLKIO     241  2304  2358*
ODCMD1     2331* 2333
ODCMD2     2332  2336*
ODCMD9     2328  2338*
ODCOMND     507   513  2319*
ODDSK1     2405*
ODDSK2     2403  2412*
ODDSK3     2413  2417*
ODDSK4     2422  2425*
ODDSK5     2424  2426*
ODDSK6     2415  2429*
ODDSK9     2410  2435*
ODDSKIO     242   525   527  2306  2400*
ODDW       2254*
ODDWHI     2255*
ODDWLO     2256*
ODVALID    2258* 2360  2409  2412  2417
ODWRAD     2257* 2372  2407  2537  2538
OFF         219*
ON          220*
PHASE0OF   3973* 4657
PHASE0ON   3974*
PHASE1OF   3976* 4658
PHASE1ON   3977*
PHASE2OF   3979* 4659
PHASE2ON   3980*
PHASE3OF   3982* 4660
PHASE3ON   3983*
PHASEOFF   3970* 4774
PHASEON    3971* 4769
PRIOR      3863* 4692  4717
PROMLEVL     56*  236   924
PROMVERS     54*  235   924
Q6H        3997*
Q6L        3996* 4240  4250  4258  4337  4368  4375  4474  4481  4488  4501  4505  4512  4516  4523  4527
           4534  4538  4549  4556
Q7H        3999*
Q7L        3998*
```

```
RAMBASE      58*   60    62    63    64
RAMEND       64*
RAMKBBUF     60* 1011  1054  1100
RAMKBLEN     61*  957
RAMLEN       59*   64
RAMMXBUG     62*  415
RAMSIZ9     812   817*
RAMSIZE     396   421   677   810*
RAMTST1     371   379*
RAMTST2     385   393*
RAMWKSTA     63*  100   101   103   104   105   106   108   109   110   113   114   116   118   120   121
            122   123   124   125   126   127   129   130   131   133   134   135   136   137   139   140
            141   142   143   146   147   148   149   150   379
RBDBLK     2908* 3684
RBDOPCD    2910* 3690
RBDUNT     2909* 3687
RBUSY      2916* 3706
RDYADR     2265* 2476
RECVOP     2288* 2513
RERRUNOW   2919*
RESTSKT    2296* 2539
RGOOD      2907*
RHDR       2223* 2224  2225  2226  2227  2228  2321  2508
RHDSKLN    2227*
RHDSKRC    2228* 2665  2679
RHPKTLN    2226*
RHPKTRC    2224* 2325  2331  2334  2336  2505  2521  2526  2613  2647  2686
RHSOR      2225* 2626  2656  2696
RHWRERR    2911* 3693
RLOSTDEV   2912*
RNOTRDY    2918* 3497  3709
ROMBASE      66*   68   158   159   160   161   162   164   165   166   167   169   170   171   172   174
            175   176   177   178   179   180   181   183   184   186   187   188   189   192   193   194
            195   196   197   198   231
ROMEND       68*  369   811
ROMLEN       67*   68
ROMTST      370   706*
ROMTST1     368*
RPTST1      539   544*  556
RPTST2      545   554*
RPTST8      558*
RPTST9      563*  563
RPTSTAT     530   537*
RRNF       2917* 3702  4327  4380  4435  4562
RSEEKERR   2915* 3699
RT1         708*  711
RT1ERR      381   387*
RT2ERR      395   400*
RWRPROT    2914* 3696
SBBOOT      592   597   619*
SBDEBUG     579*  636
SBERR      2459  2461  2463  2467*
SBEXIT     2465  2469*
SBFLPY      252   577   599*  638
SBFLPY1     604*  611
SBFLPY2     606   615*
SBFLPY3     608   618*
SBLOCAL     246   574   594*  640
```

```
SBMSG      590   595   600   621*
SBOMNI     240   576   589*  642
SBSTROB   2458  2460  2462  2473*
SBUSER     572   583   626*  675
SBUSY     3012* 3286  3655  3672
SBW1      2476* 2478
SBWAIT    2464  2475*
SBWEXIT   2477  2479*
SC10      2521* 2523
SC12      2522  2526*
SC20      2541  2543*
SC30      2557  2560*
SC32      2559  2561*
SC40      2545  2566*
SC50      2573*
SC60      2579* 2581
SC70      2580  2584*
SCERR1    2528  2586  2589*
SCERR2    2518  2575  2591*
SCERR3    2524  2582  2593*
SCEXIT    2530  2587  2590  2592  2595*
SCMD2     2686* 2691  2706
SCMD3     2687  2696*
SCMD4     2698  2702*
SCPT5SD   2891* 4023
SCPT8DD   2890* 3059
SCPT8SD   2889* 3045
SCRCERR   3021* 3213  3650  3670
SDRG      3015*
SDTOVER   3018* 3211
SDTUNDR   3019*
SELBOOT    570*
SENDOP    2289* 2572
SETGO     2485* 2603  2629
SETGO1    2492  2505*
SETINTV    430   436*
SETMB      427*
SETRECV   2497* 2637  2658  2675  2704
SETUP      233   300*  643  4784
SETUP1     306*  306
SETUP2     307*  307
SETUP3     308*  308
SHDLDD    3026*
SHDR      2232* 2233  2236  2237  2567
SHFMLN    2237* 2371  2377  2418  2419  2502
SHORTCMD  2378  2423  2671*
SHPKTRC   2233* 2566  2579  2585  2589
SHTOLN    2236* 2370  2376  2405  2421  2540  2542  2543  2556  2558
SINDEX    3014*
SLOT1AD   2864*
SLOTADR    913* 2025  2133  2179
SLOTID     456*
SLOTID1    460*  503
SLOTID2    477   479   484*
SLOTID3    485   487   490*
SLOTID8    481   497*
SLOTID9    491   493   495   500*
SLOTIDA    515   520*
```

```
SLOTIDB     508   518   525*
SLOTPTR     911*  916
SLTADOFS   2665*
SLTSTAD    2867*
SNDCMDS    2550* 2608  2680
SNDREST    2537* 2642
SNOTRDY    3032* 3493  3657  3674
SR          300  1036  1120  1121  1128  2090  2093  2094  2098  3279  3280  3307  4184  4185  4200  4578
           4579  4581  4689  4690  4724
SRAMLN4    3834*
SRAMLN5    3835*
SRAMLN6    3836*
SRAMLNW    3837*
SRAMRD4    3845*
SRAMRD5    3846*
SRAMRD6    3847*
SRAMWT     3848*
SRECTYP    3027*
SRNF       3024* 3652
SSEEKERR   3023* 3528  3668
STRADR     2264* 2474
STRK0      3017*
STROBIT    2327  2453* 2517  2574
SUI1        439*  440
SVLCMD     2851* 3098  3101  3121
SWRFAULT   3028*
SWRPROT    3030* 3662
TCDADHI    2248* 2491  2501  2537  2554
TCDADLO    2249* 2538
TCDEST     2252* 2563  2625  2655  2697  2702
TCDTALN    2250* 2485  2502  2543  2558  2560
TCHDRLN    2251* 2486  2503  2544  2561
TCMD       2243* 2244  2245  2246  2247  2248  2249  2250  2251  2252  2326  2516  2573
TCOP       2244* 2322  2323  2513  2572
TCRADHI    2245* 2512  2571
TCRADLO    2246*
TCSOCK     2247* 2324  2514  2539  2555
TIMEOUT    2283* 2334  2593
TKPS5SD    2897* 4024
TKPS8DD    2896* 3060
TKPS8SD    2895* 3046
TOINTVL    2266* 2329  2475  2519  2578
TOOLONG    2275*
TRKCNT     3862* 4688  4708  4710  4719
TRKN       3864*
USRBASE      83*  393   520   682  1963  2735  2744  3739  3748
VERT       1366* 1517  1651  1679  1708  1786
VIABASE      77*  203   204   310   835
WAITING    2270* 2325  2331  2505  2521  2566  2579
WALKBIT     380   394   720*
WB1         724*  740
WB2         725*  729
WB3         732*  736
WBERR       727   734   742*
WHOOP      2294*
WRAPON     1370*
WRATTR1    1363*
WRATTR2    1364* 1517  1651  1679  1708  1786
```

```
WRBASEX   1354*
WRBASEY   1355*
WRBITOFS  1360* 1515  1649  1665
WRCHARPT  1350* 1395  1439
WRCURADR  1352* 1511  1648  1666
WRCURSX   1358* 1580  1582  1597  1617  1619  1627  1632  1663  1707  1763  1774
WRCURSY   1359* 1606  1608  1628  1633  1635  1682  1692
WRGRORGX  1361*
WRGRORGY  1362*
WRHOMEOF  1353* 1789  1799
WRHOMEPT  1351* 1785
WRLENGTH  1376* 1391  1837  1856
WRLNGTHX  1356* 1583  1639  1719  1730  1769
WRLNGTHY  1357* 1609  1684  1694
WRRCDLEN  1374*
WRSTATE   1373* 1441  1447  1450
ZERORAM    362   418   422   825*  827
```