

# 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.



```

1* CC.PROM.TEXT -----
2*
3* CC.PROM -- Corvus CONCEPT Workstation PROM
4*
5* Copyright 1982 Corvus Systems, Inc.
6* San Jose, California
7*
8* All Rights Reserved
9*
10* v 0.1 05-04-82 LEF Original program
11* v 0.2 05-27-82 LEF Add keyboard driver (kb)
12* Add display driver (mb)
13* Add 8" Corvus floppy disk driver (kb)
14* Add 8" Corvus floppy disk boot
15* Finds first local disk for booting
16* OMNINET disk driver modifications
17* v 0.3 06-21-82 LEF Add MACSBUG interface
18* Add 5" floppy driver
19* Add 5" floppy boot
20* Finds first floppy disk for booting
21* Local disk driver modifications
22* OMNINET disk driver modifications
23* v 0.4 06-29-82 LEF Add time out to OMNINET short commands
24* v 0.5 08-18-82 KB Modified MacsBug interface
25* Modified Apple 5" floppy driver
26* v 0.6 10-29-82 LEF Swap BACKSPACE and \ keys
27* Modified display driver
28*
29* -----
30*
31* File: CC.PROM
32* Date: 29-Oct-82
33* By: L. Franklin changes by K. Hall
34*
35*
36* Stag PROM checksums. : : : :
37* : H : 362F :
38* : : : :
39* : : : :
40* : : : :
41* : L : F310 :
42* : : : :
43* : : : :
44*
45*
46*

```

```

48*      include 'CC FROM.EQ'      ,PROM equates
49*
50*      , File: CC.PROM.EQ.TEXT
51*      , Date: 28-Oct-82
52*
53*
00000000      54* PROMvers equ 0      ,Current PROM version number
55*      ,ROMvers equ 15      ,Temporary PROM version number ("1")
00000006      56* PROMlevl equ 6      ,Current PROM level number
57*
00000000      58* RAMbase equ $00000      ,Base address of low RAM
00001000      59* RAMlen equ $1000      ,Length of low RAM (4k bytes)
00006300      60* RAMkbbuf equ RAMbase+$300      ,Start of keyboard buffer
00006100      61* RAMkblen equ $100      ,Length of keyboard buffer
00000400      62* RAMmabug equ RAMbase+$400      ,Start of MACSBUG RAM
00000700      63* RAMwksta equ RAMbase+$700      ,Start of workstation RAM
00001000      64* RAMlend equ RAMbase+RAMLen      ,End address + 1 of low RAM
65*
00010000      66* ROMbase equ $10000      ,Base address of workstation PROM
00010000      67* ROMlen equ $2000      ,Length of workstation PROM
00011000      68* ROMlend equ ROMbase+ROMlen      ,End address + 1 of workstation PROM
69*
00030000      70* MXBbase equ $20000      ,Base address of MACSBUG (if present)
00030000      71* MXBlen equ $2000      ,Length of MACSBUG
00032000      72* MXBend equ MXBbase+MXBlen      ,End address + 1 of MACSBUG
00020004      73* MXBinit equ MXBbase+4      ,Address of MacsBug init vector      0.5
00020006      74* MXBentry equ MXBbase+8      ,Address of MacsBug entry vector      0.5
75*
00030000      76* IOPbase equ $30000      ,Base address of I/O page
00030F00      77* VIAbase equ IOPbase+$0F00      ,Base address of VIA registers
78*
00080000      79* DSPbase equ $80000      ,Base address of display bufter
00080000      80* DSPlen equ $0E000      ,Length of display buffer
00080000      81* DSPend equ DSPbase+DSPlen      ,End address + 1 of display buffer
82*
00080000      83* USRbase equ DSPend      ,Base address of user RAM
84*
85*
86*      , Corvus CONCEPT Workstation interrupt vector definition
87*
00000064      88* IVIv11 equ $64      ,level 1 interrupt vector (SLOTS)
00000068      89* IVIv12 equ $68      ,level 2 interrupt vector (DC1)
0000006C      90* IVIv13 equ $6C      ,level 3 interrupt vector (GMNINET)
00000070      91* IVIv14 equ $70      ,level 4 interrupt vector (DC0)
00000074      92* IVIv15 equ $74      ,level 5 interrupt vector (TIMER)
00000078      93* IVIv16 equ $78      ,level 6 interrupt vector (KYDB)
0000007C      94* IVIv17 equ $7C      ,level 7 interrupt vector
95*

```

```

97* ,
98* , Corvus CONCEPT Workstation static RAM address definition
99* ;
0000700 100* CPBtslot equ  RAMWksta+5000 ;(700-700) boot slot number
0000701 101* CPBtsrvr equ  RAMWksta+5001 ;(701-701) boot server number
102* ;(702-705)
0000706 103* CPosslot equ  RAMWksta+5006 ;(706-706) OS volume slot number
0000707 104* CPosssrvr equ  RAMWksta+5007 ;(707-707) OS volume server number
0000708 105* CPosdrv equ  RAMWksta+5008 ;(708-708) OS volume drive number
0000709 106* CPosblk equ  RAMWksta+5009 ;(709-709) OS volume block number
107* ;(70C-70C)
000070D 108* CPTprnbr equ  RAMWksta+500D ;(70D-70D) OMNINET transporter number
000070E 109* CPdiskRC equ  RAMWksta+500E ;(70E-70E) disk controller return code
000070F 110* CPomniRC equ  RAMWksta+500F ;(70F-70F) OMNINET return code
111* ;
112* ;(710-713)
0000714 113* CPBklio equ  RAMWksta+5014 ;(714-717) boot disk blk i/o subr pointer
0000718 114* CPdiskio equ  RAMWksta+5018 ;(718-71B) boot disk i/o subr pointer
115* ;
0000720 116* CPuserID equ  RAMWksta+5020 ;(720-721) user ID
117* ;(722-725)
0000726 118* CPusernm equ  RAMWksta+5026 ;(726-72F) user name (10 bytes)
119* ;
0000730 120* CPfinlv equ  RAMWksta+5030 ;(730-733) floppy interleave table pointer
0000734 121* CPfdvst equ  RAMWksta+5034 ;(734-735) floppy device size (blocks)
0000736 122* CPfbps equ  RAMWksta+5036 ;(736-737) floppy bytes per sector
0000738 123* CPfspt equ  RAMWksta+5038 ;(738-738) floppy sectors per track
0000739 124* CPftps equ  RAMWksta+5039 ;(739-739) floppy tracks per side
000073A 125* CPfspd equ  RAMWksta+503A ;(73A-73A) floppy sides per disk
000073B 126* CPfofst equ  RAMWksta+503B ;(73B-73B) floppy first track offset
000073C 127* CPftyp equ  RAMWksta+503C ;(73C-73C) floppy type
128* ;
0000740 129* CPwndrod equ  RAMWksta+5040 ;(740-763) system window record (36 bytes)
0000764 130* CPscnofs equ  RAMWksta+5064 ;(764-765) bytes per display scan line
0000766 131* CPdsflg equ  RAMWksta+5066 ;(766-766) display flags
132* ;
0000771 133* CPsi1typ equ  RAMWksta+5071 ;(771-771) slot device type for slot 1
0000772 134* CPsi2typ equ  RAMWksta+5072 ;(772-772) slot device type for slot 2
0000773 135* CPsi3typ equ  RAMWksta+5073 ;(773-773) slot device type for slot 3
0000774 136* CPsi4typ equ  RAMWksta+5074 ;(774-774) slot device type for slot 4
0000775 137* CPsi5typ equ  RAMWksta+5075 ;(775-775) slot device type for slot 5
138* ;
0000880 139* CPomnram equ  RAMWksta+5180 ;(880-88F) static RAM for OMNINET
0000900 140* CPsi1ram equ  RAMWksta+5200 ;(900-9FF) static RAM for slot 1 device
0000A00 141* CPsi2ram equ  RAMWksta+5300 ;(A00-AFF) static RAM for slot 2 device
0000B00 142* CPsi3ram equ  RAMWksta+5400 ;(B00-BFF) static RAM for slot 3 device
0000C00 143* CPsi4ram equ  RAMWksta+5500 ;(C00-CFF) static RAM for slot 4 device
144* ;

```

```

00000000 146* CPiobuf equ RAMWksta+$600 ;(D00-EFF) I/O buffer (512 bytes)
00000000 147* CPstack equ RAMWksta+$800 ;(F00-FFF) initial system stack
00000000 148* CPextcrt equ RAMWksta+$800 ;(F00-F00) external CRT flag
00000001 149* CPsysst equ RAMWksta+$801 ;(F01-F01) system initialization status
000000FC 150* CPistack equ RAMWksta+$8FC ;(FFC-FFC) initial system stack pointer
151*
0000DFD0 152* CPommbi equ DSPend-430 ;OMNINET driver buffer (48 bytes)
153* ; (OMNINET can't access below $80000)
154*
155*
156* ; Corvus CONCEPT Workstation PROM address vector definitions
157*
00010004 158* CPsysrst equ ROMbase+$004 ;(10004) system restart pointer
00010008 159* CPunqid equ ROMbase+$008 ;(10008) unique workstation ID
0001000C 160* CPromvrs equ ROMbase+$00C ;(1000C) PROM version number
0001000D 161* CPromlvl equ ROMbase+$00D ;(1000D) PROM level number
0001000E 162* CFCksm equ ROMbase+$00E ;(1000E) PROM checksum
163*
00010010 164* CPbootj equ ROMbase+$010 ;(10010) jump to OMNINET disk boot subr
00010012 165* CPboot equ ROMbase+$012 ;(10012) OMNINET disk boot subr pointer
00010016 166* CPobkio equ ROMbase+$016 ;(10016) OMNINET disk blk i/o subr pointer
0001001A 167* CPodskio equ ROMbase+$01A ;(1001A) OMNINET disk i/o subr pointer
168*
00010020 169* CPlbootj equ ROMbase+$020 ;(10020) jump to local disk boot subr
00010022 170* CPlboot equ ROMbase+$022 ;(10022) local disk boot subr pointer
00010026 171* CPlbkio equ ROMbase+$026 ;(10026) local disk blk i/o subr pointer
0001002A 172* CPlskio equ ROMbase+$02A ;(1002A) local disk i/o subr pointer
173*
00010030 174* CPlbootj equ ROMbase+$030 ;(10030) jump to floppy disk boot subr
00010032 175* CPlboot equ ROMbase+$032 ;(10032) floppy boot subr pointer
00010036 176* CPlbkio equ ROMbase+$036 ;(10036) Corvus floppy blk i/o subr pointer
0001003A 177* CPlsectio equ ROMbase+$03A ;(1003A) Corvus floppy sector i/o subr pointer
0001003E 178* CPlinit equ ROMbase+$03E ;(1003E) Corvus floppy initialization
00010042 179* CPablki equ ROMbase+$042 ;(10042) Apple floppy blk i/o subr pointer
00010046 180* CPasctio equ ROMbase+$046 ;(10046) Apple floppy sector i/o subr pointer
0001004A 181* CPainit equ ROMbase+$04A ;(1004A) Apple floppy initialization
182*
00010050 183* CPkbinit equ ROMbase+$050 ;(10050) initialize (reset) keyboard driver
00010054 184* CPkgetc equ ROMbase+$054 ;(10054) get a keyboard character
185*
00010060 186* CPdsinit equ ROMbase+$060 ;(10060) initialize display driver
00010064 187* CPdsputc equ ROMbase+$064 ;(10064) display a character
00010068 188* CPdsputs equ ROMbase+$068 ;(10068) display a string
0001006C 189* CPdsconv equ ROMbase+$06C ;(1006C) convert character to upper case
190*

```

```

00010070      192* CPivec1 equ  ROMbase+5070      ,(10070) level 1 interrupt vector (SLOTS)
00010074      193* CPivec2 equ  ROMbase+5074      ,(10074) level 2 interrupt vector (DC1)
00010078      194* CPivec3 equ  ROMbase+5078      ,(10078) level 3 interrupt vector (OMNINET)
0001007C      195* CPivec4 equ  ROMbase+507C      ,(1007C) level 4 interrupt vector (DC0)
00010080      196* CPivec5 equ  ROMbase+5080      ,(10080) level 5 interrupt vector (TIMER)
00010084      197* CPivec6 equ  ROMbase+5084      ,(10084) level 6 interrupt vector (KYDB)
00010088      198* CPivec7 equ  ROMbase+5088      ,(10088) level 7 interrupt vector
199*
200*
201* ; Corvus CONCEPT Workstation I/O page definitions
202*
00030F61      203* iObotsw equ   VIAbase+5f61      ,(30F61) boot selection switches
00030F71      204* iObepiq equ   VIAbase+5f71      ,(30F71) beep frequency
205*
206*
207* ; Slot device types (set in CPslityp..CPslStyp)
208*
00000000      209* DTndev equ    0      ,no device
00000001      210* DTl0cl equ    1      ,local disk
00000002      211* DT0mni equ    2      ,OMNINET disk
00000003      212* DTc8 equ     3      ,Corvus 8" floppy disk
00000004      213* DTc5 equ     4      ,Corvus 5" floppy disk
00000005      214* DTa5 equ     5      ,Appie 5" floppy disk
215*
216*
217* ; Miscellaneous equates
218*
00000000      219* off equ     0      ,
00000001      220* on equ     1      ,
221*
00004EF9      222* jumpto equ   $4EF9      ,"jmp" op code
223*
00000032      224* DskRead equ   $32      ,disk read command
00000033      225* DskWrit equ   $33      ,disk write command
226*

```



```

228* ,
229* , Corvus CONCEPT Workstation PROM address vectors
230* ,
0000          231*      org      ROMbase
0000 00000000  232*      data.i 0          ,(10000) initial stack pointer
0004 005130B4+ 233*      data.i setup      ,(10004) start of PROM code
0008 FFFFFFFF  234*      data.l $FFFFFFF    ,(10008) unique workstation ID
000C 50        235*      data.b PROMvers   ,(1000C) PROM version number
000D 06        236*      data.b PROMlevel ,(1000D) PROM level number
000E FFFF      237*      data.w $FFFF      ,(1000E) PROM checksum
238*      ;
0010 4EF9      239*      data.w jumpto     ,(10010) jump to OMNINET disk boot subr
0014 000103FC+ 240*      data.l SBomni    ,(10014) OMNINET disk boot subr pointer
0018 00011130+ 241*      data.l ODbkIO   ,(10018) OMNINET disk blk i/o subr pointer
001A 0001119C+ 242*      data.l ODDskIO  ,(1001A) OMNINET disk i/o subr pointer
001E 0000      243*      data.w 0
244*      ;
0020 4EF9      245*      data.w jumpto     ,(10020) jump to local disk boot subr
0022 00010408+ 246*      data.l SBlocal  ,(10022) local disk boot subr pointer
0026 00010F8C+ 247*      data.l LDbkIO   ,(10026) local disk blk i/o subr pointer
002A 00011036+ 248*      data.l LDskIO   ,(1002A) local disk i/o subr pointer
002E 0000      249*      data.w 0
250*      ;
0030 4EF9      251*      data.w jumpto     ,(10030) jump to floppy disk boot subr
0032 00010414+ 252*      data.i SBflpy   ,(10032) floppy boot subr pointer
0036 0001144C+ 253*      data.l FDbkIO   ,(10036) Corvus floppy blk i/o subr pointer
003A 00011464+ 254*      data.l FDsecIO  ,(1003A) Corvus floppy sector i/o subr pointer
003E 000114DA+ 255*      data.i FDinit   ,(1003E) Corvus floppy initialization
0042 00011856+ 256*      data.l ADbkIO   ,(10042) Apple floppy blk i/o subr pointer
0046 0001187E+ 257*      data.l ADsecIO  ,(10046) Apple floppy sector i/o subr pointer
004A 00011AF4+ 258*      data.l ADinit   ,(1004A) Apple floppy initialization
004E 0000      259*      data.w 0
260*      ;
0050 00010722+ 261*      data.l KBinit   ,(10050) initialise (reset) keyboard driver
0054 000107C0+ 262*      data.l KBgetch ,(10054) get a keyboard character
0058 00000000  263*      data.l 0
005C 00000000  264*      data.l 0      ,(1005C)
265*

```

0060	0001096E+	267*	data.1 DSinit	;(10060) initialize display driver
0064	000109DE+	268*	data.1 DSputch	;(10064) display a character
0068	000109CE+	269*	data.1 DSputst	;(10068) display a string
006C	000109E4+	270*	data.1 DSctuc	;(1006C) convert character to upper case
		271*		
0070	00010630+	272*	data.1 INTslot	;(10070) level 1 interrupt vector (SLOTS)
0074	0001061E+	273*	data.1 INTdci	;(10074) level 2 interrupt vector (DC1)
0078	00010616+	274*	data.1 INTomni	;(10078) level 3 interrupt vector (GMNINET)
007C	00010604+	275*	data.1 INTdc0	;(1007C) level 4 interrupt vector (DC0)
0080	000105FA+	276*	data.1 INTlimr	;(10080) level 5 interrupt vector (TIMER)
0084	000105E8+	277*	data.1 INTkybd	;(10084) level 6 interrupt vector (KYDB)
0088	000105E4+	278*	data.1 INTlv17	;(10088) level 7 interrupt vector
008C	00000000	279*	data.1 0	;(1008C)
		180*		
		184*	list :	
		187*		

```

289* ;
290* ; Initialise Corvus CONCEPT hardware
291* ;
292* ; $30F71 - T2C1
293* ; $30F75 - shift register
294* ; $30F77 - aux control register
295* ; $30F81 - CRTC pointer
296* ; $30F83 - CRTC data
297* ;
298* ; Note that lower bytes are odd!!!
299* ;
00B4 46FC 2700 300* Setup move.w #2700,sr ;set priority to 7, nmi interrupt only
00B8 2E7C 0000 0FFC 301* move.l #CPIstack,a7 ;set system stack pointer
302* ;
303* ; delay for possible Apple floppy reset *kb 8/23/82* 0.5
304* ; 0.5
00BE 303C FFFF 305* MOVE.W #FFFF, D0 ;Must wait at least 1 second 0.5
00C2 51C8 FFFE 306* Setup1 DBRA D0, Setup1 ;* 0.5
00C6 51C8 FFFE 307* Setup2 DBRA D0, Setup2 ;* 0.5
00CA 51C8 FFFE 308* Setup3 DBRA D0, Setup3 ;* 0.5
309* ;
00CE 2C7C 0003 0F00 310* move.l #VIAbase,a6 ;get pointer to VIA I/O locations
00D4 4C78 0F00 311* clr.w CPentctrl.w ;reset system flags (CPentctrl, CPsysst)
00D8 1D7C 0017 0007 312* move.b #417,$07(a6) ;kybd control, 600 baud, 8 bit word
00DE 1D7C 000B 0005 313* move.b #50B,$05(a6) ;kybd command, no parity, no interrupts
00E4 1D7C 003E 0027 314* move.b #53E,$27(a6) ;dcom0 control, 9600 baud, 7 bit word
00EA 1D7C 00AB 0025 315* move.b #5AB,$25(a6) ;dcom0 command
00F0 1D7C 005E 0047 316* move.b #53E,$47(a6) ;dcom1 control, 9600 baud, 7 bit word
00F6 1D7C 00AB 0045 317* move.b #5AB,$45(a6) ;dcom1 command
00FC 1D7C 0000 007F 318* move.b #500,$7F(a6) ;VIA port A
0102 1D7C 0000 0061 319* move.b #500,$61(a6) ;VIA port B
0108 1D7C 0080 0067 320* move.b #580,$67(a6) ;VIA data direction A
010E 1D7C 0037 0065 321* move.b #537,$65(a6) ;VIA data direction B
0114 1D7C 0010 0077 322* move.b #510,$77(a6) ;free run shift register, counter
011A 1D7C 000F 0075 323* move.b #50F,$75(a6) ;symmetrical wave shape
0120 1D7C 00A0 0071 324* move.b #5A0,$71(a6) ;fairly low initial frequency
0124 4A2E 00C1 325* tst.b 5C1(a6) ;turn off possible OMNINET interrupt
012A 102E 0001 326* move.b 501(a6),d0 ;clear keyboard data buffer
012E 102E 0021 327* move.b 521(a6),d0 ;clear dcom0 data buffer
0132 102E 0041 328* move.b 541(a6),d0 ;clear dcom1 data buffer
329* ;

```

```

331* ;
332* ; Check Corvus CONCEPT hardware (test 1)
333* ;
334* ; Verify ports making no data accesses
335* ;
0134 0C2E 003E 0027 336*      cmpi.b  #3E,$27(a6)      ;dcom0 control
013C 6438          337*      bne.s  CHerr          ;
013E 0C2E 00AB 0025 338*      cmpi.b  #AB,$25(a6)      ;dcom0 command
0144 6430          339*      bne.s  CHerr          ;
0146 0C2E 003E 0047 340*      cmpi.b  #3E,$47(a6)      ;dcom1 control
014C 6428          341*      bne.s  CHerr          ;
014E 0C2E 00AB 0045 342*      cmpi.b  #AB,$45(a6)      ;dcom1 command
0154 6420          343*      bne.s  CHerr          ;
0156 0C2E 0017 0007 344*      cmpi.b  #17,$07(a6)      ;kybd control
015C 6418          345*      bne.s  CHerr          ;
015E 0C2E 000B 0005 346*      cmpi.b  #0B,$05(a6)      ;kybd command
0164 6410          347*      bne.s  CHerr          ;
0166 0C2E 0080 0067 348*      cmpi.b  #80,$67(a6)      ;VIA data direction A
016C 6408          349*      bne.s  CHerr          ;
016E 0C2E 0037 0065 350*      cmpi.b  #37,$65(a6)      ;VIA data direction B
0174 6712          351*      beq.s  CHend          ;
          352*      ;
0176 303C FFFE          353*      CHerr  move.w  #FFFF,d0      ;short delay before error tone
017A 51C8 FFFE          354*      CHerrl  dbra  d0,CHerrl      ;*
017E 08F8 0000 0F01 355*      bset   #0,CPSysst.w  ;set test 1 failed flag
0184 6100 0410          356*      bsr    Flash          ;*
          357*      ;
0188 303C FFFE          358*      CHend  move.w  #FFFF,d0      ;short delay before clearing screen
018C 51C8 FFFE          359*      CHendi  dbra  d0,CHendi      ;*
0190 207C 0008 0000 360*      move.l  #DSPbase,a0      ;get pointer to start of display screen
0196 227C 0008 E000 361*      move.l  #DSPend,a1      ;get pointer to end of display screen
019C 6100 03F0          362*      bsr    ZeroRam          ;clear display screen
01A0 1D7C 00FF 0075 363*      move.b  #FF,$75(a6)      ;turn off initial tone
          364*      ;
          365*      ;
          366*      ; RomTst1 -- Check Corvus CONCEPT PROM (test 2)
          367*      ;
01A6 207C 0001 000E 368*      RomTst1  move.l  #CPcksum,a0      ;get pointer to start of PROM
01AC 227C 0001 2000 369*      move.l  #ROMend,a1      ;get pointer to end of PROM
01B2 6100 0320          370*      bsr    RomTst          ;check PROM
01B6 670A          371*      beq.s  RamTst1          ;PROM ok, go on
01B8 08F8 0001 0F01 372*      bset   #1,CPSysst.w  ;set test 2 failed flag
01BE 6100 03D6          373*      bsr    Flash          ;*
          374*      ;

```

```

374* ;
377* RamTst1 -- Check Corvus CONCEPT static RAM (test 3)
378* ;
01C2 207C 0000 0700 379* RamTst1 move.l #RAMwksta,a0 ;get pointer to start of RAM
01C6 6100 031C 380* bsr WalkBit ;is RAM valid?
01CC 660C 381* bne.s RT1err ;no, report error
01CE 227C 0000 0F00 382* move.l #CPstack,a1 ;get pointer to end of RAM
; (leave room for stack)
01D4 6100 0342 384* bsr March ;is RAM valid?
01D8 670A 385* beq.s RamTst2 ;yes, go on
386* ;
01DA 08F8 0002 0F01 387* RT1err bset #2,CPsysst.w ;set test 3 failed flag
01E0 6100 03B4 388* bsr Flash ;*
389* ;
390* ;
391* RamTst2 -- Check Corvus CONCEPT dynamic RAM (test 4)
392* ;
01E4 207C 0000 E000 393* RamTst2 move.l #USRbase,a0 ;get pointer to start of RAM
01EA 6100 02FA 394* bsr WalkBit ;is RAM valid?
01EE 660A 395* bne.s RT2err ;no, report error
01F0 6100 0372 396* bsr RamSize ;get dynamic RAM size (a1 = RAM size)
01F4 6100 0322 397* bsr March ;is RAM valid?
01F8 670A 398* beq.s MemTest ;yes, go on
399* ;
01FA 08F8 0003 0F01 400* RT2err bset #3,CPsysst.w ;set test 4 failed flag
0200 6100 0394 401* bsr Flash ;*
402* ;
403* ;
404* MemTest -- Check Corvus CONCEPT dynamic RAM (test 5)
405* ;
0204 6100 033E 406* MemTest bsr IncTest ;test user dynamic RAM
0208 670A 407* beq.s MemClr ;no error, clear memory
020A 08F8 0004 0F01 408* bset #4,CPsysst.w ;set test 5 failed flag
0210 6100 03B4 409* bsr Flash ;*
410* ;

```

```

412* ;
413* MemClr -- Clear memory
414* ;
0214 207C 0000 0400 415* MemClr move.l #RAMmbug,a0 ,get pointer to start of RAM
021A 227C 0000 0EFE 416* move.l #CPStack-2,a1 ,get pointer to end of RAM
417* , (leave room for stack)
0220 6100 036C 418* bsr ZeroRam ,zero RAM
419* ;
0224 207C 0000 0000 420* move.l #DSPbase,a0 ,get pointer to start of RAM
022A 6100 0338 421* bsr RamSize ,get dynamic RAM size (a1 = RAM size)
022E 6100 035E 422* bsr ZeroRam ,zero RAM
423* ;
424* ;
425* SetMB -- Set MACSBUG RAM
426* ;
0232 41F9 0002 0004 427* SetMB lea MBinit,L,a0 ,is debug PRGM present? 0.5
0238 43F9 0002 000C 428* lea MBbase+%C.L,a1 ,* init vector should = base + %C 0.5
023E 83D0 429* cmpa.l (a0),a1 ,* 0.5
0240 4602 430* bne.s SetIntV ,no, go on
0242 4E90 431* jsr (a0) ,initialize MACSBUG
432* ;
433* ;
434* SetIntV -- Set up interrupt vectors
435* ;
0244 207C 0001 0070 436* SetIntV move.l #CPivectl,a0 ,get pointer to interrupt vector table
024A 227C 0000 0064 437* move.l #IVivll,a1 ,get pointer to interrupt vectors
0250 7006 438* moveq #6,d0 ,get number of vectors to move
0252 22D8 439* move.l (a0)+(a1)+ ,move pointers to interrupt vectors
0254 51C8 FFFC 440* dbra d0,SUI1 ,*
441* ;
0258 6100 04C8 442* bsr KBinit ,initialize keyboard
025C 6100 0710 443* bsr DSinit ,initialize display
444* ;
0260 41FA 03FB+ 445* lea msg1,a0 ,msg - "Corvus CONCEPT Initialization"
0264 6100 0760 446* bsr DSputst ,output message
0268 41FA FE26+ 447* lea msgcopy,a0 ,msg - copyright notice
026C 6100 0758 448* bsr DSputst ,output message
0270 41FA 040E+ 449* lea msg2,a0 ,msg - carriage returns
0274 6100 0750 450* bsr DSputst ,output message
451* ;

```

```

453* ,
454* ; SlotID -- Examine slots for known devices
455* ;
0278 2A7C 0000 0771 456* SlotID move.l #CPs11typ,45 ;get pointer to slot types table
027E 227C 0003 0200 457* move.l #630200,a1 ;get pointer to slot 1 interface PROM
0284 7C01 458* moveq #1,d6 ;get index for slot 1
459* ;
0284 1229 0001 460* SlotID1 move.b 01(a1),d1 ;get interface prom code (ID)
028A E189 461* lsl.l #8,d1 ;*
028C 1229 0003 462* move.b 03(a1),d1 ;*
0290 E189 463* lsl.l #8,d1 ;*
0292 1229 0005 464* move.b 05(a1),d1 ;*
0294 E189 465* lsl.l #8,d1 ;*
0298 1229 0007 466* move.b 07(a1),d1 ;*
029C 1429 0009 467* move.b 09(a1),d2 ;get interface prom code (ID)
02A0 E18A 468* lsl.l #8,d2 ;*
02A2 1429 000B 469* move.b 11(a1),d2 ;*
02A6 E18A 470* lsl.l #8,d2 ;*
02A8 1429 000D 471* move.b 13(a1),d2 ;*
02AC E18A 472* lsl.l #8,d2 ;*
02AE 1429 000F 473* move.b 15(a1),d2 ;*
02B2 4A39 0003 9FFF 474* tst.b #99FFF,L ;disable interface RAM
475* ;
02B8 B2BC A920 A900 476* cmp.l #A920A900,d1 ;is this a local disk?
02BE 6614 477* bne.s SlotID2 ;no, check next device
02C0 B4BC A903 A93C 478* cmp.l #A903A93C,d2 ;is this a local disk?
02C4 660C 479* bne.s SlotID2 ;no, check next device
02C8 6100 0DB8 480* bsr LDsync ;sync with local disk
02CC 6D34 481* blt.s SlotID8 ;bypass slot if disk did not respond
02CE 1B8C 0001 60FF 482* move.b #DTLoc1,-1(a5,d6);set device type
483* ;
02D4 B2BC A220 A000 484* SlotID2 cmp.l #A220A000,d1 ;is this an Apple floppy?
02DA 660E 485* bne.s SlotID3 ;no, check next device
02DC B4BC A203 863C 486* cmp.l #A203863C,d2 ;*
02E2 6606 487* bne.s SlotID3 ;no, check next device
02E4 1B8C 0005 60FF 488* move.b #DTa5,-1(a5,d6);set device type
489* ;
02EA B2BC 434F 5256 490* SlotID3 cmp.l #'CORV',d1 ;is this a Corvus floppy?
02F0 661A 491* bne.s SlotID9 ;no, check next device
02F2 B4BC 5553 3031 492* cmp.l #'US01',d2 ;*
02F8 6612 493* bne.s SlotID9 ;no, check next device
02FA 1B8C 0003 60FF 494* move.b #DTc8,-1(a5,d6);set device type
0300 600A 495* bra.s SlotID9 ;check next slot
496* ;
0302 08F8 0005 0F01 497* SlotID8 bset #5,CPsysst.w ;set test & failed flag
0308 6100 028C 498* bsr Flash ;*
499* ;
030C D2FC 0200 500* SlotID9 adda.w #200,a1 ;update interface PROM pointer
0310 5246 501* addq #1,d6 ;update slot number
0312 BC7C 0004 502* cmp.w #4,d6 ;have we looked at all slots?
0314 6F00 FF4E 503* ble SlotID1 ;no, check next slot
504* ;

```

```

031A 7020          504*   moveq  #iniOp,d0      ,get OMNINET Transporter number
031C 6100 0DC0    507*   bsr   ODcmd         ,*
0320 6D38          508*   blt.s SlotIDb      ,if error, go on
0322 11C7 070D    509*   move.b d7,CPtprnbr.w ,save OMNINET Transporter number
                    510*
0324 7002          511*   moveq  #EchoOp,d0   ,is OMNINET Transporter number in use?
0328 1207          512*   move.b d7,d1       ,*
032A 6100 0DBA    513*   bsr   ODcmd         ,*
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 0006 E000 520* SlotIDa move.l #USRbase,a0 ,send broadcast message to disk server
0344 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 7CFE          524*   moveq  #-1,d6      ,*
0350 6100 0E3E    525*   bsr   ODdisk10    ,*
0354 7A32          526*   moveq  #32,d5      ,*
0356 6100 0E38    527*   bsr   ODdisk10    ,*
                    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*

```



```

534* ,
535* , RptStat -- Report results of system initialization tests
536* ,
0366 7200 537* RptStat moveq #0,d1 ,initialize test number
0368 4A38 0F01 538* tst.b CPsysst.w ,any system errors?
036C 6608 539* bne.s RptSt1 ,yes, report them
036E 41FA 039E+ 540* lea msg32,a0 ,msg - All system tests passed
0372 6100 0652 541* bsr DSputst ,output message
542* , ---- bra.s RptSt8 ,output carriage returns
543* ,
0376 0338 0F01 544* RptSt1 btst d1,CPsysst.w ,did current test pass?
037A 671A 545* bofi.s RptSt12 ,yes, go on
037C 41FA 0374+ 546* lea msg30,a0 ,msg - System test
0380 6100 0644 547* bsr DSputst ,output message
0384 1001 548* move.b d1,d0 ,get test number
0386 0600 0031 549* addi.b #531,d0 ,*
038A 6100 064C 550* bsr DSputst ,output test number
038E 41FA 036F+ 551* lea msg31,a0 ,msg - failed
0392 6100 063E 552* bsr DSputst ,output message
553* ,
0396 5241 554* RptSt2 addq #1,d1 ,increment test number
0398 627C 0007 555* cmp.w #7,d1 ,finished with all tests?
039C 67D8 556* ble.s RptSt1 ,no, process next test
557* ,
039E 41FA 02E0+ 558* RptSt8 lea msg2,a0 ,output carriage returns
03A2 6100 0622 559* bsr DSputst ,*
03A6 1D7C 000F 0075 560* move.b #50F,575(a6) ,symmetrical wave shape
03AC 1D7C 00A0 0071 561* move.b #5A0,571(a6) ,output a low pitch tone
03B2 303C FFFE 562* move.w #5FFFE,d0 ,short delay
03B6 51C8 FFFE 563* RptSt9 dbra d0,RptSt9 ,*
03BA 1D7C 00FF 0075 564* move.b #5FF,575(a6) ,turn off tone
565* ,

```

```

567* ,
568* ; SelBoot -- Select boot type
569* ;
03C0 1039 0003 0F61 570* SelBoot move b IObootsw.L,d0 ;get boot selection switches
03C6 0240 00C0 571* andi.w #0C0,d0 ;*
03CA 6706 008A 572* beq SBUser ;00 - user select
03CE 0C00 0040 573* cmpi.b #040,d0 ;
03D2 6734 574* beq.s SBlocal ;,01 - local disk boot
03D4 0C00 0080 575* cmpi.b #080,d0 ;
03D8 6722 576* beq.s SBomni ;,02 - OMNINET disk boot
03DA 6038 577* bra.s SBflpy ;,03 - floppy disk boot
578* ;
03DC 579* SBdebug
03DC 41F9 0002 0004 580* lea MXBinit.L,a0 ;,*kb is debug PROM present? 0 5
03E1 43F9 0002 000C 581* lea MXBbase+4C.L,a1 ;,*kb changed test to same as in 0 5
03E8 83D0 582* cmpi.i (a0),a1 ;,*kb SetMB inst prom 0 5
03EA 466A 583* bne.s SBUser ;no. ask user for boot device
03EC 41FA 02A7+ 584* lea msg4,a0 ;msg - MACSBUG I/O on DataComm 0 0 0
03F0 6100 05D4 585* bsr DSPulst ;output message
03F4 2079 0002 0008 586* movea.l MXBentry.L,a0 ;,*kb yes, go to debugger 0 5
03FA 4ED0 587* jmp (a0) ;,*kb 0 5
588* ;
03FC 41FA 01D3+ 589* SBomni lea msg11,a0 ;msg - "OMNINET disk boot"
0400 6146 590* bsr.s SBmsg ;output message
0402 6100 0CC8 591* bsr Oboot ;load OS boot code
0406 603E 592* bra.s SBboot ;transfer control to boot code
593* ;
0408 41FA 02CF+ 594* SBlocal lea msg12,a0 ;msg - "Local disk boot"
040C 613A 595* bsr.s SBmsg ;output message
040E 6100 0AEA 596* bsr Lboot ;load OS boot code
0412 6032 597* bra.s SBboot ;transfer control to boot code
598* ;
0414 41FA 02C9+ 599* SBflpy lea msg13,a0 ;msg - "Floppy disk boot"
0418 612E 600* bsr.s SBmsg ;output message
041A 227C 0000 0771 601* movea.l 4CPSl1typ,a1 ;get pointer to slot 1 type
0420 7001 602* moveq #1,d0 ;get initial slot number
603* ;
0422 1231 00FF 604* SBflpy1 move.b -1(a1,d0),d1 ;get device type
0426 0C01 0003 605* cmpi.b #DT08,d1 ;is this a Corvus floppy disk interface?
042A 6710 606* beq.s SBflpy2 ;yes, use it for booting
042C 0C01 0005 607* cmpi.b #DTA5,d1 ;is this an Apple floppy disk interface?
0430 6710 608* beq.s SBflpy3 ;yes, use it for booting
0432 5240 609* addq #1,d0 ;update slot number
0434 B07C 0004 610* cmp.w #4,d0 ;have we looked at all slots?
0438 6FEB 611* ble.s SBflpy1 ;no, check next slot
043A 605C 612* bra.s GoToBt1 ;output error message
613* ;

```

```

043C 6100 0FD0      615* SBfloppy2 bsr   Fboot      ;load OS boot code
0440 6054           616*          bra.s   GoToBt      ;transfer control to boot code
                                617*
0444 6100 15D0      618* SBfloppy3 bsr   Aboot      ;load OS boot code
0436 604E           619* SBboot  bra.s   GoToBt      ;transfer control to boot code
                                620*
0448 6100 057C      621* SBmsg   bsr     DSputst    ;output message
044C 41FA 0298+     622*          lea     msg19,a0    ;msg - "disk boot"
0450 6100 0574      623*          bsr     DSputst    ;output message
0434 4E75           624*          rts              ;return
                                625*
0456 41FA 025A+     626* SBuser  lea     msg10,a0    ;msg - "Select boot device"
045A 6100 056A      627*          bsr     DSputst    ;output message
045E 6100 0360      628*          bsr     KBgetch    ;get reply
0462 6100 0550      629*          bsr     DScvtUC    ;convert character to upper case
0466 1F60           630*          move.b  d0,-(sp)    ;save reply
0468 6100 056E      631*          bsr     DSputch    ;echo reply
046C 700D           632*          moveq   #DSCcr,d0    ;output carriage return
046E 6100 0568      633*          bsr     DSputch    ;*
0472 101F           634*          move.b  (sp)+,d0    ;restore reply
0474 0C00 0044      635*          cmpi.b  #'D',d0      ;debug?
0478 6700 FF62      636*          beq     SBdebug     ;yes, do it
047C 0C00 0046      637*          cmpi.b  #'F',d0      ;Corvus floppy boot?
0480 6792           638*          beq.s  SBfloppy    ;yes, do it
0482 0C00 004C      639*          cmpi.b  #'L',d0      ;local disk boot?
0486 6700 FF80      640*          beq     SBlocal    ;yes, do it
048A 0C00 004F      641*          cmpi.b  #'O',d0      ;OMNINET disk boot?
048E 6700 FF6C      642*          beq     SBomni     ;yes, do it
0492 6000 FC20      643*          bra     Setup      ;no, start over again
                                644*

```

```

646* ;
647* ; GoToBt -- Transfer control to boot code
648* ;
649* ; Enter: AO.L = Boot code entry point pointer
650* ;
651* ; Values passed in registers to the boot are.
652* ;
653* ;
654* ; D0 : low user RAM address
655* ; -----
656* ; D1 : high user RAM address
657* ; -----
658* ; D2 : low user RAM address (same as D0)
659* ; -----
660* ; D3 : high user RAM address (same as D1)
661* ; -----
662* ; D4 : 0 ; 0 ; boot slot ; boot server ;
663* ; -----
664* ; D5 : 0 ;
665* ; -----
666* ; D6 : 0 ;
667* ; -----
668* ; D7 : 0 ;
669* ; -----
670* ;

0496 6C0C 671* GoToBt bge.s GoToBt2 ;go on if no boot load error
672* ;
0498 41FA 01E9* 673* GoToBt1 lea msg3,a0 ;msg - "Boot error"
049C 6100 8528 674* bsr DSputst ;output message
04A0 6000 FFB4 675* bra SBUser ;select boot device again
676* ;
04A4 4100 00BE 677* GoToBt2 bsr RamSize ;get dynamic RAM size (a1 = RAM size) 0 6
04A8 B3FC 0009 0000 678* cmpa.l #690000,a1 ;are we in PROM? 0 6
04AE 6604 679* bne.s GoToBt3 ;yes, go on 0 6
04B0 43FA 024E 680* lea CPbtslot,a1 ;set RAM size to protect code 0 6
681* ; 0 6
04B4 203C 0008 E000 682* GoToBt3 move.l #USRbase,d0 ;D0 - low user RAM address
04BA 2209 683* move.l a1,d1 ;D1 - high user RAM address
04BC 2400 684* move.l d0,d2 ;D2 - low user RAM address
04BE 2601 685* move.i d1,d3 ;D3 - high user RAM address
04C0 4284 686* clr.l d4 ;D4 - 0
04C2 1838 0760 687* move.b CPbtslot.w,d4 ;D4 - boot slot
04C4 E14C 688* lsl.w #8,d4 ;
04C8 1838 0781 689* move.b CPbtsrvr.w,d4 ;D4 - boot slot/boot server
04CC 4285 690* clr.l d5 ;D5 - 0
04CE 4286 691* clr.l d6 ;D6 - 0
04D0 4287 692* clr.l d7 ;D7 - 0
04D2 4ED0 693* jmp (a0) ;enter boot code
694* ;

```

```

696* ,
697* RomTst -- Compute checksum for PROM
698* , (PROM checksum is included in address range)
699* ,
700* , Enter: A0.L = PROM start pointer
701* , A1.L = PROM end pointer
702* ,
703* , Exit: EG = PROM checksum valid
704* , NE = PROM checksum error
705* ,
04D4 2448 706* RomTst move.l a0,a2 ;get starting address
04D6 4240 707* clr.w d0 ;
04D8 321A 708* RT1 move.w (a2)+,d1 ;
04DA B340 709* eor.w d1,d0 ;
04DC B5C9 710* cmpa.l a1,a2 ;
04DE 6DF8 711* bit.s RT1 ;
04E0 B07C FFFF 712* cmp.w #FFFF,d0 ;
04E4 4E75 713* rts ;return
714* ,
715* ,
716* WalkBit -- Walking ones and zeros
717* ,
718* , Enter: A0.L = RAM start pointer
719* ,
04E6 2448 720* WalkBit move.l a0,a2 ;get starting address
04E8 4240 721* move.l a0,a1 ;get ending address
04EA D3FC 0000 0010 722* adda.l #510,a1 ;
723* ,
04F0 303C 7FFF 724* WB1 move.w #5FFFE,d0 ;
04F4 3480 725* WB2 move.w d0,(a2) ;
04F6 B052 726* cmp.w (a2),d0 ;
04F8 661C 727* bne.s WBerr ;
04FA E358 728* roi #1,d0 ;
04FC 65F6 729* bcs.s WB2 ;
730* ,
04FE 303C 0001 731* move.w #50001,d0 ;
0502 3480 732* WB3 move.w d0,(a2) ;
0504 B052 733* cmp.w (a2),d0 ;
0506 660E 734* bne.s WBerr ;
0508 E340 735* asl #1,d0 ;
050A 64F6 736* bcc.s WB3 ;
737* ,
050C D5FC 0000 0002 738* adda.l #2,a2 ;
0512 B5C9 739* cmpa.l a1,a2 ;
0514 6DDA 740* bit.s WB1 ;
741* ,
0516 4E75 742* WBerr rts ;return
743* ,

```

```

745* ;
746* , March --
747* ,
748* ; Enter: A0.L = RAM start pointer
749* ; A1.L = RAM end pointer
750* ;
0510 2448 751* March move.l a0,a2 ,
051A 4280 752* cfr.l d0 ,
753* ,
051C 34C0 754* MR1 move.w d0,(a2)+ ,
051E B5C9 755* cmpa.l a1,a2 ,
0520 66FA 756* bne.s MR1 ,
757* ,
0522 3400 758* move.w d0,d2 ,
0524 4642 759* not.w d2 ,
0526 3222 760* MR2 move.w -(a2),d1 ,
0528 B240 761* cmp.w d0,d1 ,
052A 6616 762* bne.s MRerr ,
052C 3482 763* move.w d2,(a2) ,
052E B5C8 764* cmpa.l a0,a2 ,
0530 66F4 765* bne.s MR2 ,
766* ,
0532 3002 767* move.w d2,d6 ,
0534 4642 768* not.w d2 ,
0536 3212 769* MR3 move.w (a2),d1 ,
0538 B240 770* cmp.w d0,d1 ,
053A 6606 771* bne.s MRerr ,
053C 34C2 772* move.w d2,(a2)+ ,
053E B5C9 773* cmpa.l a1,a2 ,
0540 66F4 774* bne.s MR3 ,
775* ,
0542 4E75 776* MRerr rts ,return
777*

```

```

779* ;
780* ; IncTest --
781* ;
782* ;      Enter: A0.L = RAM start pointer
783* ;           A1.L = RAM end pointer
784* ;
0544 2448      785* IncTest move.l a0,a1      ;
0544 323C 0101 786*      move.w #5101,d1      ;
787* ;
054A 34C1      788* IT01  move.w d1,(a2)+      ;
054C E359      789*      rol.w #1,d1      ;
054E B5C9      790*      cmpa.l a1,a2      ;
0550 6DF8      791*      bit.s IT01      ;
792* ;
0552 2448      793*      move.l a0,a3      ;
0554 323C 0101 794*      move.w #5101,d1      ;
795* ;
0558 B25A      796* IT02  cmp.w (a2)+,d1      ;
055A 6406      797*      bne.s IT99      ;
055C E359      798*      rol.w #1,d1      ;
055E B5C9      799*      cmpa.l a1,a2      ;
0560 6DF6      800*      bit.s IT02      ;
801* ;
0562 4E75      802* IT99  rts      ;return
803*

```

```

805* ;
806* ; RamSize -- Get end of user RAM pointer
807* ;
808* ;      Exit:  A1.L  = RAM end pointer
809* ;
0564 227C 0009 0000 810* RamSize move.l $190000,a1 ;
056A 0C97 0001 2000 811* cmp.l $BROMend,(sp) ;are we in PROM?
0570 4E1A 812* bgt.s RamSize9 ;no, return
0572 23FC 000F FFEC 813* move.l $FFFFFC,$FFFFC.L;get actual RAM size
057A 000F FFEC ;
057C 23FC 000B FFEC 814* move.l $BFFFC,$BFFFC.L,*
0582 000B FFEC ;
0586 2279 000F FFEC 815* move.l $FFFFC.L,a1 ;,*
816* ;
058C 4E75 817* RamSize9 rts ;return
818* ;
819* ;
820* ; ZeroRam -- Move 0 to RAM subroutine
821* ;
822* ;      Enter:  A0.L  = RAM start pointer
823* ;             A1.L  = RAM end pointer
824* ;
058E 4298 825* ZeroRam clr.l (A0)+ ;
0590 B1C9 826* cmp.l a1,a0 ;
0592 6FFA 827* ble.s ZeroRam ;
0594 4E75 828* rts ;return
829* ;
830* ;
831* ; Flash -- Flash display screen subroutine
832* ;
0596 4BE7 80C2 833* Flash move.l a0-a1/a6/d0,-(sp);save registers
059A 227C 0008 DF00 834* move.l #DSPend-$30,a1 ;get pointer to end of display screen
05A0 2C7C 0003 0F00 835* move.l #VIABase,a6 ;get pointer to VIA I/O locations
05A6 1D7C 000F 0075 836* move.b $50F,$75(a6) ;symmetrical wave shape
05AC 1D7C 0040 0071 837* move.b $540,$71(a6) ;output a high pitch error tone
05B2 207C 0008 0000 838* FL1 move.l #DSPbase,a0 ;get pointer to start of display screen
05B8 4658 839* FL2 not.w (a0)+ ;
05BA B1C9 840* cmp.l a1,a0 ;
05BC 4DFA 841* bit.s FL2 ;
05BE 207C 0008 0000 842* move.l #DSPbase,a0 ;get pointer to start of display screen
05C4 4658 843* FL3 not.w (a0)+ ;
05C6 B1C9 844* cmp.l a1,a0 ;
05C8 4DFA 845* bit.s FL3 ;
05CA 303C FFFE 846* move.w $FFFE,d0 ;short delay
05CE 51C8 FFFE 847* FL4 dbra d0,FL4 ;,*
05D2 1D7C 00FF 0075 848* move.b $5FF,$75(a6) ;turn off initial tone
05D8 303C FFFE 849* move.w $FFFE,d0 ;short delay
05DC 51C8 FFFE 850* FL5 dbra d0,FL5 ;,*
05E0 4CDF 4301 851* move.l (sp)+,a0-a1/a6/d0;restore registers
05E4 4E75 852* rts ;return
853* ;

```



```

855* ,
856* ; INTlv17 -- process level 7 interrupt (ignore interrupt)
857* ,
05E6 4E73 858* INTlv17 rte ;return from interrupt
859* ,
860* ,
861* ; INTkybd -- process KEYBOARD interrupt (ignore interrupt)
862* ,
05E8 0039 0002 0003 863* INTkybd ori.b #02,030F05.L ;lvl 6 (KYBD) - turn off recv interrupt
05EE 0F05
05F0 0239 00F3 0003 864* andi.b #F3,030F05.L ;lvl 6 (KYBD) - turn off smit 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,030F7D.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 #02,030F25.L ;lvl 4 (DC0) - turn off recv interrupt
060A 0F25
060C 0239 00F3 0003 877* andi.b #F3,030F25.L ;lvl 4 (DC0) - turn off smit interrupt
0612 0F25
0614 4E73 878* rte ;return from interrupt
879* ,
880* ;
881* ; INTomni -- process OMNINET interrupt (ignore interrupt)
882* ,
0616 4A3F 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 #02,030F45.L ;lvl 2 (DC1) - turn off recv interrupt
0624 0F45
0626 0239 00F3 0003 890* andi.b #F3,030F45.L ;lvl 2 (DC1) - turn off smit interrupt
062C 0F45
062E 4E73 891* rte ;return from interrupt
892* ,

```

```

894* ,
895* ; INTslot -- process SLOT interrupt (ignore interrupt)
896* ,
0630 48E7 8080 897* INTslot move.l D0/A0,-(SP) ;save registers
0634 41F9 0003 0F7F 898*      lea    $30F7F,L,A0 ;get pointer to port A DRA
063A 1010 899*      move.b (A0),D0 ;read port A w/o handshake
063C 0840 0007 900*      bchg  #7,D0 ;toggle IGI
0640 1080 901*      move.b D0,(A0) ;write new IOX
0642 4CDF 0101 902*      move.l (SP)+,D0/A0 ;restore registers
0644 4E73 903*      rte ;return from interrupt
904* ;
905* ; SlotAdr -- compute slot address given slot number
906* ;
907* ;      Enter:  D6.B - Slot number
908* ;
909* ;      Exit:   A1.L - I/O port address
910* ;
00030100 911* SlotPtr equ    $30100 ;address of slot 0 (non-existent)
912* ;
0648 2F04 913* SlotAdr move.l d6,-(sp) ;save register
064A 4084 914*      ext.w d6 ;compute disk port address for slot
064C EB4E 915*      lsl.w #5,d6 ;*
064E 227C 0003 0100 916*      move.l #SlotPtr,a1 ;*
0654 D2C6 917*      adda.w d6,a1 ;*
0656 2C1F 918*      move.l (sp)+,d6 ;restore register
0658 4E75 919*      rts ;return
920* ;

```

```

065A 0D 0D          922* msg1  data.b  DSCCr,DSCCr
065C 43eF717675732043 923*      data.b  'Corvus CONCEPT Initialisation ('
0664 4F4E434550542049
066C 6E67746916C697A
0674 6174674F6E202028
067C 30 2E36 29      924*      data.b  PROMvers+930,' ',PROMlevl+930,')'
0680 0D 0D 00      925* msg2  data.b  DSCCr,DSCCr,0
0683 426F677420657272 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 eE2044617461436F
06AC eD6D20300D 00
06B2 53e56C65637420e2 928* msg10 data.b  'Select boot device (D,F,L,O): ',0
06BA 6F6F742064657469
06C2 63652028442C462C
06CA 4C2C4F295A2000
06D1 4F4D4E494E455400 929* msg11 data.b  'OMNINET',0
06D9 4C6F63e16C00    930* msg12 data.b  'Local',0
06DF 466C6F70707900 931* msg13 data.b  'Floppy',0
06E6 206467736B20626F 932* msg19 data.b  ' disk boot',DSCCr,0
06EE 6F740D 00
06F1 53777374654D2074 933* msg30 data.b  'System test ',0
06FA 6573742000
06FF 206661694C45640D 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*

```

```

939*      include 'CC.PROM.KB'      ,keyboard driver
940* ;
941* ; File: CC.PROM.KB
942* ; Date: 29-Oct-82
943* ; By: Keith Ball
944* ;
945* ; KEYBOARD DRIVER FOR PROM (kb)
946* ;
947* ;
948* ; EQUATES FOR ALL KEYBOARD SOFTWARE
949* ;
950* ; KEYBOARD DATA AREA DEFINITIONS
951* ;
00000000 952* KBBflgs EQU      0      ,FLAG JUST HI ORDER BYTE
00000002 953* KBBfrnt EQU     KBBflgs+2 ,FRONT PTR SAVE
00000006 954* KBBrear EQU     KBBfrnt+4 ,REAR PTR SAVE
0000000A 955* KBBsrsv EQU     KBBrear+4 ,STATUS REG SAVE AREA
0000000C 956* KBBbufc EQU     KBBsrsv+2 ,KEYBOARD BUFFER
000000F4 957* KBBlen EQU     RAMklen-KBBbufc,NUMBR OF BYTES IN BUFFER
958* ;
959* ; FLAG BIT DEFINITIONS
960* ;
00000000 961* KBFull EQU      0      ,BUFFER FULL FLAG
00000001 962* KBEmpty EQU    1      ,BUFFER EMPTY FLAG
00000002 963* KBFclos EQU   2      ,KEY CLOSURE FLAG
00000003 964* KBFshft EQU   3      ,SHIFT KEY
00000004 965* KBFontl EQU   4      ,CONTROL KEY
00000005 966* KBFlck EQU    5      ,SHIFT LOCK KEY
967* ;
968* ; MISCELLANEOUS EQUATES
969* ;
0000001F 970* KBmst40 EQU    $1F     ,MASK TO CLEAR D7-D5 (CONTROL CODE)
971* ;
972* ; TABLE VALUES FOR PROCESSING CHARACTERS
973* ;
0000007F 974* KBCqual EQU     $7F     ,QUALIFIER VALUES ) THEN THIS
000000FE 975* KBCshft EQU     $FE     ,TABLE VALUE FOR SHIFT
000000FD 976* KBCentl EQU     $FD     ,TABLE VALUE FOR CONTROL
000000FC 977* KBClock EQU     $FC     ,TABLE VALUE FOR SHIFT LOCK
000000FF 978* KBCnoch EQU     $FF     ,TABLE VALUE FOR NO CHAR CDE
979* ;
980* ; SPECIAL ASCII CHARACTERS
981* ;
00000061 982* KBClca EQU     'a'     ,LOWER CASE A
0000007A 983* KBClex EQU     'x'     ,LOWER CASE Z
0000003F 984* KBCqmr EQU     '?'     ,QUESTION MARK
985* ;

```

```

987*
988* , ADDRESSES OF KEYBOARD UART'S I/O REGISTERS
989*
00330F01 990* KBRdata EQU $30F01 ,DATA INPUT PORT
00330F03 991* KBRstat EQU $30F03 ,STATUS REGISTER
00330F05 992* KBRcmd EQU $30F05 ,COMMAND REGISTER
00330F07 993* KBRcntl EQU $30F07 ,CONTROL REGISTER
994*
995* , COMMAND AND CONTROL REGISTER VALUES
996*
00000002 997* KBccOff EQU $02 ,TURN OFF UART (CMD)
00000017 998* KBcc600 EQU $17 ,600 BAUD AND 8 BIT XMIT (CTL)
00000008 999* KBccBrk EQU $08 ,XMIT A BREAK (CMD)
00000009 1000* KBccGo EQU $09 ,TURN ON INTS & UART (CMD)
1001*
00000700 1002* KBdsInt EQU $0700 ,DISABLE 68000 INTERRUPTS
1003*
1004*
1005* , KBinit - initialize (reset) keyboard
1006*
1007* , REGISTER A2 IS USED AS POINTER TO COMMAND REGISTER
1008* , REGISTER A3 IS ADDRESS OF KBRD DATA AREA
1009*
0721 46E7 80F6 1010* KBinit MOVEM.L D0/A0-A3,-(SP) ,save registers
0726 47F8 0300 1011* LEA RAMkbuf.W,A3
072A 45F9 0063 0F03 1012* LEA KBRcmd.L,A2
0730 148C 0002 1013* MOVE.B #KBccOff,(A2) ,TURN OFF KBRD
0734 41EB 0000 1014* LEA KBBfigs(A3),A0 ,CLEAR INT HANDLER FLAGS
0738 4290 1015* CLR.L (A0) ,INCLUDES QUALIFIERS
073A 08D0 0001 1016* BSET #KBFeaty,(A0) ,BUFFER IS EMPTY
1017*
1018* , INITIALIZE FRONT & REAR POINTERS
1019*
073E 41EB 000C 1020* LEA KBBbuf(A3),A0
0741 43EB 0002 1021* LEA KBBfrnt(A3),A1
0746 22C8 1022* MOVE.L A0,(A1)+
0748 2288 1023* MOVE.L A0,(A1)
074A 41FA 0034+ 1024* LEA KBintr,A0 ,SETUP AUTOVECTOR 6
074E 21C8 0078 1025* MOVE.L A0,IV1v16.W ,WITH ADDR OF INT HANDLER
1026*
1027* , TURN ON KEYBOARD UART
1028*
0751 1039 0003 0F03 1029* MOVE.B KBRstat.L,D0 ,RESET UART
0758 1039 0003 0F01 1030* MOVE.B KBRdata.L,D0 ,CLEAR RECEIVE
075E 13FC 0017 0003 1031* MOVE.B #KBcc600,KBRcntl.L ,8 BITS, 600 BAUD EMISSION
0764 0F07
0766 148C 0008 1032* MOVE.B #KBccBrk,(A2) ,FORCE BREAK OF KBRD
076A 303C 8235 1033* MOVE.W #33333,D0 ,DELAY FOR UART TO DO BREAK
076E 51C8 FFFE 1034* KBinit1 DBF D0,KBinit1 ,NEED MINIMUM OF 33.3 MILLISECS
0772 148C 0009 1035* MOVE.B #KBccGo,(A2) ,TURN ON UART & INTERRUPTS
0776 46FC 2500 1036* move.w #42500,sr ,set priority to 6, KYBD intr only
077A 4CDF 0F01 1037* MOVEM.L (SP)+,D0/A0-A3 ,restore registers
077E 4E75 1038* RTS
1039*

```

```

1041* ;
1042* ; KBintr - Keyboard interrupt service routine
1043* ;
1044* ; BEGIN INTERRUPT SERVICE ROUTINE. THIS IS THE ENTRY POINT. IT'S ADDRESS
1045* ; MUST BE PLACED IN AUTO VECTOR INTERRUPT & VECTOR BEFORE KEYBOARD INTERRUPT
1046* ; IS TURNED ON.
1047* ;
1048* ; REGISTER USAGE:  D0 - KEYCODE
1049* ;                   D1 - CHARACTER
1050* ;                   A0 - ADDRESS OF FLAG BYTE
1051* ;                   A2 - BASE ADDRESS OF KBRD DATA AREA
1052* ;
0780 48E7 FFFE 1053* KBintr MOVEM.L D0-A6,-(SP) ;SAVE REGISTERS ON STACK
0784 45F8 0300 1054* LEA  RAMkbbuf.W,A2 ;BASE ADDR OF KBRD DATA AREA
0788 417E 1055* BSR.S  KBgetky ;GET KEYCODE FROM VART DATA PGRT
1056* ;
1057* ; IF BIT 7 OF KEYCODE SET THEN CLOSURE ELSE RELEASE
1058* ;
078A 41EA 0000 1059* LEA  KBBflgs(A2),A0
078E 0890 0002 1060* BCLR  #KBFclos,(A0) ;ASSUME RELEASE
0792 0800 0007 1061* BTST  #7,D0 ;KEYCODE BIT D7 CLEAR?
0796 4708 1062* BEQ.S  KBintri ;YES
0798 08D0 0002 1063* BSET  #KBFclos,(A0)
079C 0880 0007 1064* BCLR  #7,D0
1065* ;
1066* ; GET CHARACTER CODE FOR THIS KEYCODE
1067* ;
07A0 43FA 010C+ 1068* KBintri LEA  KBstable,A1 ;ASSUME SHIFT TABLE
07A4 0810 0003 1069* BTST  #KBFshft,(A0)
07A8 4604 1070* BNE.S  KBintr2 ;SHIFT BIT SET
07AA 43FA 0162+ 1071* LEA  KBrtable,A1 ;ELSE USE REGULAR TABLE
07AE 1231 0000 1072* KBintr2 MOVE.B 0(A1,D0.W),D1 ;INDEX TABLE BY KEYCODE
1073* ;
1074* ; IF CHAR(D1) = $FF THEN IGNORE AND EXIT
1075* ;
07B2 0C01 00FF 1076* CMPI.B #KBCnoch,D1
07B4 4702 1077* BEQ.S  KBintr9
07B8 415E 1078* BSR.S  KBproky ;ELSE PROCESS KEYCODE
1079* ;
1080* ; EXIT INTERRUPT SERVICE ROUTINE
1081* ;
07BA 4CDF 7FEF 1082* KBintr9 MOVEM.L (SP)+,D0-A6 ;RESTORE REGISTERS
07BE 4E73 1083* RTE ;EXIT INTERRUPT
1084* ;

```

```

1086* ,
1087* ; KBgetch - Get a keyboard character
1088* ;
1089* ; Register usage: A0 = Front pointer
1090* ; A1 = address of end of buffer + 1
1091* ; A2 = updated front pointer
1092* ; A3 = address of front pointer
1093* ; A4 = address of flag byte
1094* ; A5 = address of keyboard data area
1095* ; A6 = address of Status Register save area
1096* ;
1097* ; Exit. D0 B - Next character in buffer
1098* ;
07C0 48E7 00FE 1099* KBgetch MOVEM.L A0-A6,-(SP) ;save all address registers
07C4 48F6 0300 1100* LEA RAMkbbuf W,A5 ;keyboard data area
07C8 49ED 0000 1101* LEA KBBflgs(A5),A4 ;address of flag byte
1102* ;
1103* ; Wait for a character in the Buffer.
1104* ;
07CC 0814 0001 1105* KBgchr1 BTST #KBFempty,(A4) ;while (Buffer_empty) do,
07D0 00FA 1106* BNE.S KBgchr1 ;*
1107* ;
1108* ; have char, check for wrap around before get char
1109* ;
07D2 47ED 0002 1110* LEA KBBfrnt(A5),A3 ;pointer to Front save loc
07D6 4053 1111* MOVE.L (A3),A0 ;Front pointer
07D8 43ED 0100 1112* LEA KBBbufr+KBBlen(A5),A1 ;end of buffer + 1
07DC 2448 1113* MOVE.L A0,A2 ;
07DE 528A 1114* ADDQ.L #1,A2 ;add one to pointer to get next addr
07E0 85C8 1115* CMPA.L A0,A2 ;Front=end of buffer + 1 ?
07E2 6604 1116* BNE.S KBgchr2 ;No
07E4 45ED 000C 1117* LEA KBBbufr(A5),A2 ;yes, then pointer wraps back to beginning
1118* ;
07E8 40ED 000A 1119* KBgchr2 LEA KBBsrsv(A5),A6 ;
07EC 40D6 1120* MOVE.W SR,(A6) ;
07EE 007C 0700 1121* ORI.W #KBDsInt,SR ;*** disable interrupts
07F2 1010 1122* MOVE.B (A0),D0 ;get char
07F4 248A 1123* MOVE.L A2,(A5) ;save new Front value
07F6 85ED 0006 1124* CMPA.L KBBrear(A5),A2 ;if Front=Rear then
07FA 6604 1125* BNE.S KBgchr3 ;Buffer_empty := true;
07FC 08D4 0001 1126* BSET #KBFempty,(A4) ;
1127* ;
0800 46D6 1128* KBgchr3 MOVE.W (A6),SR ;*** enable interrupts
0802 4CDF 7700 1129* MOVEM.L (SP)+,A0-A6 ;restore callers address regs
0806 4E75 1130* RTS
1131*

```

```

1133* ;
1134* ; KBgetty - GET KEYCODE (IGNORES ERRORS)
1135* ;
1136* ; EXIT . (DO) - UART DATA PORT BYTE
1137* ;
0800 4280 1138* KBgetty CLR.L DO ;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*

```



```

1149* ;
1150* ; KBproky - PROCESS CHARACTER OR QUALIFIER
1151* ;
1152* ; Enter: D1 = CHARACTER CODE FROM TABLE
1153* ; D0 = KEYCODE
1154* ; A0 = ADDRESS OF FLAGS
1155* ;
0818 0C01 007F 1156* KBproky CMPI.B #KBCqual,D1 ;IS IT A QUALIFIER
081C 623C 1157* BHI.S KBpro3 ;YES
1158* ;
1159* ; IGNORE REST OF KEYS IF NOT CLOSURE
1160* ;
081E 081C 0002 1161* BTST #KBFclos,(A0)
0821 6738 1162* BEQ.S KBpro9
1163* ;
1164* ; TEST FOR CONTROL
1165* ;
0824 0810 0004 1166* BTST #KBFent1,(A0)
0828 670C 1167* BEQ.S KBpro1 ;NO,TRY SHIFT LOCK
082A 0C01 003F 1168* CMPI.B #KBCqmrk,D1
082E 6306 1169* BLS.S KBpro1
0830 0201 001F 1170* ANDI.B #KBmsk40,D1 ;CLEAR BITS D7,D6,D5 OF CHAR
0834 601A 1171* BRA.S KBpro2 ;PUT CHAR
1172* ;
1173* ; TEST FOR SHIFT LOCK
1174* ;
0836 0810 0005 1175* KBpro1 BTST #KBFlock,(A0)
083A 6714 1176* BEQ.S KBpro2 ;KEY NOT DOWN
083C 0C01 0061 1177* CMPI.B #KBC1ca,D1
0840 650E 1178* BCS.S KBpro2 ;NOT WITHIN RANGE
0842 0C01 007A 1179* CMPI.B #KBC1cr,D1
0846 6208 1180* BHI.S KBpro2 ;NOT WITHIN RANGE
0848 43FA 0064+ 1181* LEA KBstable,A1
084C 1231 0000 1182* MOVE.B 0(A1,D0.W),D1 ;INDEX TABLE BY KEYCODE
1183* ;
1184* ; IF BUFFER NOT FULL PUT CHARACTER
1185* ;
0850 0810 0000 1186* KBpro2 BTST #KBFfull,(A0)
0854 6606 1187* BNE.S KBpro9
0856 6106 1188* BSR.S KBputch
0858 6002 1189* BRA.S KBpro9
1190* ;
1191* ; PROCESS A QUALIFIER KEY
1192* ;
085A 6128 1193* KBpro3 BSR.S KBqual
085C 4E75 1194* KBpro9 RTS
1195* ;

```

```

1197* ;
1198* ; KBputch - PUT ONE CHARACTER IN BUFFER
1199* ;
1200* ; Enter: D1 = BYTE TO PUT IN BUFFER
1201* ; A0 = ADDRESS OF FLAGS
1202* ; A2 = ADDRESS OF KEYBOARD DATA AREA
1203* ;
1204* ;
1205* ; PUT CHARACTER IN CIRCULAR QUEUE AT REAR
1206* ;
085E 4BEA 0006 1207* KBputch LEA KBBrear(A2),A5
0861 2655 1208* MOVE.L (A5),A3
0864 16C1 1209* MOVE.B D1,(A3)+ ;UPDATE POINTER ALSO
1210* ;
1211* ; IF REAR > ENDBUFFER THEN REAR := @BUFFER
1212* ;
0864 49EA 0100 1213* LEA KBBbufr+KBBlen(A2),A4
086A B7CC 1214* CMPA.L A4,A3
086C 6604 1215* BNE.S KBput1 ;NOT BEYOND BUFFER
086E 47EA 000C 1216* LEA KBBbufr(A2),A3
1217* ;
1218* ; IF FRONT = REAR THEN BUFFER FULL
1219* ;
0872 B7EA 0002 1220* XHput1 CMPA.L KBBfrnt(A2),A3
0876 6604 1221* BNE.S KBput2
0878 08D0 0000 1222* BSET #KBFfull,(A0)
087C 2A8B 1223* KBput2 MOVE.L A3,(A5) ;UPDATE REAR IN MEMORY
087E 0890 0001 1224* BCLR #KBFempty,(A0) ;SHOW BUFFER NOT EMPTY
0882 4E75 1225* RTS
1226* ;

```

```

1228* ;
1229* ; KBqual - PROCESS QUALIFIER KEYS
1230* ;
1231* ; Enter: D1 = CHARACTER CODE FROM TABLE
1232* ; A0 = ADDRESS OF FLAGS
1233* ;
0884 0C01 00FE 1234* KBqual CMPI.B #KBCshft,D1 ;IS IT SHIFT?
0888 6604 1235* BNE.S KBqual1 ;NO
088A 7403 1236* MOVEQ #KBFshft,D2 ;BIT POSITION OF SHIFT
088C 6012 1237* BRA.S KBqual3 ;CHANGE FLAG
1238* ;
088E 0C01 00FD 1239* KBqual1 CMPI.B #KBCent1,D1 ;IS IT CONTROL?
0892 6604 1240* BNE.S KBqual2 ;NO
0894 7404 1241* MOVEQ #KBFent1,D2 ;BIT POSITION OF CONTROL
0896 6008 1242* BRA.S KBqual3 ;CHANGE FLAG
1243* ;
0898 0C01 00FC 1244* KBqual2 CMPI.B #KBClock,D1 ;IS IT SHIFT LOCK?
089C 660E 1245* BNE.S KBqual9 ;NO,THEN IT'S GARBAGE
089E 7405 1246* MOVEQ #KBFlock,D2
1247* ;
1248* ; IF CLOSURE THEN SET FLAG ELSE CLEAR FLAG
1249* ;
08A0 0810 0002 1250* KBqual3 BTST #KBFclos,(A0)
08A4 6704 1251* BEQ.S KBqual8
08A6 05D0 1252* BSET D2,(A0)
08A8 6002 1253* BRA.S KBqual9
08AA 0590 1254* KBqual8 BCLR D2,(A0)
08AC 4E75 1255* KBqual9 RTS
1256* ;

```

```

1258* ;
1259* ; THE SHIFT TABLE
1260* ; TABLE IS INDEXED BY KEYCODE. EACH BYTE REPRESENTS THE ENTRY FOR
1261* ; THE CORRESPONDING KEYCODE.
1262* ;
1263* ; 0 1 2 3 4 5 6 7 8 9 A B C D E F
1264* ; KBStable
08AE 1265* ; .. 3 9 .. 6 , - cr .. i 7 .. 4 8 5 2
08AE FF 38 39 FF 36 2C 1266* 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
1267* ; + .. ( ; cr ) bs .. ) ? P _ : " " ..
1268* ; DATA.B $2B,$FF,$7B,$7C,$0D,$7D,$08,$FF,$29,$3F,$50,$5F,$3A,$7E,$22,$FE
1269* ; + .. ( bs cr ) ; .. ) ? P _ : " " .. ,0 6
08BE 2B FF 7B 08 0D 7D 1270* 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
1271* ; .. .. .. .. .. .. .. .. .. .. $ % R T F C V B
08CE FF FF FF FF FF FF 1272* 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
1273* ; @ # W E S D X C esc ! .. Q .. A .. Z
08DE 40 23 57 45 53 44 1274* 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
1275* ; ^ & Y U H J N M .. .. .. sp .. 0 ..
08EE 5E 26 59 55 48 4A 1276* 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
1277* ; * ( ; O K L ( ) .. .. .. .. ..
08FE 2A 28 49 4F 4B 4C 1278* 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
1279*

```

```

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
1287* ; KBrtable
090E 1288* ; ... 3 9 ... 6 , - cr ... i 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,$9E
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,$9E ;0.6
0924 5C FF 30 1F 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 e s i ... q ... a ... x
093E 31 33 77 65 73 64 1297* DATA.B $32,$33,$77,$65,$73,$64,$78,$63,$1B,$31,$FF,$71,$74,$6C,$61,$9E,$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,$5E
0954 6E 6D FD FF FF 20
095A FF 30 FF 2E
1300* ; , 8 9 i o k l , ... ..
095E 38 39 49 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* ;

```

```

1304*      include 'CC.PROM.DS'      ,display driver
1305* ;
1306* ; File: CC.PROM.DS
1307* ; Date: 29-Oct-82
1308* ;
1309* ; DISPLAY DRIVER FOR PROM (mb) 05/18/82
1310* ;
1311* ; BOTH horizontal and vertical display driver
1312* ; contains default window records, copies them into memory
1313* ; contains default character sets
1314* ; no CRTST code: no window functions
1315*
00000060 1316* DSdefOf equ 96      ,default bytes per scan line
0008D55E 1317* DShomeH equ $8D55E ,horizontal home location
0008D504 1318* DShomeV equ $8D504 ,vertical home location
00000006 1319* DScellW equ 6        ,character cell width
0000000A 1320* DScellY equ 10     ,character cell height
000002CF 1321* DSmaxXH equ 719   ,120*DScellW-1
0000022F 1322* DSmaxYH equ 559   ,56*DScellY-1
0000022D 1323* DSmaxXV equ 557   ,93*DScellW-1
000002CF 1324* DSmaxYV equ 719 ,72*DScellY-1
1325* ;
0000000D 1326* DSCr equ 90D      ,carriage return character
0000001B 1327* DSCesc equ 91B    ,escape character
00000020 1328* DSCbInk equ 920   ,blank character
00000061 1329* DSClca equ 961    ,lower case "a"
0000007A 1330* DSClcr equ 97A   ,lower case "r"
00000020 1331* DSCdiff equ 920
1332* ;
1333* ; Character Set Record Equates
1334* ;
1335* ;,Stblloc equ 0      ,character set data pointer (not used)
00000004 1336* CSlpch equ 4      ,scanlines per character
00000006 1337* CSbpch equ 6      ,bits per character
00000008 1338* CSfstch equ 8     ,first character code - ascii
0000000A 1339* CSlastch equ 10  ,last character code - ascii
0000000C 1340* CSmask equ 12     ,mask used in positioning cells
00000010 1341* CSattr1 equ 16   ,attributes
1342* ; bit 0 = 1 - vertical orientation
00000011 1343* CSattr2 equ 17     ,currently unused
00000012 1344* CSdata equ 18  ,offset of char data from char record
1345* ;

```

```

1347* ,
1348* , Window Record Equates
1349* ,
00000000 1350* WRcharpt equ 0 ,character set pointer
00000004 1351* WRhomept equ 4 ,home (upper left) pointer
00000008 1352* WRcuradr equ 8 ,current location pointer
0000000C 1353* WRhomeof equ 12 ,bit offset of home location
0000000E 1354* WRbasex equ 14 ,home x value, relative to root window
00000010 1355* WRbasey equ 16 ,home y value, relative to root window
00000012 1356* WRlngths equ 18 ,maximum x value, relative to window (bits)
00000014 1357* WRlngthy equ 20 ,maximum y value, relative to window (bits)
00000016 1358* WRcursx equ 22 ,current x value (bits)
00000018 1359* WRcursy equ 24 ,current y value (bits)
0000001A 1360* WRbitofs equ 26 ,bit offset of current address
0000001C 1361* WRgrorgx equ 28 ,graphics - origin x (bits relative to home loc)
0000001E 1362* WRgrorgy equ 30 ,graphics - origin y (bits relative to home loc)
00000020 1363* WRattr1 equ 32 ,attributes
00000021 1364* WRattr2 equ 33 ,attributes
1365* ,
00000000 1366* vert equ 0 ; 1 = vertical, 0 = horizontal screen
00000001 1367* graphic equ 1 ; 1 = graphics, 0 = character mode
00000002 1368* curson equ 2 ; 1 = cursor on, 0 = cursor off
00000003 1369* invcurs equ 3 ; 1 = inverse, 0 = underline cursor
00000004 1370* wrapon equ 4 ; 1 = wrap, 0 = clip at eoln
00000005 1371* noscroll equ 5 ; 1 = no scroll, 0 = scroll
1372* ,
00000011 1373* WRstate equ 34 ,used for decoding escape sequences
00000023 1374* WRrdlen equ 35 ,window description record length
1375* ,
00000024 1376* WRlength equ 36 ,actual window record length
1377* ,

```

```

1379* ;
1380* ; DSinit - Initialize display driver
1381* ;
094E 48E7 08E0 1382* DSinit MOVEM.L D4/A0-A2,-(SP) ;save registers
0972 31FC 0960 0764 1383* MOVE.W #DSdefOf,CPscnofs.W ;set bytes per scan line
0978 41FA 0380+ 1384* LEA DSwndH,A0 ;assume horizontal orientation
097C 45FA 03D0+ 1385* LEA DSsetH,A2 ;*
0980 0839 0003 0003 1386* BTST #3,IObootsw.L ;is display horizontal?
0986 0F61
0988 6708 1387* BOFF.S DSinit1 ;yes, go on
098A 41FA 03D4+ 1388* LEA DSwndV,A0 ;set vertical orientation
098E 45FA 03F4+ 1389* LEA DSsetV,A2 ;*
0992 227C 0000 0740 1390* DSinit1 MOVE.L #CPwndrcd,A1 ;get pointer to RAM window record
0998 7823 1391* MOVEQ #WRIlength-1,D4 ;get window record length 0 6
099A 12D8 1392* DSinit2 MOVE.B (A0)+,(A1)+ ;copy window record to RAM 0 6
099C 51CC FFFC 1393* DBRA D4,DSinit2 ;*
09A0 207C 0000 0740 1394* MOVE.L #CPwndrcd,A0 ;get RAM window record pointer
09A6 214A 0000 1395* MOVE.L A2,WCharpt(A0) ;set character set record pointer
09AA 6100 01EE 1396* BSR DScur ;display cursor on screen
09AE 4CDF 0710 1397* MOVEM.L (SP)+,D4/A0-A2 ;restore registers
09B2 4E75 1398* RTS ;return
1399* ;
1400* ;
1401* ; DSscvUC - Convert character to upper case
1402* ;
1403* ; Enter: D0.B = ASCII character
1404* ;
1405* ; Exit: D0.B = upper case ASCII character
1406* ;
09B4 0C00 0061 1407* DSscvUC CMPI.B #DSClca,D0 ;is character lower case?
09B8 650A 1408* BLO.S DSscvU1 ;no, return
09BA 0C00 007A 1409* CMPI.B #DSClcs,D0 ;*
09BE 6204 1410* BHI.S DSscvU1 ;no, return
09C0 0400 0020 1411* SUB1.B #DSCdiff,D0 ;convert character to upper case
09C4 4E75 1412* DSscvU1 RTS ;return
1413* ;
1414* ;
1415* ; DSputst - Display a string
1416* ;
1417* ; Enter: A0.L - Character string pointer
1418* ; (terminated by 0)
1419* ;
09C6 48E7 8080 1420* DSputst movem.l D0/A0,-(SP) ;save registers
1421* ;
09CA 1018 1422* DSpst1 move.b (a0)+,d0 ;get next character
09CC 6704 1423* beq.s DSpsit9 ;finished, return
09CE 6108 1424* bsr.s DSputch ;output character
09D0 60F8 1425* bra.s LSp,-*1 ;get next character
1426* ;
09D2 4CDF 0101 1427* DSpsit9 movem.l (SP)+,D0/A0 ;restore registers
09D6 4E75 1428* rts ;return
1429* ;

```



```

1431* ;
1432* ; DSputch - Display a character
1433* ;
1434* ; Enter: D0.B - Character to output
1435* ;
09D8 48E7 FFFE 1436* DSputch MOVEM.L D0-D7/A0-A6,-(SP) ;save registers
09DC 0240 007F 1437* andi.w #57F,D0 ;make character 7 bits
09E0 207C 0000 0749 1438* MOVE.L #CPwndrcd,A0 ;get RAM window record pointer
09E6 2468 0000 1439* MOVE.L WRcharpt(A0),A2 ;get character set record pointer
09EA 4283 0000 1440* CLR.L D3 ;
09EC 1628 0022 1441* MOVE.B WRstate(A0),D3 ;
09F0 E34B 0000 1442* LSL.W #1,D3 ;convert to state table index
09F2 43FA 0332+ 1443* LEA DSctbl,A1 ;
09F6 3631 3000 1444* MOVE.W 0(A1,D3.W),D3 ;D3 = dist from DSctbl
09FA 4EF1 3000 1445* JMP 0(A1,D3.W) ;go to current state processing
;
09FE 5228 0012 1447* DSnext ADDQ.B #1,WRstate(A0) ;increment for next state
0A02 6004 0000 1448* BRA.S DSexit ;return
;
0A04 4228 0012 1450* DSreset CLR.B WRstate(A0) ;reset current state
1451* ;
0A08 4CDF 7FFF 1452* DSexit MOVEM.L (SP)+,D0-D7/A0-A6 ;restore registers
0A0C 4E75 0000 1453* RTS ;return
1454* ;
0AGE 803C 001B 1455* DSct0 CMP.B #DScsc,D0 ;is char ESC?
0A12 67EA 0000 1456* BEQ.S DSnextSt ;yes, go to next state
0A14 806A 0008 1457* CMP.W CSfrstch(A2),D0 ;ascinum < first char?
0A18 6508 0000 1458* BLO.S DSct1 ;yes, it's a control char
0A1A 6148 0000 1459* BSR.S DSshwCh ;display character
0A1C 6100 6102 1460* BSR DSincx ;inccurx
0A20 60E6 0000 1461* BRA.S DSexit ;return
1462* ;
0A22 5140 0000 1463* DSct1 SUBQ.W #8,D0 ;commence decoding ctrl char
0A24 6BE2 0000 1464* BMI.S DSexit ;
0A26 0C40 0005 1465* CMPI.W #5,D0 ;ascinum in [8..13]?
0A2A 62DC 0000 1466* BMI.S DSexit ;yes, do cursor ctrl
0A2C 47FA 02DA+ 1467* LEA DSctbl,A3 ;A3=>jump table for ctrl chars
0A30 E348 0000 1468* LSL.W #1,D0 ;make it word count
0A32 487A FFD4+ 1469* PEA DSexit ;ensure RTS to exit
0A36 5033 0000 1470* MOVE.W 0(A3,D0),D0 ;D0 is offset from DSctbl
0A3A 4EF3 0000 1471* JMP 0(A3,D0) ;jump to proper routine
1472* ;

```

0A3E	4241	1474*	DSesc	CLR.W	D1	;initialise index reg
0A40	47FA	1475*		LEA	DSeTbl1,A3	;A3=> beginning of table
		1476*				;
0A44	8073	1477*	DSesc1	CMP.W	0(A3,D1),D0	;does table entry match char?
0A48	670A	1478*		BEQ.S	DSesc2	;yes, go on
0A4A	5841	1479*		ADDQ.W	#4,D1	;go to next entry
0A4C	4A73	1480*		TST.W	0(A3,D1)	;end of table?
0A50	6AF2	1481*		BPL.S	DSesc1	;no, loop
0A52	60B0	1482*		BRA.S	DSreset	;return
		1483*				;
0A54	3001	1484*	DSesc2	MOVE.W	D1,D0	;set D0 to table offset
0A56	5440	1485*		ADDQ.W	#2,D0	;
0A58	487A	1486*		PEA	DSreset	;ensure RTS to reset state
0A5C	3033	1487*		MOVE.W	0(A3,D0),D0	;D0 is offset from DSeTbl1
0A60	4EF3	1488*		JMP	0(A3,D0)	;jump to proper routine
		1489*				;

```

1491* ;
1492* ; DSshwCh - Display character
1493* ;
1494* ; Enter. A0.L = window record pointer
1495* ; A2.L = character set record pointer
1496* ; D0.W = ASCII character
1497* ;
1498* ; Note. Character set must be in bytes, not words
1499* ;
0A64 6100 FF4E 1500* DSshwCh BSR DSscvtUC ;convert character to upper case
0A68 806A 0008 1501* CMP.W CSfrstch(A2),D0 ;is character in character set?
0A6C 6D06 1502* BLT.S DSshow1 ;no, output space
0A6E 806A 000A 1503* CMP.W CSlastch(A2),D0 ;*
0A72 6F04 1504* BLE.S DSshow2 ;yes, output character
0A74 303C 0020 1505* DSshow1 MOVE.W #DSCbink,D0 ;no, output space
1506* ;
0A78 906A 0008 1507* DSshow2 SUB.W CSfrstch(A2),D0 ;get relative character position
0A7C 47FA 0318+ 1508* LEA DSscset+V+CSdata,a3 ;get pointer to character data
0A80 C0FC 0006 1509* MULLU #DScellW,D0 ;*
0A84 D7C0 1510* ADDA.L D0,A3 ;*
0A86 2868 0008 1511* MOVE.L WRcuradr(A0),A4 ;get current character address
0A8A 322A 0004 1512* MOVE.W CS1pch(A2),D1 ;get number of scan lines for character
0A8E 5341 1513* SUBQ.W #1,D1 ;get count for DBRA
0A90 3438 0764 1514* MOVE.W CPscnofs.W,D3 ;get scan line length
0A94 5A28 001A 1515* MOVE.W WRbitofs(A0),D5 ;get bit offset of character in cell
0A98 2C2A 000C 1516* MOVE.L CSmask(A2),D6 ;get character mask
0A9C 0828 0000 0021 1517* BTST #vert,WRattr2(A0) ;is this vertical orientation?
0AA2 671C 1518* BOFF.S DSshow6 ;no, output horizontal character
1519* ;
1520* ; output vertical orientation character
1521* ;
0AA4 2006 1522* MOVE.L D6,D0 ;
0AA6 4680 1523* NOT.L D0 ;D0 = inverted mask
0AA8 EABE 1524* ROR.L D5,D6 ;D6 = positioned mask
0AAA 141B 1525* DSshow3 MOVE.B (A3)+,D2 ;D2 = char data
0AAC E14A 1526* LSL.W #8,D2 ;
0AAE 4842 1527* SWAP D2 ;get char in high word
0AB0 C480 1528* AND.L D0,D2 ;clear rest of source char
0AB2 EAAA 1529* LSR.L D5,D2 ;position source char
0AB4 CD94 1530* AND.L D6,(A4) ;clear dest char area
0AB6 8594 1531* OR.L D2,(A4) ;move in character
0AB8 98C3 1532* SUBA.W D3,A4 ;
0ABA 51C9 FFEE 1533* DBRA D1,DSshow3 ;repeat for D1:=CS1pch-1 to 0
0ABE 605A 1534* BRA.S DSshow9 ;return
1535* ;

```

```

1537*      ;
1538*      ; output horizontal orientation character
1539*      ;
OAC0 4A44      1540* DSshow6 TST.W  -(A4)          ;A4==long word with cell
OAC1 EBBE      1541* ROL.L  D5,D6          ;D6 = positioned mask
OAC4 2806      1542* MOVE.L  D6,D4          ;
OAC6 4684      1543* NOT.L  D4           ;D4 = inverted mask
OAC8 7007      1544* moveq  #7,d0         ;use 8 bits of character data
1545*      ;
OACA 4282      1546* DSshow7 clr.l  d2         ;clear current scan line of character
OACC 4A40      1547* tst.w  d0           ;have we used 8 bits of character data?
OACE 6D3C      1548* bit.s  DSshw76        ;yes, pad with space
OADD 012B 0000 1549* btst  d0,0(a3)         ;construct next horizontal character
OAD4 6704      1550* boff.s DSshw71        ;" from vertical character data
OAD6 08C2 0000 1551* bset  #0,d2          ;"
OADA 012B 0001 1552* DSshw71 btst  d0,1(a3)    ;"
OADE 6704      1553* boff.s DSshw72        ;"
OAE0 08C2 0001 1554* bset  #1,d2          ;"
OAE4 012B 0002 1555* DSshw72 btst  d0,2(a3)    ;"
OAE8 6704      1556* boff.s DSshw73        ;"
OAEA 08C2 0002 1557* bset  #2,d2          ;"
OAE E 012B 0003 1558* DSshw73 btst  d0,3(a3)    ;"
OAF2 6704      1559* boff.s DSshw74        ;"
OAF4 08C2 0003 1560* bset  #3,d2          ;"
OAF8 012B 0004 1561* DSshw74 btst  d0,4(a3)    ;"
OAF C 6704      1562* boff.s DSshw75        ;"
OAFE 08C2 0004 1563* bset  #4,d2          ;"
OBF0 012B 0005 1564* DSshw75 btst  d0,5(a3)    ;"
OBF4 6704      1565* boff.s DSshw76        ;"
OBF8 08C2 0005 1566* bset  #5,d2          ;"
OBF C 5340      1567* DSshw76 subq  #1,d0         ;indicate another bit used
OBE E EBAA      1568* LSL.L  D5,D2          ;shift char into position
OBI0 CD94      1569* AND.L  D6,(A4)         ;
OBI2 8594      1570* OR.L  D2,(A4)         ;
OBI4 98C3      1571* SUBA.W D3,A4          ;
OBI6 51C9 FF82 1572* DBRA  D1,DSshow7      ;
OBI A 4E75      1573* DSshow9 RTS          ;return
1574*      ;

```

```

1576* ;
1577* ; DScrsR -- cursor right
1578* ;
0B1C 6100 007E 1579* DScrsR BSR DScurs0 ;remove cursor
0B20 3228 0016 1580* DSincx MOVE W WRcursx(A0),D1 ;get current cursor X position
0B24 5C41 1581* ADDQ W #DScellW,D1 ;increment 1 character space 0.6
0B26 3141 0016 1582* MOVE W D1,WRcursx(A0) ;save new cursor X position 0.6
0B2A B268 0012 1583* CMP W WRlngthx(A0),D1 ;at end of line? 0.6
0B2E 6C08 1584* BGE S DSrlini ;yes, do carriage return 0.6
0B30 6068 1585* BRA S DScurs ;show cursor 0.6
1586* ;
1587* ;
1588* ; DScrsU -- cursor up
1589* ;
0B32 6168 1590* DScrsU BSR S DScurs0 ;remove cursor
0B34 6048 1591* BRA S DSdecy ;decrement cursor Y position
1592* ;
1593* ;
1594* ; DSrtrn -- return
1595* ;
0B36 6164 1596* DSrtrn BSR S DScurs0 ;remove cursor
0B38 4268 0016 1597* DSrlini CLR W WRcursx(A0) ;zero current cursor X position
0B3C 0838 0001 0766 1598* BTST #1,CPdspflg.w ;auto line feed?
0B41 6704 1599* BOFF S DSincy ;no, increment cursor Y position
0B44 6054 1600* BRA S DScurs ;show cursor
1601* ;
1602* ;
1603* ; DScrsD -- cursor down
1604* ;
0B46 6154 1605* DScrsD BSR S DScurs0 ;remove cursor
0B48 3228 0016 1606* DSincy MOVE W WRcursy(A0),D1 ;get current cursor Y position
0B4C 8641 000A 1607* ADDI W #DScellY,D1 ;increment 1 character space 0.6
0B50 3141 0018 1608* MOVE W D1,WRcursy(A0) ;save new cursor Y position 0.6
0B54 B268 0014 1609* CMP W WRlngthy(A0),D1 ;at bottom of screen? 0.6
0B58 6F40 1610* BLE S DScurs ;on bottom line? 0.6
0B5A 6000 0084 1611* BRA DScIAL ;yes, wrap to home position 0.6
1612* ;
1613* ;
1614* ; DScrsL -- cursor left
1615* ;
0B5E 613C 1616* DScrsL BSR S DScurs0 ;remove cursor
0B60 4A68 0016 1617* DSdecx TST W WRcursx(A0) ;at beginning of line?
0B64 6712 1618* BEQ S DSwrapx ;yes, wrap to previous line
0B66 5D68 0016 1619* SUBQ W #DScellW,WRcursx(A0) ;decrement 1 character space
0B6A 602E 1620* BRA S DScurs ;show cursor
1621* ;

```

```

1623* ;
1624* ; DScrsH -- cursor home
1625* ;
OB6C 412E      1626* DScrsH BSR.S DScrs0      ;remove cursor
OB6E 4268 0016 1627* DScrsH CLR.W WRcursx(A0) ;zero current cursor X position
OB72 4268 0018 1628* CLR.W WRcursy(A0)      ;zero current cursor Y position
OB76 6022      1629* BRA.S DScrs          ;show cursor
1630*
OB78 4112      1631* DSwrapx BSR.S DSwrap      ;
OB7A 3140 0016 1632* MOVE.W D0,WRcursx(A0)    ;
OB7E 4A68 0018 1633* DSdecy TST.W WRcursy(A0) ;at top line?
OB82 4716      1634* BEQ.S DScrs          ;yes, show cursor
OB84 0468 000A 0018 1635* SUBI.W #DScellY,WRcursy(A0) ;decrement 1 character space
OB8A 609E      1636* BRA.S DScrs          ;show cursor
1637*
OB8C 4280      1638* DSwrap CLR.L D0          ;get current cursor X position
OB8E 3028 0012 1639* MOVE.W WRlength(A0),D0    ;*
OB92 7406      1640* MOVEQ #DScellW,D2        ;get character width
OB94 80C2      1641* DIVU D2,D0          ;
OB96 C0C2      1642* MULU D2,D0          ;
OB98 4E75      1643* RTS              ;return
1644*
OB9A 4130      1645* DScurs BSR.S DScrsAd      ;compute cursor address
OB9C 322A 0004 1646* DScurs0 MOVE.W CS1pch(A2),D1 ;get scan lines per character
OBA0 5341      1647* SUBQ.W #1,D1          ;set loop counter
OBA2 2868 0008 1648* MOVE.L WRcuradr(A0),A4    ;get current cursor address
OBA6 3A28 001A 1649* MOVE.W WRbitofs(A0),D5    ;get current cursor bit offset
OBAA 2E2A 000C 1650* MOVE.L CSmask(A2),D7    ;get character mask
OBAE 0828 0000 0021 1651* BTST #vert,WRattr2(A0) ;vertical orientation?
OBB4 4704      1652* BOFF.S DScurs1        ;no
OBB6 EABF      1653* ROR.L D5,D7          ;
OBB8 6004      1654* BRA.S DScurs2        ;
OBBA 4A64      1655* DScurs1 TST.W -(A4)      ;
OBBC EBBF      1656* ROL.L D5,D7          ;
OBBE 4687      1657* DScurs2 NOT.L D7       ;D7 = positioned inverted mask
OBC0 BF94      1658* DScurs3 EOR.L D7,(A4)    ;invert character
OBC2 98F8 0744 1659* SUBA.W CPscnois.W,A4    ;*
OBC4 51C9 FFF8 1660* DBRA D1,DScurs3      ;*
OBCA 4E75      1661* RTS              ;return
1662*
OBCC 4CA8 0040 0016 1663* DScrsAd MOVEM.W WRcursx(A0),D5-D6 ;get current cursor position
OBD2 4100 00F8 1664* BSR DSadr          ;compute cursor address
OBD6 3147 001A 1665* MOVE.W D7,WRbitofs(A0) ;save cursor bit offset
OBDA 214C 0008 1666* MOVE.L A4,WRcuradr(A0) ;save cursor address
OBDE 4E75      1667* RTS              ;return
1668*

```

```

1670* ;
1671* ; DScIAL -- clear screen
1672* ;
0BE0 6100 FF8C 1673* DScIAL BSR DScrsH1 ;home cursor
1674*
1675* ;
1676* ; DScIES -- clear to end of screen
1677* ;
0BE4 6138 1678* DScIES BSR S DScIEL ;first clear this line
0BE6 0828 0000 0021 1679* BTST #vert,WRattr2(A0) ;vertical orientation?
0BEC 6618 1680* BON S DScIES2 ;yes, clear vertical screen
1681* ; --- clear to end of horizontal screen
0BEE 3C28 0018 1682* MOVE.W WRcursY(A0),D6 ;get current cursor Y position
0BF2 9446 000A 1683* DScIES1 ADDI.W #DScellY,D6 ;increment to next line
0BF6 3028 0014 1684* MOVE.W WRlngthY(A0),D0 ;get bottom of screen limit
0BFA 8C40 1685* CMP.W D0,D6 ;at bottom of screen?
0BFC 4C1E 1686* BGE.S DScIES9 ;yes, return
0BF2E 9046 1687* SUB.W D6,D0 ;compute number of scan lines to clear
0C00 4243 1688* CLR.W D3 ;set starting X position to 0
0C02 6158 1689* BSR.S DScIrH ;clear to bottom of screen
0C04 6016 1690* BRA.S DScIES9 ;return
1691* ; --- clear to end of vertical screen
0C06 3028 0018 1692* DScIES2 MOVE.W WRcursY(A0),D0 ;get current cursor Y position
0C0A 6640 000A 1693* DScIES3 ADDI.W #DScellY,D0 ;increment to next line
0C0E 8068 0014 1694* CMP.W WRlngthY(A0),D0 ;at bottom of screen?
0C12 4C05 1695* BGE.S DScIES9 ;yes, return
0C14 3C00 1696* MOVE.W D0,D6 ;
0C16 4245 1697* CLR.W D5 ;
0C18 6124 1698* BSR.S DScIrV ;clear one vertical line
0C1A 60EE 1699* BRA.S DScIES3 ;repeat until all lines cleared
1700* ;
0C1C 4E75 1701* DScIES9 RTS ;return
1702*
1703* ;
1704* ; DScIEL -- clear to end of line
1705* ;
0C1E 6100 FF7C 1706* DScIEL BSR DScurs0 ;remove cursor
0C22 4CA8 0060 0016 1707* MOVEM.W WRcursX(A0),D5-D6 ;get current cursor X and Y
0C28 0828 0000 0021 1708* BTST #vert,WRattr2(A0) ;vertical orientation
0C2E 4704 1709* BOFF.S DScIEL1 ;no, clear horizontal line
0C30 610C 1710* BSR.S DScIrV ;clear one vertical line
0C32 4006 1711* BRA.S DScIEL2 ;show cursor
1712* ;
0C34 7009 1713* DScIEL1 MOVEQ #DScellY-1,D0 ;D0 = #scanlines to clear
0C36 3605 1714* MOVE.W D5,D3 ;
0C38 6122 1715* BSR.S DScIrH ;clear one horizontal line
0C3A 6000 FF40 1716* DScIEL2 BRA DScurs0 ;show cursor
1717*

```

```

0C3E 3828 0012      1719* DSclrV MOVE.W WRingth(A0),D4      ;get length of line
0C41 9845           1720* SUB.W D5,D4                          ;compute number of scan lines clear
0C44 6100 0086      1721* BSR DSaddr                          ;compute cursor address
0C48 3238 0744      1722* MOVE.W CPsenofs.W,D1                ;get bytes per scan line
0C4C 1C2A 000C      1723* MOVE.L CSmask(A2),D6                ;get character mask
0C50 EEBE           1724* ROR.L D7,D4                          ;align character mask
0C53 CD94           1725* DSclrV1 AND.L D6,(A4)                ;clear one scan line
0C54 98C1           1726* SUBA.W D1,A4                          ;compute address of next scan line
0C5A 51CC FFFA      1727* DBRA D4,DSclrV1                     ;repeat to end of line
0C5A 4E75           1728* RTS                                  ;return
1729*
0C5C 3828 0012      1730* DSclrH MOVE.W WRingth(A0),D4      ;DS = x, D6 = y
0C60 5244           1731* ADDQ.W #1,D4                          ;
0C62 3A04           1732* MOVE.W D4,D5                          ;
0C64 9843           1733* SUB.W D3,D4                          ;
0C66 6100 0064      1734* BSR DSaddr                          ;A4 = addr(x,y), D7 = bitnum
0C6A 9847           1735* SUB.W D7,D4                          ;
0C6C 4A47           1736* TST.W D7                              ;
0C6E 6602           1737* BNE.S DSclrH1                        ;
0C70 4A5C           1738* TST.W (A4)+                          ;
0C72 3604           1739* DSclrH1 MOVE.W D4,D3                ;
0C74 0243 000F      1740* ANDI.W #8F,D3                          ;
0C78 E844           1741* ASR.W #4,D4                          ;
0C7A 5344           1742* SUBQ.W #1,D4                          ;
0C7C 72FF           1743* MOVEQ #-1,D1                          ;
0C7E 3401           1744* MOVE.W D1,D2                          ;
0C80 EF69           1745* LSL.W D7,D1                          ;
0C82 E66A           1746* LSR.W D3,D2                          ;
0C84 2A4C           1747* DSclrH2 MOVE.L A4,A5                ;
0C86 4A47           1748* TST.W D7                              ;
0C88 6702           1749* BEQ.S DSclrH3                        ;
0C8A C35D           1750* AND.W D1,(A5)+                       ;
0C8C 3C04           1751* DSclrH3 MOVE.W D4,D6                ;
0C8E 6B04           1752* BMI.S DSclrH5                        ;
0C90 425D           1753* DSclrH4 CLR.W (A5)+                 ;
0C92 51CE FFFC      1754* DBRA D6,DSclrH4                     ;
0C94 4A43           1755* DSclrH5 TST.W D3                              ;
0C98 6702           1756* BEQ.S DSclrH6                        ;
0C9A C555           1757* AND.W D2,(A5)                         ;
0C9C 98F8 0744      1758* DSclrH6 SUBA.W CPsenofs.W,A4        ;
0CA0 51C8 FFE2      1759* DBRA D6,DSclrH2                     ;
0CA4 4E75           1760* RTS                                  ;return
1761*

```





```

1810* ;
1811* ; jump tables
1812* ;
0D08 FE56      1813* DSctBl  DATA.W DSersL-DScTb1      ;ctl-H: back space
0D0A FF9E      1814*      DATA.W DSstab-DScTb1      ;ctl-I: tab
0D0C FE3E      1815*      DATA.W DSersD-DScTb1      ;ctl-J: line feed
0D0E FE2A      1816*      DATA.W DSersU-DScTb1      ;ctl-K: cursor up
0D10 FE14      1817*      DATA.W DSersR-DScTb1      ;ctl-L: cursor right
0D12 FE2E      1818*      DATA.W DSrtrn-DScTb1      ;ctl-M: carriage return
1819*
0D14 0048 FE58      1820* DSctBl  DATA.W $48,DSersH-DSeTb1      ;esc-H: home cursor
0D18 004A FECC      1821*      DATA.W $4A,DSclAL-DSeTb1      ;esc-J: clear screen
0D1C 004B FF8A      1822*      DATA.W $4B,DSclEL-DSeTb1      ;esc-K: clear to end of line
0D20 0059 FED0      1823*      DATA.W $59,DSclES-DSeTb1      ;esc-Y: clear to end of screen
0D24 FFFF      1824*      DATA.W -1
1825*
0D26 FCE8      1826* DSctBl  DATA.W DSst0-DScTb1      ;state 0
0D28 FD18      1827*      DATA.W DSesc-DScTb1      ;state 1
1828*
0D2A 00010D4E+    1829* DSwndH  DATA.L DSsetH      ;WRcharpt
0D2E 0008D55E      1830*      DATA.L DShomeH      ;home
0D32 0008D55E      1831*      DATA.L DShomeH      ;address
0D36 0000 0000 0000 1832*      DATA.W 0,0,0      ;WRhomeof,WRbases,WRbasey
0D3C 02CF 022F      1833*      DATA.W DSmaxXH,DSmaxYH      ;right,bottom
0D40 0000 0000 0000 1834*      DATA.W 0,0,0      ;x,y,WRbitofs
0D46 0000 022F      1835*      DATA.W 0,DSmaxYH      ;WRgrorgx,WRgrorgy
0D4A 00 1C      1836*      DATA.B 0,$1C      ;attr1,attr2
0D4C 00 24      1837*      DATA.B 0,WRlength
1838*
0D4E 00010D96+    1839* DSsetH  DATA.L DSsetV+CSdata      ;character set record pointer
0D52 000A 8086      1840*      DATA.W DScellY,DScellW      ;CSliph, CSbph
0D56 0020 005A      1841*      DATA.W 32,90      ;CSfrstch, CSLstch
0D5A FFFF FFC0 0000 1842*      DATA.W $FFFF,$FC0,0      ;mask, dummy, attribs
1843*
1844* ; use vertical character set data
1845* ;
1846*

```

```

0D60 00010D64+      1848* DSwndV DATA.L DScsetV      ;WRcharpt
0D64 0008D506      1849*      DATA.L DShomeV      ;,home
0D68 0008D506      1850*      DATA.L DShomeV      ;,address
0D6C 0000 0000 0000 1851*      DATA.W 0,0,0      ;,WRhomeof,WRbasex,WRbasey
0D72 022D 02CF      1852*      DATA.W DSmaxIV,DSmaxTV ;,right,bottom
0D76 0000 0000 0000 1853*      DATA.W 0,0,0      ;,x,y,WRbitofs
0D7C 0000 02CF      1854*      DATA.W 0,DSmaxIV      ;,WRgrorgx,WRgrorgy
0D80 00 10      1855*      DATA.B 0,$10      ;,attr1,attr2
0D82 00 24      1856*      DATA.B 0,WRIlength ;,state, WRrodien
      1857*
0D84 00010E94+      1858* DSsetV DATA.L DScsetV+CSdata ;,character set record pointer
0D88 0006 000A      1859*      DATA.W DScellW,DScellY ;,CSlphch, CSBpch
0D8C 0020 005A      1860*      DATA.W 32,90      ;,CSfirstch, CSlastch
0D90 003F FFFF 0100 1861*      DATA.W $003F,$FFFF,256 ;,mask, dummy, attribs
      1862*
      1863*      ; vertical character set data
      1864*
0D96 00 00 00 00 00 00 00 1865*      DATA.B 0,0,0,0,0,0,0 ;,blank
0D9C 00 00 FD 00 00 00 00 1866*      DATA.B 0,0,$FD,0,0,0 ;,?
0DA2 00 00 00 00 00 00 00 1867*      DATA.B 0,$E0,0,$E0,0,0 ;,"
0DA8 26 FE 26 FE 28 00 1868*      DATA.B $28,$FE,$28,$FE,$28,0 ;,8
0DAE 24 54 FE 54 48 00 1869*      DATA.B $24,$54,$FE,$54,$48,0 ;,9
0DB4 C4 C8 10 26 46 00 1870*      DATA.B $C4,$C8,$10,$26,$46,0 ;,%
0DBA 6C 92 6A 94 0A 00 1871*      DATA.B $6C,$92,$6A,$94,$0A,0 ;,6
0DC0 00 00 20 C6 00 00 1872*      DATA.B 0,0,$20,$C6,0,0,0 ;,
0DC6 00 38 44 82 00 00 1873*      DATA.B 0,$38,$44,$82,0,0,0 ;,(
0DCC 00 00 82 44 38 00 1874*      DATA.B 0,0,$82,$44,$38,0 ;,)
0DD2 08 2A 1C 2A 08 00 1875*      DATA.B $08,$2A,$1C,$2A,$08,0 ;,*
0DD8 08 08 3E 08 08 00 1876*      DATA.B $08,$08,$3E,$08,$08,0 ;,+
0DDE 00 01 07 00 00 00 1877*      DATA.B 0,$01,$07,0,0,0,0 ;,;
0DE4 1C 10 10 10 10 00 1878*      DATA.B $10,$10,$10,$10,$10,0 ;,-
0DEA 00 00 02 00 00 00 1879*      DATA.B 0,0,$02,0,0,0,0 ;,;
0DF0 04 08 10 20 40 00 1880*      DATA.B $04,$08,$10,$20,$40,0 ;,/
0DF6 7C 8A 92 A2 7C 00 1881*      DATA.B $7C,$8A,$92,$A2,$7C,0 ;,0
0DFC 00 42 FE 02 00 00 1882*      DATA.B 0,$42,$FE,$02,0,0,0 ;,1
0E02 46 8A 92 92 62 00 1883*      DATA.B $46,$8A,$92,$92,$62,0 ;,2
0E08 84 82 92 B2 CC 00 1884*      DATA.B $84,$82,$92,$B2,$CC,0 ;,3
0E0E 18 28 48 FE 08 00 1885*      DATA.B $18,$28,$48,$FE,$08,0 ;,4
0E14 E4 A2 A2 A2 9C 00 1886*      DATA.B $E4,$A2,$A2,$A2,$9C,0 ;,5
0E1A 3C 52 92 92 1C 00 1887*      DATA.B $3C,$52,$92,$92,$1C,0 ;,6
0E20 80 8E 90 A0 C0 00 1888*      DATA.B $80,$8E,$90,$A0,$C0,0 ;,7
0E26 6C 92 92 92 6C 00 1889*      DATA.B $6C,$92,$92,$92,$6C,0 ;,8
0E2C 62 92 92 94 78 00 1890*      DATA.B $62,$92,$92,$94,$78,0 ;,9
0E32 00 00 24 00 00 00 1891*      DATA.B 0,0,$24,0,0,0,0 ;,:
0E38 00 01 26 00 00 00 1892*      DATA.B 0,$01,$26,0,0,0,0 ;,;
0E3E 00 10 28 44 82 00 1893*      DATA.B 0,$10,$28,$44,$82,0 ;,<
0E44 00 28 28 28 28 00 1894*      DATA.B 0,$28,$28,$28,$28,0 ;,=
0E4A 00 82 44 28 10 00 1895*      DATA.B 0,$82,$44,$28,$10,0 ;,>
0E50 40 80 9A A0 40 00 1896*      DATA.B $40,$80,$9A,$A0,$40,0 ;,?
0E56 7C 82 9A 9A 7A 00 1897*      DATA.B $7C,$82,$9A,$9A,$7A,0 ;,@
0E5C 3E 48 88 48 3E 00 1898*      DATA.B $3E,$48,$88,$48,$3E,0 ;,A
0E62 FE 92 92 92 6C 00 1899*      DATA.B $FE,$92,$92,$92,$6C,0 ;,B
0E68 7C 82 82 82 44 00 1900*      DATA.B $7C,$82,$82,$82,$44,0 ;,C
0E6E FE 82 82 82 7C 00 1901*      DATA.B $FE,$82,$82,$82,$7C,0 ;,D

```

0E74	FE 92 92 82 82 00	1902*	DATA B	9FE,592,592,582,582,0	, E
0E7A	FE 90 90 80 80 00	1903*	DATA B	9FE,590,590,580,580,0	, F
0E80	7C 82 8A 8A 4C 00	1904*	DATA B	97C,582,58A,58A,54C,0	, G
0E86	FE 10 10 10 FE 00	1905*	DATA B	9FE,510,510,510,9FE,0	, H
0E8C	00 82 FE 82 00 00	1906*	DATA B	0,582,9FE,582,0,0	, I
0E92	04 82 82 FC 80 00	1907*	DATA B	504,482,582,9FC,580,0	, J
0E98	FE 10 28 44 82 00	1908*	DATA B	9FE,510,528,544,582,0	, K
0E9E	FE 02 02 02 02 00	1909*	DATA B	9FE,502,502,502,502,0	, L
0EA4	FE 40 30 40 FE 00	1910*	DATA B	9FE,540,530,540,9FE,0	, M
0EAA	FE 20 10 08 FE 00	1911*	DATA B	9FE,520,510,508,9FE,0	, N
0EB0	7C 82 82 82 7C 00	1912*	DATA B	97C,582,582,582,97C,0	, O
0EB6	FE 90 90 90 60 00	1913*	DATA B	9FE,590,590,590,560,0	, P
0EBC	7C 82 8A 84 7A 00	1914*	DATA B	97C,582,58A,584,57A,0	, Q
0EC2	FE 90 98 94 62 00	1915*	DATA B	9FE,590,598,594,562,0	, R
0EC8	64 92 92 92 4C 00	1916*	DATA B	564,592,592,592,54C,0	, S
0ECE	80 80 FE 80 80 00	1917*	DATA B	980,580,9FE,580,980,0	, T
0ED4	FC 02 02 02 FC 00	1918*	DATA B	9FC,502,502,502,9FC,0	, U
0EDA	F8 04 02 04 F8 00	1919*	DATA B	9F8,504,502,504,9F8,0	, V
0EE8	FC 02 1C 02 FC 00	1920*	DATA B	9FC,502,51C,502,9FC,0	, W
0EE6	C6 28 10 28 C6 00	1921*	DATA B	9C6,528,510,528,9C6,0	, X
0EEC	C0 20 1E 20 C0 00	1922*	DATA B	9C0,520,51E,520,9C0,0	, Y
0EF2	86 8A 92 A2 C2 00	1923*	DATA B	986,58A,592,5A2,9C2,0	, Z
0EF8	0000	1924*	DATA W	0	
		1925*			

```

1927*      include 'CC.FROM.LD'      ;local disk driver
1928* ;
1929* ; File: CC.FROM.LD.TEXT
1930* ; Date: 29-Jun-82
1931* ; By: L. Franklin
1932* ;
1933* ;
1934* ; Lboot -- Local Corvus disk boot processing
1935* ;
0EFA 227C 0000 0771 1936* Lboot  movea.l #CPslotyp.a1      ;get pointer to slot 1 type
0F00 7001          1937*      moveq    #1,d0          ;get initial slot number
          1938* ;
0F01 1231 03FF          1939* Lboot10 move.b  -(a1,d0),d1      ;get device type
0F04 323C 0001          1940*      cmp.b    #DTloc1,d1      ;is this a local disk interface?
0F0A 670C          1941*      beq.s    Lboot30      ;yes, use it for booting
0F0C 5240          1942*      addq    #1,d0          ;update slot number
0F0E B07C 0004          1943*      cmp.w    #4,d0          ;have we looked at all slots?
0F12 4FEE          1944*      ble.s    Lboot10      ;no, check next slot
0F14 7EFF          1945*      moveq    #-1,d7          ;set error return code
0F16 8056          1946*      bra.s    Lboot90      ;return (can not find local disk)
          1947* ;
0F18 11C0 0760          1948* Lboot30 move.b  d0,CPbtslot.w      ;set boot slot number
0F1C 4238 0701          1949*      clr.b    CPbtsrvr.w      ;set boot server number
0F20 4DFA 006A+        1950*      lea    LDblk10,a6      ;set boot disk blk i/o subr pointer
0F24 21CE 0714          1951*      move.l   #46,CPblk10.w      ;*
0F28 4DFA 010C+        1952*      lea    LDdsk10,a6      ;set boot disk i/o subr pointer
0F2C 21CE 0718          1953*      move.l   #46,CPdsk10.w      ;*
          1954* ;
          1955* ; fall through to Lboot10 (used by OMNINET boot too)
          1956* ;
          1957* ;

```

```

1959* ;
1960* ; Lboot80 -- Get 4 boot blocks from Corvus disk
1961* ; (code shared by local disk boot and OMNINET disk boot)
1962* ;
0F30 207C 0088 E000 1963* Lboot80 movea.l #USRbase,a0 ;get block buffer pointer
0F36 1C38 8701 1964* move.b CPBtsrvr.w,d6 ;get boot server number
0F3A 6D32 1965* bit.s Lboot90 ;just return if invalid server number
0F3C E14E 1966* lsl.w #0,d6 ;*
0F3E 1C38 0780 1967* move.b CPBtslot.w,d6 ;get boot slot number
1968* ;
0F42 10BC 0014 1969* move.b #614,(a0) ;set "boot" command
0F46 103C 0007 1970* move.b #607,d0 ;set boot block number
0F4A 323C 0603 1971* move.w #6403,d1 ;*
0F4E 6120 1972* bsr.s LDgetBB ;get next boot block
0F50 6D1C 1973* bit.s Lboot90 ;just return if error
1974* ;*
0F52 323C 0403 1975* move.w #6403,d1 ;*
0F56 6118 1976* bsr.s LDgetBB ;get next boot block
0F58 6D14 1977* bit.s Lboot90 ;just return if error
1978* ;*
0F5A 323C 0203 1979* move.w #6203,d1 ;*
0F5E 6110 1980* bsr.s LDgetBB ;get next boot block
0F60 6D0C 1981* bit.s Lboot90 ;just return if error
1982* ;*
0F62 323C 0083 1983* move.w #63,d1 ;*
0F66 6108 1984* bsr.s LDgetBB ;get next boot block
0F68 6D04 1985* bit.s Lboot90 ;just return if error
0F6A D0FC 0084 1986* adda.w #64,a0 ;get pointer to boot code
1987* ;*
0F6E 4E75 1988* Lboot90 rts ;return
1989* ;*
0F70 1140 0001 1990* LDgetBB move.b d0,(a0) ;set boot block number
0F74 7402 1991* moveq #2,d2 ;get number of bytes to send
0F76 7A33 1992* moveq #DskWrit,d5 ;get "write" command
0F78 4E96 1993* jsr (a6) ;send write command
0F7A D0C1 1994* adda.w d1,a0 ;*
0F7C 343C 0201 1995* move.w #513,d2 ;get number of bytes to receive
0F80 7A32 1996* moveq #DskRead,d5 ;get "read" command
0F82 4E96 1997* jsr (a6) ;send read command
0F84 6D04 1998* bit.s LDgetBX ;just return if error
0F86 90C1 1999* suba.w d1,a0 ;*
0F88 5340 2000* subq #1,d0 ;update boot block number
0F8A 4E75 2001* LDgetBX rts ;return
2002* ;*

```

```

2004* ;
2005* ; LDblkIO - Read or write a local disk block subroutine
2006* ;
2007* ; Enter:  A0.L - Buffer address
2008* ;         D0.W - Block number
2009* ;         D1.W - Drive number
2010* ;         D5.W - Read (432) or Write (433) command
2011* ;         D6.B - Slot number
2012* ;
2013* ; Exit:   A0.L - Next free location in buffer
2014* ;         D0.W - Updated block number
2015* ;         D7.W - IORESULT (disk controller status)
2016* ;
2017* ; All other registers are preserved.
2018* ;
2019* ; Corvus controller status register [3(a1)]:
2020* ;
2021* ; bit 7: controller ready   off - ready       on - not ready
2022* ; bit 6: bus direction    off - host to cntlr on - cntlr to host
2023* ;
0F8C 48E7 2040 2024* LDblkIO  movem i a1/d0-d2,--(sp) ;Save registers
0F90 6100 F6B6 2025*      bsr   SlotAdr      ;A1 = I/O port address
0F94 3405 2026*      move.w d5,d2      ;Send a read (432) or
0F96 6168 2027*      bsr.s LDsend1      ; write (433) block command
0F98 3401 2028*      move.w d1,d2      ;
0F9A 6150 2029*      bsr.s LDsend      ;Send drive number
0F9C 3400 2030*      move.w d0,d2      ;
0F9E 614C 2031*      bsr.s LDsend      ;Send LSB of block
0FA0 E05A 2032*      lsr.w #8,d2      ;
0FA2 6148 2033*      bsr.s LDsend      ;Send MSB of block
0FA4 0C45 0053 2034*      cmpi.w #DskWrt,d5 ;Are we reading or writing?
0FA8 661C 2035*      bne.s LDrd1   ;Reading
2036* ;
2037* ; Write block processing
2038* ;
0FAA 343C 01FF 2039*      move.w #1FF,d2      ;Block size - 1
0FAE 0819 0007 0003 2040* LDw101  btsl #7,3(a1) ;Test controller status
0FB4 66F8 2041*      don.s LDw101      ;Wait until controller ready
0FB6 1358 0001 2042*      move.b (a0)+,1(a1) ;Send a byte
0FB8 51CA FFF2 2043*      dbra d2,LDw101     ;Loop until done
0FBE 6156 2044*      bsr.s LDwait      ;Wait for line to turn
0FC0 1E29 0001 2045*      move.b 1(a1),d7     ;Fetch result code
0FC4 601C 2046*      bra.s LDtrtn   ;Return
2047* ;

```

```

2049*
2050*
2051*
0FC6 614E 2052* LDri01 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* LDri03 btst #7,3(a1) ,Test controller status
0FD2 66F8 2056* bon.s LDri03 ,Wait until controller ready
0FD4 0829 0006 0003 2057* btst #6,3(a1) ,Test bus direction
0FDA 4706 2058* boff.s LDtrtn ,Finished if "host to controller"
0FDC 10E9 0001 2059* move.b 1(a1),(a0)+ ,Store next byte
0FE0 60EA 2060* bra.s LDri03 ,Go get any more
2061*
0FE2 4CDF 0207 2062* LDtrtn movem.l (sp)+,a1/d0-d2 ,Restore registers
0FE4 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*

```



```

2069* ;
2070* ; LDsend -- Send a byte to the disk port subroutine
2071* ;
2072* ; Enter: A1.L - I/O port address
2073* ; D2.B - Byte to send
2074* ;
2075* ; All registers are preserved.
2076* ;
0FFC 0829 0007 0003 2077* LDsend btst #7,3(al) ;Test controller status
0FF2 64F8 2078* bon.s LDsend ;Wait until controller ready
0FF4 1342 0001 2079* move.b d2,1(al) ;Send the byte
0FF8 4E75 2080* rts ;Return
2081* ;
2082* ;
2083* ; LDsend1 -- Send first byte to the disk port subroutine
2084* ;
2085* ; Enter: A1.L - I/O port address
2086* ; D2.B - Byte to send
2087* ;
2088* ; All registers are preserved.
2089* ;
0FFA 46DF 2090* LDsend0 move.w (sp)+,sr ;enable interrupts
0FFC 4E71 2091* nop ;leave some time for interrupt processing
0FFE 4E71 2092* nop ;*
1000 40E7 2093* LDsend1 move.w sr,-(sp) ;save interrupt level
1001 007C 0700 2094* ori.w #0700,sr ;disable interrupts
1006 0829 0007 0003 2095* btst #7,3(al) ;test controller status
100C 66EC 2096* bon.s LDsend0 ;wait until controller ready
100E 1342 0001 2097* move.b d2,1(al) ;send first byte
1012 46DF 2098* move.w (sp)+,sr ;enable interrupts
1014 4E75 2099* rts ;return
2100* ;
2101* ;
2102* ; LDwait -- Wait for the line to turn subroutine
2103* ;
2104* ; Enter: A1.L - I/O port address
2105* ;
1016 2F00 2106* LDwait move.l d0,-(sp) ;save register
1018 7064 2107* moveq #100,d0 ;wait a little bit
101A 51C8 FFEE 2108* LDwait1 dbra d0,LDwait1 ;*
101E 201F 2109* move.l (sp)+,d0 ;restore register
1020 6102 2110* bsr.s LDwait2 ;check two times in case of glitch
1022 4E71 2111* nop ;*
2112* ;
1024 0829 0007 0003 2113* LDwait2 btst #7,3(al) ;test controller status
102A 66F8 2114* bon.s LDwait2 ;wait until controller ready
102C 0829 0006 0003 2115* btst #6,3(al) ;test bus direction
1032 67F0 2116* boff.s LDwait2 ;wait until "controller to host"
1034 4E75 2117* rts ;return
2118* ;

```

```

2120* ;
2121* ; LDdskIO - Read from/Write to Corvus disk
2122* ;
2123* ;      Enter:  A0.L - Buffer address
2124* ;             D2.W - Count
2125* ;             D5.W - Read ($32) or Write ($33) command
2126* ;             D6.B - Slot number
2127* ;
2128* ;      Exit:  D7.W - IORESULT (disk controller status)
2129* ;
2130* ;      All other registers are preserved.
2131* ;
1036 48E7 F0C0      2132* LDdskIO movem.l d0-d3/a0-a1,-(sp);Save registers
103A 6100 F40C      2133* bsr SlotAdr ;A1 = I/O port address
103E 3602           2134* move.w d2,d3 ;get count
1040 5343           2135* subq.w #1,d3 ;Set DBRA loop length
1042 0C45 0033      2136* cmpi.w #DskWrit,d5 ;Are we reading or writing?
1044 6614           2137* bne.s LDdsk2 ;Reading
2138* ;
2139* ; Write Corvus disk processing
2140* ;
1048 1418           2141* move.b (a0)+,d2 ;get first byte
104A 61B4           2142* bsr.s LDsend1 ;send first byte
104C 4006           2143* bra.s LDdsk1a ;send rest of bytes
104E 1418           2144* LDdsk1 move.b (a0)+,d2 ;get next byte
1050 6100 FF9A      2145* bsr LDsend ;send next byte
1054 51CB FF88      2146* LDdsk1a dbra d3,LDdsk1 ;loop until done
1058 7E00           2147* moveq #0,d7 ;force successful result code
105A 601E           2148* bra.s LDdsk9 ;finished
2149* ;
2150* ; Read Corvus disk processing
2151* ;
105C 61B8           2152* LDdsk2 bsr.s LDwait ;Wait for the line to turn
105E 1E29 0001      2153* move.b 1(a1),d7 ;Fetch result code
1062 10C7           2154* move.b d7,(a0)+ ;Store first byte
2155* ;
1064 0829 0007 0003 2156* LDdsk3 btst #7,3(a1) ;Test controller status
106A 64F8           2157* bon.s LDdsk3 ;Wait until controller ready
106C 0829 0006 0003 2158* btst #6,3(a1) ;Test bus direction
1072 6706           2159* boff.s LDdsk9 ;Finished if "host to controller"
1074 10E9 0001      2160* move.b 1(a1),(a0)+ ;Store next byte
1078 60EA           2161* bra.s LDdsk3 ;Go get any more
2162* ;
107A 4CDF 030F      2163* LDdsk9 movem.l (sp)+,d0-d3/a0-a1;Restore registers
2164* ; ---- move.b d7,CPdiskRC.w ;Save current disk return code
107E 4887           2165* ext.w d7 ;Set return condition code
1080 4E75           2166* rts ;Return
2167* ;

```

```

2169* ;
2170* ; LDsync -- Synchronise with Corvus disk controller
2171* ;
2172* ; Enter. D6.B - slot number
2173* ;
2174* ; Exit. D7.W - 0 = no timeout (EQ), -1 = timeout (NE)
2175* ;
2176* ; All other registers are preserved.
2177* ;
1082 48E7 8040 2178* LDsync moven.l d0/ai,-(sp) ;save registers
1086 6100 F3C9 2179* bsr SlotAdr ;get slot address
108A 3E3C 07D0 2180* move.w #2000,d7 ;set timeout counter
2181* ;
108E 137C 00FF 0001 2182* LDsync1 move.b #$FF,1(a1) ;send invalid command to controller
1094 303C 0400 2183* move.w #1024,d0 ;wait about 1 ms
1098 51C8 FFFE 2184* LDsync2 dbra d0,LDsync2 ;*
109C 0829 0006 0003 2185* btst #6,3(a1) ;test bus direction
10A2 6606 2186* bon.s LDsync3 ;go on if "controller to host"
10A4 51CF FFE8 2187* dbra d7,LDsync1 ;send invalid command again
10A8 600C 2188* bra.s LDsync5 ;set timeout error and return
2189* ;
10AA 8829 0007 0003 2190* LDsync3 btst #7,3(a1) ;test controller status
10B0 6706 2191* boff.s LDsync6 ;go on if controller ready
10B2 51CF FFF6 2192* dbra d7,LDsync3 ;check controller status again
2193* ;
10B6 7EFF 2194* LDsync5 moveq #-1,d7 ;indicate controller timeout
10B8 600A 2195* bra.s LDsync9 ;return
2196* ;
10BA 0C29 008F 0001 2197* LDsync6 cmpi.b #$8F,1(a1) ;did controller respond with error?
10C0 66CC 2198* bne.s LDsync1 ;no, send invalid command again
10C2 7E00 2199* moveq #0,d7 ;indicate no controller timeout
2200* ;
10C4 4CDF 0201 2201* LDsync9 moven.l (sp)+,d0/ai ;restore registers
10C8 4A47 2202* tst.w d7 ;set return condition code
10CA 4E75 2203* rts ;return
2204* ;
2205* ;

```

```

2207*      include 'CC.PROM.OD'      ;OMNINET disk driver
2208* ;
2209* ; File: CC.PROM.OD.TEXT
2210* ; Date: 29-Jun-82
2211*
2212* ;
2213* ; OMNINET disk driver data area equates
2214* ;
00000000 2215* DCmd   EQU   0      ,byte - disk command offset
00000001 2216* DCdrv  EQU   1+DCmd ,byte - offset for drive number
00000002 2217* DCbkl0 EQU   2+DCmd ,byte - LSB of block number to read or write
00000003 2218* DCbkl1 EQU   3+DCmd ,byte - MSB      "      "
00000004 2219* DClen  EQU   4+DCmd ,word - length of request (in bytes)
2220* ;
2221* ;      result vector and header used for all setuprcv commands
2222* ;
00000006 2223* RHdr   EQU   6      ;
00000006 2224* RHpktRc EQU   0+RHdr ,byte - return code from transporter
00000007 2225* RHsor  EQU   1+RHdr ,byte - the source of the message
00000008 2226* RHpktLN EQU   2+RHdr ,word - total length of data portion of packet
0000000A 2227* RHdskLN EQU   4+RHdr ,word - length of info returned from drive
0000000C 2228* RHdskRc EQU   6+RHdr ,byte - return code from drive
2229* ;
2230* ;      result vector and header for all sendsq commands
2231* ;
0000000E 2232* SHdr   EQU   14     ;
0000000E 2233* SHpktRc EQU   0+SHdr ,byte - return code from transporter
2234* ;      EQU   1+SHdr ,byte - unused
2235* ;      EQU   2+SHdr ,word - unused
00000012 2236* SHtoLN  EQU   4+SHdr ,word - number of bytes to send to drive
00000014 2237* SHfmLN  EQU   6+SHdr ,word - number of bytes expected from drive
2238* ;
00000016 2239* GData  EQU   22     ,word - area to receive "GO" into
2240* ;
2241* ;      area used for constructing Transporter commands
2242* ;
00000018 2243* TCmd   EQU   24     ;
00000018 2244* TCop   EQU   0+TCmd ,byte - op code
00000019 2245* TCradhi EQU   1+TCmd ,byte - result address HI
0000001A 2246* TCradlo EQU   2+TCmd ,word - result address MED, LO
0000001C 2247* TCsock  EQU   4+TCmd ,byte - socket number
0000001D 2248* TCdAdhi EQU   5+TCmd ,byte - data buffer address HI
0000001E 2249* TCdAdlo EQU   6+TCmd ,word - data buffer address MED, LO
00000020 2250* TCdAtLN EQU   8+TCmd ,word - data length
00000022 2251* TCHdrLN  EQU  10+TCmd ,byte - header length
00000023 2252* TCdest  EQU  11+TCmd ,byte - destination host number
2253* ;
00000024 2254* ODdw   EQU   36     ,lint - temporary buffer (for 3 byte numbrs)
00000025 2255* ODdwhi EQU   37     ,byte - temporary HI
00000026 2256* ODdwl0 EQU   38     ,word - temporary MID, LO
00000028 2257* ODwrAD EQU   40     ,lint - to save buffer address for CWrites
0000002C 2258* ODvalid EQU   44     ,word - for marking buifer as valid
2259*

```

```

2261* ,
2262* ,
2263* ,
00030FA1 2264* StrAdr EQU  $30FA1 ,address of Transporter register
00030F7F 2265* RdyAdr EQU  $30F7F ,address of VIA register A, used for Omninet ready
0000FFFE 2266* Tointvl EQU  $FFFE ,timeout interval
2267* ,
2268* , Transporter Return Codes
2269* ,
000000FF 2270* Waiting EQU  $FF ,
000000FE 2271* CmdAcpt EQU  $FE ,
000000C0 2272* Echoed EQU  $C0 ,echo command was successful
2273* ,
00000080 2274* GaveUp EQU  $80 ,aborted a send command after MaxRetries tries
00000081 2275* TooLong EQU  $81 ,last message sent was too long for the receiver
00000082 2276* NoSckt EQU  $82 ,sent to an uninitialized socket
00000083 2277* HdrErr EQU  $83 ,sender's header length did not match receiver's
00000084 2278* BadSock EQU  $84 ,illegal socket number
00000085 2279* Inuse EQU  $85 ,tried to set up a receive on an active socket
00000086 2280* BadDest EQU  $86 ,sent to an illegal host number
2281* ,
00000090 2282* NoTrans EQU  $90 ,could not strobe cmd addr to Transporter
00000091 2283* TimeOut EQU  $91 ,timed out waiting for an Omninet event
00000092 2284* NoBufr EQU  $92 ,tried a CRRead without a valid write buffer
2285* ,
2286* , Transporter Opcodes
2287* ,
000000F0 2288* RecvOp EQU  $F0 ,SETUPRECV opcode
00000040 2289* SendOp EQU  $40 ,SENDMSG opcode
00000020 2290* InitOp EQU  $20 ,INIT opcode
00000010 2291* EndOp EQU  $10 ,ENDRECV opcode
00000008 2292* DebOp EQU  $08 ,PEEK/POKE opcode
00000002 2293* EchoOp EQU  $02 ,ECHOCHD opcode
00000001 2294* WhoOp EQU  $01 ,WHOAMI opcode
2295* ,
000000A0 2296* RestSkt EQU  $A0 ,dest. socket for REST packet
000000E0 2297* CnstSkt EQU  $E0 ,socket for Constellation protocol
2298* ,

```

```

2300* ;
2301* ; Oboot -- OMNINET disk server boot processing
2302* ;
10CC 11FC 0005 0700 2303* Oboot   move.b #5,CPbtslot.w      ;set boot slot number
10D1 40FA 003C+ 2304*         lea   ODbk10,a6        ;set boot disk blk i/o subr pointer
10D4 21CE 0714 2305*         move.l a6,CPblkio.w      ;,*
10DA 40FA 00B4+ 2306*         lea   ODdsk10,a6       ;set boot disk i/o subr pointer
10DE 21CE 0718 2307*         move.l a6,CPdskio.w      ;,*
10E2 6000 FEAC 2308*         bra   Lboot80          ;load boot code
2309*         ; (Lboot80 is in CC.PROM.LD)
2310* ;
2311* ;
2312* ; ODcmdnd -- send simple command to Transporter
2313* ;
2314* ;         Enter: D0.B - Transporter command
2315* ;         D1.B - Destination host number (if echo)
2316* ;
2317* ;         Exit: D7.B - IORESULT (OMNINET status)
2318* ;
10E6 40E7 8040 2319* ODcmdnd  move.l a1-a2/d0,-(sp)      ;save registers
10EA 227C 0008 DF00 2320*         move.l #CPomnibf,a1      ;get pointer to Data Area
10F0 4069 0004 2321*         pea   RHdr(a1)          ;get pointer to result record
10F4 235F 0018 2322*         move.l (sp)+,TCop(a1)      ;set result record pointer
10F8 1340 0018 2323*         move.b d0,TCop(a1)      ;set Transporter command
10FC 1341 001C 2324*         move.b d1,TCsock(a1)     ;set destination host number (echo)
1100 137C 00FF 0004 2325*         move.b #Waiting,RHpktRC(A1) ;set Transporter waiting flag
1104 45E9 0018 2326*         lea   TCmd(A1),A2        ;get command address
110A 6100 00EE 2327*         bsr   Strobe1          ;strobe command address to Transporter
110E 661A 2328*         bne.s ODcmd9          ;Transporter not responding
1110 303C FFFE 2329*         move.w #TOintvl,D0      ;get timeout interval
2330*         ;
1114 0C29 00FF 0004 2331* ODcmd1  cmpi.b #Waiting,RHpktRC(A1) ;has Transporter responded?
111A 660A 2332*         bne.s ODcmd2          ;yes, ready to return
111C 51C8 FFF6 2333*         dbra  D0,ODcmd1          ;timeout yet?
1120 137C 0091 0004 2334*         move.b #TimeOut,RHpktRC(A1) ;yes, set timeout error and return
2335*         ;
1124 1E29 0006 2336* ODcmd2  move.b RHpktRC(A1),d7      ;get Transporter return code
2337*         ;
112A 4CDF 3601 2338* ODcmd9  move.l (sp)+,a1-a2/d0      ;restore registers
2339* ; ---- move.b d7,CPomniRC.w ;save current OMNINET return code
112E 4E75 2340*         rts          ;return
2341* ;

```

```

2343*
2344* ODbkIO - 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 (532) or Write (533) 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* ODbkIO movem.l d0-d2/a1-a2,-(sp),Save registers
1134 227C 0008 DF00 2359*      move.l @CPommbf,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
1144 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  ODbkI2 ;Reading
2369*
115C 337C 0204 0012 2370* ODbkI1 move.w #516,SHtoLN(A1) ;number of bytes to send to drive
1162 4269 0014 2371*      clr.w  SHfLn(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  ODbkI3 ;return
2375*
1170 337C 0004 0012 2376* ODbkI2 move.w #4,SHtoLN(A1) ;number of bytes to send to drive
1174 337C 0200 0014 2377*      move.w #512,SHfLn(A1) ;number of bytes expected back
117C 6100 024C 2378*      bsr   ShortCmds ;
2379*
1180 4CDF 0607 2380* ODbkI3 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*

```

```

2388* ;
2389* , ODDsk10 - Read from/write to Corvus disk
2390* ;
2391* ; Enter: A0.L - Buffer address
2392* ; D2.W - Count
2393* ; D5.W - Read ($32) or Write ($33) command
2394* ; D4.W - Destination host number * 256
2395* ;
2396* ; Exit: D7.W - IORESULT (OMNINET/disk controller status)
2397* ;
2398* ; All other registers are preserved.
2399* ;
1190 48E7 E0E0 2400* ODDsk10 movem.l d0-d2/a0-a2,-(sp),Save registers
1194 227C 0008 DF00 2401* move.l #CPomnibf,a1 ,A1 points to the start of the Data Area
119A 0C45 0033 2402* cmpi.w #DskWrit,D5 ;do we want to read or write
119E 6616 2403* bne.s ODDsk2 ;read
2404* ;
11A0 3342 0012 2405* ODDsk1 move.w D2,SHtoLN(A1) ;
11A4 2358 0000 2406* move.l (A0)+,DCmd(A1) ;move first four bytes of send data to DiskCmd
11A8 2348 0028 2407* move.l A0,ODwrAD(A1) ;save address of REST of data
11AC 4247 2408* clr.w D7 ;force successful IOResult
11AE 337C FFFF 002C 2409* move.w #-1,ODvalid(A1) ;mark send buffer as valid...
11B4 603C 2410* bra.s ODDsk9 ;return
2411* ;
11B6 0C69 FFFF 002C 2412* ODDsk2 cmpi.w #-1,ODvalid(A1) ;is send buffer valid??
11BC 6706 2413* beq.s ODDsk3 ;yes, go on
11BE 3E3C 0092 2414* move.w #NoBufR,D7 ;set IOResult to "no buffer" error
11C2 602C 2415* bra.s ODDsk6 ;return
2416* ;
11C4 4269 002C 2417* ODDsk3 clr.w ODvalid(A1) ;mark send buffer as invalid
11C8 3342 0014 2418* move.w D2,SHfmLN(A1) ;
11CC 0469 0001 0014 2419* sub1.w #1,SHfmLN(A1) ;subtract one for the return code
11D2 D1FC 0000 0001 2420* adda.l #1,A0 ;inc buffer pointer past return code
11D8 0C69 0004 0012 2421* cmpi.w #4,SHtoLN(A1) ;are we doing a longcmd?
11DE 6206 2422* bhi.s ODDsk4 ;yes
11E0 6100 01E0 2423* bsr ShortCmds ;no
11E4 6004 2424* bra.s ODDsk5 ;
11E6 6100 016E 2425* ODDsk4 bsr LongCmds ;
11EA 91FC 0000 0001 2426* ODDsk5 suba.l #1,A0 ;dec buffer pointer past return code
2427* ;

```



```
11F0          2429* 0Ddsk6 ;
                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* 0Ddsk9 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*
```

```

2441* ;
2442* ; Stroblt -- Strobe command address to Transporter
2443* ;
2444* ; Enter: A2 = command address
2445* ;
2446* ; Exit: D7 = Transporter strobe status
2447* ;
2448* ; EQ = successful
2449* ; NE = Transporter not responding
2450* ;
2451* ; All other registers are preserved
2452* ;
11FA 48E7 C000 2453* Stroblt movem.l D0-D1,-(sp) ;save registers
11FE 7E00 2454* moveq #0,d7 ;assume no Transporter error
1200 200A 2455* move.l A2,D0 ;get command address
1202 E198 2456* rol.l #8,D0 ;move command address to msb
2457* ;
1204 611A 2458* bsr.s SBstrob ;strobe address HI
1206 670C 2459* beq.s SBerr ;
1208 4116 2460* bsr.s SBstrob ;strobe address MED
120A 6708 2461* beq.s SBerr ;
120C 6112 2462* bsr.s SBstrob ;strobe address LO
120E 6704 2463* beq.s SBerr ;
1210 6116 2464* bsr.s SBwait ;wait for Transporter ready
1212 6604 2465* bne.s SBexit ;
2466* ;
1214 3E3C 0090 2467* SBerr move.w #NoTrans,d7 ;no transporter ...
2468* ;
1218 4CDF 0003 2469* SBexit movem.l (sp)+,D0-D1 ;restore registers
121C 4A47 2470* tst.w d7 ;set return condition code
121E 4E75 2471* rts ;return
2472* ;
1220 E198 2473* SBstrob rol.l #8,D0 ;shift address byte to lsb
1222 13C0 0003 OFAI 2474* move.b D0,StrAdr.L ;strobe address
1228 323C FFFE 2475* SBwait move.w #TOintvl,D1 ;get timeout interval
122C 0839 0000 0003 2476* SBW1 bts.l #0,RdyAdr.L ;is transporter ready?
1232 0F7F
1234 6600 0006 2477* bon SBVexit ;yes, return
1238 51C9 FFF2 2478* dbra D1,SBW1 ;repeat until transporter ready
123C 4E75 2479* SBVexit rts ;return
2480* ;

```

```

2482* ;
2483* ; SetGo -- set up a receive for the 'GO' packet
2484* ;
123E 337C 0002 0020 2485* SetGo move.w #2,TCdtaLN(A1) ;2 bytes of data
1244 4129 0012 2486* clr.b TChdrLN(A1) ;no header
1248 4869 0016 2487* pea GData(A1) ;get address of data area
2488* ;---- move.l (SP)+,ODdw(A1) ;
2489* ;---- move.b ODdwhi(A1),TCdADhi(A1) ;load data buffer address
2490* ;---- move.w ODdwlo(A1),TCdADlo(A1) ;*
124C 235F 001C 2491* move.l (SP)+,TCdADhi-1(A1) ;same as above -- TCsock destroyed
1250 6010 2492* bra.s SetGoI
2493* ;
2494* ; SetRecv -- set up a receive for the disk results and read data
2495* ; returns result in D0
2496* ;
1252 2497* SetRecv
1498* ;---- move.l A0,ODdw(A1) ;load data buffer address
2499* ;---- move.b ODdwhi(A1),TCdADhi(A1) ;*
2500* ;---- move.w ODdwlo(A1),TCdADlo(A1) ;*
1252 2348 001C 2501* move.l A0,TCdADhi-1(A1) ;same as above -- TCsock destroyed
1256 3369 0014 0020 2502* move.w SHfnLN(A1),TCdtaLN(A1) ;
125C 137C 0003 0022 2503* move.b #3,TChdrLN(A1) ;disk results have a hdr len of 3
2504* ;
1262 137C 00FF 0006 2505* SetGoI move.b #Waiting,RHpktRC(A1) ;set result to FF to see it change
2506* ;
2507* ; prepare the command vector
1268 4869 0006 2508* pea RHdr(A1) ;load result vector address
2509* ;---- move.l (SP)+,ODdw(A1) ;*
2510* ;---- move.b ODdwhi(A1),TCrADhi(A1) ;*
2511* ;---- move.w ODdwlo(A1),TCrADlo(A1) ;*
126C 235F 0018 2512* move.l (SP)+,TCrADhi-1(A1) ;same as above -- TCop destroyed
1270 137C 00F0 0018 2513* move.b #RecvOp,TCop(A1) ;set up a receive
1276 137C 00B0 001C 2514* move.b #CnstSkt,TCsock(A1) ; on socket B0
2515* ;
127C 45E9 0018 2516* lea TCmd(A1),A2 ;get command address
1280 6100 FF78 2517* bsr StrobIt ;strobe command address to Transporter
1284 6600 00C4 2518* bne SCerr2 ;Transporter not responding
1288 303C FFFE 2519* move.w #TOintvl,D0 ;for time out
2520* ;
128C 0C29 00FF 0006 2521* SC10 cmpi.b #Waiting,RHpktRC(A1)
1292 6608 2522* bne.s SC12 ;wait till result changes
1294 51C8 FFF6 2523* dbra D0,SC10
1298 6000 00B4 2524* bra SCerr3 ;timeout error
2525* ;
129C 1029 0006 2526* SC12 move.b RHpktRC(A1),d0 ;get Transporter return code
12A0 0C00 00FE 2527* cmpi.b #CmdAcpt,d0 ;was command accepted?
12A4 6D00 009E 2528* bit SCerr1 ;no, fatal error
12A8 4240 2529* clr.w D0 ;indicate success
12AA 6000 00A6 2530* bra SCexit ;return
2531* ;

```

```

2533* ;
2534* ; SndRest -- send the rest of the data (from long command) to the disk server
2535* ; result of call is in D0, 0 = success
2536* ;
12AE 1349 0029 001D 2537* SndRest move.b ODWrAD+1(A1),TCdADhi(A1),load data buifer address
12BA 3369 002A 001E 2538* move.w ODWrAD+2(A1),TCdADio(A1),*
12BA 137C 00A0 001C 2539* move.b #RestSkt,TCsock(A1),*
12C0 0469 0004 0012 2540* subi.w #4,SHtoLN(A1),*
12C6 4C04 2541* bge.s SC20,*
12C8 4269 0012 2542* cfr.w SHtoLN(A1),*result was negative, make it zero
12CC 3369 0012 0020 2543* SC20 move.w SHtoLN(A1),TCdtaLN(A1),*send length - 4 bytes
12D2 4229 0022 2544* cfr.b TChdrLN(A1),*no header for rest packets
12D6 6032 2545* bra.s SC40,*send it
2546* ;
2547* ; SndCmds -- send a disk command to the disk server
2548* ; result of call is in D0, 0 = success
2549* ;
12D8 4869 0000 2550* SndCmds pea DCmd(A1),*data is the Disk command
2551* ; ---- move.l (SP)+,ODdw(A1),*
2552* ; ---- move.b ODdwhi(A1),TCdADhi(A1),*load data buifer address
2553* ; ---- move.w ODdwlo(A1),TCdADio(A1),*
12DC 235F 001C 2554* move.l (SP)+,TCdADhi-1(A1),*same as above -- TCsock destroyed
12E0 137C 00B0 001C 2555* move.b #CnstSkt,TCsock(A1),*
12E4 0C69 0004 0012 2556* cmpl.w #4,SHtoLN(A1),*are we sending less than 4 bytes
12EC 6C08 2557* bge.s SC30,*no
12EE 3369 0012 0020 2558* move.w SHtoLN(A1),TCdtaLN(A1),*less
12F4 4006 2559* bra.s SC32,*
12F6 337C 0004 0020 2560* SC30 move.w #4,TCdtaLN(A1),*disk command is 4 bytes long
12FC 137C 0004 0022 2561* SC32 move.b #4,TChdrLN(A1),*send header is 4 bytes
1302 E05E 2562* ror.w #8,d6,*set destination host number
1304 1346 0023 2563* move.b d6,TCdest(A1),*
1308 E15E 2564* rol.w #8,d6,*
2565* ;
130A 137C 00FF 000E 2566* SC40 move.b #Waiting,SHpktRC(A1),*set result to FF to see it change
1310 4869 000E 2567* pea SHdr(A1),*load result vector address
2568* ; ---- move.l (SP)+,ODdw(A1),*
2569* ; ---- move.b ODdwhi(A1),TCrADhi(A1),*
2570* ; ---- move.w ODdwlo(A1),TCrADio(A1),*
1314 235F 0018 2571* move.l (SP)+,TCrADhi-1(A1),*same as above -- TCop destroyed
1318 137C 0040 0018 2572* move.b #SendOp,TCop(A1),*sendmsg opcode
131E 45E9 0018 2573* SC50 lea TCmd(A1),A2,*get command address
1322 6100 FED6 2574* bsr StrobIt,*strobe command address to Transporter
1326 6622 2575* bne.s SCerr2,*Transporter not responding
2576* ;

```

```

1328 303C FFFE      2578*      move.w #TOintvl,D0      ;for time out
132C 0C2F 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 4A2F 000E      2585*      tst.b SHpktRC(A1)      ;did it work?
1340 6DD2      2586*      bit.s SCerr1      ;no, fatal error
1342 600E      2587*      bra.s SCexit      ;return
      2588*      ;
1344 102F 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*

```

```

1354          2599* LongCmds
          2600* ;
          2601* ; 1. set up a receive for the GO message
          2602* ;
1354 6100 FEE6          2603*      bsr   SetGo           ;
135A 6D62          2604*      bit.s  LcmdErr          ;if DO ( 0 then fatal DRW error
          2605* ;
          2606* ; 2. send disk command
          2607* ;
135C 6100 FF7A          2608*      bsr   SndCmds          ;doit
1360 6D5C          2609*      bit.s  LcmdErr          ;if DO ( 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  #CmdAopt,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*      bit.s  LcmdErr          ;no, set error return
          2618* ;
          2619* ; 4. validate GO packet
          2620* ;
1370 0829 0007 0016          2621* Lcmd3  btst   #7,Gdata(A1)      ;
1376 46DE          2622*      bon.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*      bit.s  LcmdErr          ;
1390 60D0          2631*      bra.s  Lcmd1          ;
          2632* ;

```

```

2634* ,
2635* , 5. set up receive for results
2636* ,
1372 6100 FE8E 2637* Lcmd5 bsr SetRecv ,
1396 6D26 2638* bit s LcmdErr ,if D0 < 0 then fatal DRW error
2639* ,
2640* , 6. send REST
2641* ,
1398 6100 FF14 2642* bsr SndRest ,
139C 6D20 2643* bit s LcmdErr ,
2644* ,
2645* , 7. wait for results
2646* ,
139E 1029 0006 2647* Lcmd6 move.b RHpktRC(A1),D0 ,get Transporter return code
13A2 0C00 00FE 2648* cmpi.b #CmdAcpt,D0 ,has return code changed?
13A6 67F6 2649* beq.s Lcmd6 ,no, wait some more
13A8 4A00 2650* tst.b D0 ,successful receive?
13AA 6D12 2651* bit s LcmdErr ,no, set error return
2652* ,
2653* , 8. validate results
2654* ,
13AC 1029 0013 2655* Lcmd7 move.b TCdest(A1),D0 ,
13B0 6D29 0007 2656* cmp.b RHsor(A1),D0 ,did response come from the right place?
13B4 670C 2657* beq.s LcmdOK ,yes
13B6 6100 FE9A 2658* bsr SetRecv ,No, set up receive again....
13BA 6D02 2659* bit s LcmdErr ,if D0 < 0 then fatal DRW error
13BC 6D04 2660* bra.s Lcmd5 ,go back and wait again...
2661* ,
13BE 1E00 2662* LcmdErr move.b D0,D7 ,get error return code
13C0 6004 2663* bra.s LcmdEx ,return
2664* ,
13C2 1E29 000C 2665* LcmdOK move.b RHdskRC(A1),D7 ,get disk server return code
2666* ,
13C6 4887 2667* LcmdEx ext.w D7 ,make return code a word
13C8 4E75 2668* rts ,return for ShortCmds or LongCmds
2669* ,

```

```

13CA          2671* ShortCmds
                2672* ;
                2673* ; 1. set up a receive for the results
                2674* ;
13CA 6100 FE86 2675*      bsr      SetRecv      ;
13CE 6DEE      2676*      bit.s   LcmdErr      ;if DO ( 0 then fatal DRW error
                2677* ;
                2678* ; 2. send disk command to disk server
                2679* ;
13D0 6100 FF06 2680*      bsr      SndCmds      ,doit
13D4 6DE8      2681*      bit.s   LcmdErr      ;if DO ( 0 then fatal DRW error
                2682* ;
                2683* ; 3. wait to receive results
                2684* ;
13D6 2E3C 0004 0000 2685*      move.l  $$40000,d7      ;for time out
13DC 1029 0006      2686*      Scmd2   move.b   RHpk(RC(A1),d0      ;get Transporter return code
13E0 6C0C      2687*      bge.s   Scmd3      ;successful receive, go on
13E2 0C00 00FE      2688*      cmpi.b  #CmdAcpt,d0      ;has return code changed?
13E6 66D6      2689*      bne.s   LcmdErr      ;yes, set error return
13E8 5387      2690*      subq.l  #1,d7      ;time out?
13EA 66F0      2691*      bne.s   Scmd2      ;no, wait some more
13EC 6DD0      2692*      bra.s   LcmdErr      ;set error return
                2693* ;
                2694* ; 4. validate results
                2695* ;
13EE 1E29 0007      2696*      Scmd3   move.b   RHsor(A1),D7      ;get source of response
13F2 0C29 00FF 0023 2697*      cmpi.b  $$FF,TCdest(a1) ;is this a broadcast?
13F8 6606      2698*      bne.s   Scmd4      ;no, go on
13FA 1347 000C      2699*      move.b  d7,RHdskRC(A1) ;save disk server number
13FE 60C2      2700*      bra.s   LcmdOk      ;return
                2701* ;
1400 BE29 0013      2702*      Scmd4   cmp.b   TCdest(A1),D7      ;did response come from the right place?
1404 678C      2703*      beq.s   LcmdOK      ;yes
1406 6100 FE4A      2704*      bsr      SetRecv      ;set up receive again...
140A 6DB2      2705*      bit.s   LcmdErr      ;if DO ( 0 then fatal DRW error
140C 60CE      2706*      bra.s   Scmd2      ;go back and wait again...
                2707* ;

```



```

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* ; ----  cfr.b  CPbtsrvr.w      ,set boot server number (already 0)
1414 487A 0034*      2723*      pea   FDbkIO      ,set boot disk blk i/o subr pointer
141A 11DF 0714      2724*      move.l  (sp)+,CPbkio.w      ;*
141E 487A 0044*      2725*      pea   FDsecIO      ,set boot disk sector i/o subr pointer
1422 11DF 0718      2726*      move.l  (sp)+,CPdskio.w      ;*
2727* ; ----  moveq  #0,d0      ; (already 0)
2728* ; ----  move.b  d0,CPosstrv.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   FD18ssd      ,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
1434 7000          2736*      moveq  #0,d0      ;
1438 5200          2737*      move.w  d0,d1      ;
143A 7A32          2738*      moveq  #DskRead,d5      ,get read block function code
2739* ;
143C 610E          2740*      bsr.s  FDbkIO      ,read block 1 of boot code
143E 6D0A          2741*      blt.s  Fboot90      ,just return if error
1440 610A          2742*      bsr.s  FDbkIO      ,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*

```

```

2749* ;
2750* ; PHILOSOPHY: The user views floppy as a set of 512 byte blocks
2751* ; The driver then translates this block to track address, sector
2752* ; address, side.
2753* ; It then makes the necessary number of request to read sectors.
2754* ; Partial sectors are not read or written, the excess is ignored
2755* ; Sector length of an Apple floppy is 256 bytes.
2756* ;
2757* ; RESTRICTION: Bytes per sector must be exact divisor of 512 (block size).
2758* ; The block address must be less than (2**15)/bytes per sector,
2759* ; so that when sector is formed, it fits in the D3.W.
2760* ;
2761* ; RESULTS OF SOME COMMANDS:
2762* ;
2763* ; 1) TRACK REG is incremented by 1 during STEPIN even
2764* ; though there is a seek error because the TRACK addr
2765* ; requested exceeds the maximum track address allowed
2766* ;
2767* ; 2) STEPOUT. the track register is not decremented below 0
2768* ; after TRK00. The seek error bit is set.
2769* ;
2770* ;
2771* ;
2772* ; FDbk10 - Read/Write a Corvus floppy disk block subroutine
2773* ;
2774* ; Enter: A0.L - Buffer address
2775* ;         D0.W - Block number
2776* ;         D1.W - Drive number
2777* ;         D5.W - Read (532) or Write (533) command
2778* ;
2779* ; Exit:  A0.L - Next free location in buffer
2780* ;         D0.W - Updated block number
2781* ;         D7.W - IORESULT
2782* ;
2783* ; All other registers are preserved.
2784* ;
144C 48E7 FE7E 2785* FDbk10 MOVEM.L D0-D6/A1-A6,-(SP) ;
1450 343C 0200 2786* MOVE.W #BLKSZ,D2 ;BLOCK SIZE IN BYTES
1454 3600 2787* MOVE.W D0,D3 ;
1456 6100 00D0 2788* BSR FDrdrw ;
145A 4CDF 7E7F 2789* MOVEM.L (SP)+,D0-D6/A1-A6 ;restore registers
145E 5240 2790* ADDQ.W #1,D0 ;INC BASE BLOCK
1460 4A07 2791* TST.B D7 ;set return condition code
1462 4E75 2792* RTS ;return
2793* ;

```

```

2795* ;
2796* ; FDsectIO - Read/Write a Corvus floppy disk sector
2797* ;
2798* ; Enter: A0.L - Buffer address
2799* ;         D1.W - Bytes per sector (128 for single density)
2800* ;         D3.W - Track number
2801* ;         D4.W - Sector number
2802* ;         D5.W - Read ($32) or Write ($33) command
2803* ;
2804* ; Exit: D7.W - IORESULT
2805* ;
2806* ; All other registers are preserved.
2807* ;
2808* ;DsectIO MOVEM.L D0-D6/A0-A6,-(SP) ;save registers
2809* ; BSR FDgetadr ;set address registers
2810* ; ;A1 = ptr to device description info
2811* ; ;A2 = ptr to slot controller registers
2812* ; ;A3 = ptr to slot static RAM
2813* ;
2813* ; CLR.L D0 ;
2814* ; BSR FDlcmd1 ;turn on motor and setup controller
2815* ; BSR FDseek ;get to track
2816* ; BNE.S FDsio9 ;if error, return
2817* ; CMPI.W #DskWrit,D5 ;only do write if cmd is a write
2818* ; BNE.S FDsio2 ;else do a read
2819* ; BSR FDsecW ;
2820* ; BRA.S FDsio9 ;
2821* ;
2822* ;
1464 0C45 0033 2823* FDsectIO CMPI.W #DskWrit,D5 ;make sure cmd is a read cmd
1468 6604 2824* BNE.S FDsio1 ;it is
146A 6000 0378 2825* BRA FDEopod ;it isn't, return error
2826* ;
146E 48E7 FEFE 2827* FDsio1 MOVEM.L D0-D6/A0-A6,-(SP) ;save registers
1472 6100 014C 2828* BSR FDgetadr ;set address registers
2829* ; ;A1 = ptr to device description info
2830* ; ;A2 = ptr to slot controller registers
2831* ; ;A3 = ptr to slot static RAM
1476 4280 2832* CLR.L D0 ;
1478 6100 008A 2833* BSR FDlcmd1 ;turn on motor and setup controller
147C 6100 025C 2834* BSR FDseek ;get to track
1480 6604 2835* BNE.S FDsio9 ;if error, return
2836* ;
1482 6100 00D6 2837* FDsio2 BSR FDsecR ;read sector specified by D4.W
2838* ;
1486 6100 0046 2839* FDsio9 BSR FDMtrf ;turn off motor
148A 4CDF 7FFF 2840* MOVEM.L (SP)+,D0-D6/A0-A6 ;restore registers
148E 4A07 2841* TST.B D7 ;set return condition code
1490 4E75 2842* RTS ;return
2843* ;

```

```

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 CP511ram ,ADDRESS OF FIRST RAM FOR SLOTS
00000000 2851* SVLCMD equ 0 ,SAVE OF LOCAL COMMAND
2852* ;
00000200 2853* BLKSZ equ 512 ,05 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 SLOTSSEL
2860* ;
00030001 2861* NDEV1AD equ $30001 ,ADRS OF NDEV0 (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* NNM11 equ 0 ,BIT POSITION FOR EACH STATUS BIT
00000001 2870* NNM12 equ 1 ,
00000002 2871* NNM13 equ 2 ,
00000003 2872* NNM14 equ 3 ,
2873* ;
00000004 2874* NIRQ1 equ 4 ,
00000005 2875* NIRQ2 equ 5 ,
00000006 2876* NIRQ3 equ 6 ,
00000007 2877* NIRQ4 equ 7 ,
2878* ;

```

```

2880* ;
2881* ; Bytes per sector
2882* ;
00000080 2883* BPS81SD equ 128 ;Single density 8"
00000100 2884* BPS81DD equ 256 ;Double density 8"
00000140 2885* BPS51SD equ 256 ;Apple 5 1/4" floppy
2886* ;
2887* ; Sectors per track
2888* ;
0000001A 2889* SCPT8SD equ 26 ;Single density 8"
0000001A 2890* SCPT8DD equ 26 ;Double density 8"
00000010 2891* SCPT3SD equ 16 ;Apple 5 1/4" floppy
2892* ;
2893* ; Tracks per side
2894* ;
0000004D 2895* TKPS8SD equ 77 ;Single density 8"
0000004D 2896* TKPS8DD equ 77 ;Double density 8"
00000023 2897* TKPS3SD equ 35 ;Apple 5 1/4" floppy
2898* ;
2899* ; Number of blocks per disk
2900* ;
000001F4 2901* NBLK8SD equ 500 ;Single density 8" single sided
000003E9 2902* NBLK8DD equ 1001 ;Double density 8" single sided
00000118 2903* NBLK5SD equ 280 ;Apple 5 1/4" floppy
2904* ;
2905* ; Error return codes
2906* ;
00000000 2907* RGOOD equ 0 ;disk access successful
FFFFFFF7 2908* RBDBLK equ -1 ;Block requested is out of range
FFFFFFFE 2909* RBDUNT equ -2 ;bad unit number or driver not implemented
FFFFFFFD 2910* RBDOPCO equ -3 ;Requested unit I/O function is not valid
FFFFFFFC 2911* RHWRERR equ -4 ;Hardware error
FFFFFFFB 2912* RLOSTDEV equ -5 ;Lost device. i.e. device went offline
2913* ;
FFFFFFF0 2914* RWRPROT equ -16 ;the unit is write protected
FFFFFFEF 2915* RSEEKERR equ -17 ;SEEK Error
FFFFFFEE 2916* RBUSY equ -18 ;device busy
FFFFFFED 2917* RRNE equ -19 ;record not found - maybe disk is bad
FFFFFFEC 2918* RNOTRDY equ -20 ;device not ready
FFFFFFE0 2919* RERRUNOWN equ -64 ;error origin unknown
2920* ;

```

```

2922* ;
2923* ; This section contains the equates for Floppy Disk cont FD1793
2924* ; All references are w.r.t NDEV1AD in A2
2925* ; Prefix of 'L' means that this reference is to the 'LOCAL'
2926* ; logic on the disk controller board
2927* ; Prefix of FDC means that this referenc is to the Floppy disk controller
2928* ;
2929* ; Local disk controller board equates
2930* ;
00000000 2931* LSTRR equ 0 ;index to the local Status reg
00000000 2932* LCMDR equ 0 ;index to the local command reg
2933* ;
00000000 2934* LSDRO equ 0 ;BIT 0 =1 DRQ
00000001 2935* LSINT equ 1 ;INTERRUPT REQUEST
00000004 2936* LS1SD2SD equ 4 ;=0 if 2 sided , =1 if one sided
00000005 2937* LSRINMIN equ 5 ;=1 if 8 inch , =0 if min
00000006 2938* LSDSKCHG equ 6 ;=0 if disk changd, 1 if not
00000007 2939* LSPMHFM equ 7 ;=1 if sigi density =0 if double
2940* ;
2941* ; Command register equates
2942* ;
00000000 2943* LCFLPSD1 equ 0 ;=0 if side 0 , =1 if side 1
00000001 2944* LCDE0 equ 1 ;drive select bit 0
00000004 2945* LCDE1 equ 4 ;drive select bit 1
00000005 2946* LCMOTOROF equ 5 ;=1 if motor to be turned off
00000006 2947* LCBINMIN equ 6 ;
00000006 2948* LCFLP8IN equ 6 ;=1 to select 8 in, =0 for 5 1/4
00000007 2949* LCEMFHM equ 7 ;=1 to select singl density , 0 for dbl
2950* ;

```

```

2952* ;
2953* ; Floppy disk controllers equates
2954* ;
00000010 2955* FDCAD    equ  $10    ,Floppy disk controller base index
2956* ;
2957* ; address of the internal registers of FDC
2958* ;
00000010 2959* FDCCMDR   equ  FDCAD+0 ,ADRS OF FDC COMMAND REG
00000010 2960* FDCSTRR   equ  FDCAD+0 ,ADRS OF FDC STATUS REG
00000012 2961* FDCSTRKR  equ  FDCAD+2 ,ADRS OF FDC TRACK REG
00000014 2962* FDCSECR   equ  FDCAD+4 ,ADRS OF FDC SECTOR REG
00000016 2963* FDCDATR   equ  FDCAD+6 ,ADRS OF FDC DATA REG
2964* ;
2965* ; Command code equates
2966* ;
00000000 2967* CRESTORE  equ  0        ,0 0 0 0 H V R1 R0
00000010 2968* CSEEK    equ  $10     ,0 0 0 1 h w r1 r0
00000020 2969* CSTEP     equ  $20     ,0 0 1 U h v r1 r0
00000040 2970* CSTEPIN   equ  $40     ,0 1 0 U h v r1 r0
00000060 2971* CSTEPOUT  equ  $60     ,0 1 1 U h v r1 r0
2972* ;
2973* ; Type II commands
2974* ;
00000080 2975* CRDSEC    equ  $80     ,1 0 0 m F2 E F1 0
000000A0 2976* CWRSEC    equ  $A0     ,1 0 1 m F2 E F1 0
2977* ;
2978* ;Type III commands
2979* ;
000000C0 2980* CRDAM     equ  $C0     ,1 1 0 0 0 E 0 0
000000E0 2981* CRDTRK   equ  $E0     ,1 1 1 0 0 E 0 0
000000F0 2982* CWRTRK   equ  $F0     ,1 1 1 1 0 E 0 0
000000D0 2983* CFRCINT  equ  $D0     ,1 1 0 1 I1 I2 I3 I4
2984* ;
2985* ; FLAGS equates -- all flags have prefix of F
2986* ;
00000068 2987* FH1d     equ  $8        ,if =1 load head in the beginning
2988* ; ,if =0 unload head in beginning
00000004 2989* FVerify    equ  $4        ,if =1 verify destination trk else not
00000010 2990* FUpdttrk  equ  $10     ,if =1 update TRK reg after each STEP
00000000 2991* FSTPRT3ms  equ  $0        ,step rate = 3 milliseconds
00000001 2992* FSTPRT6ms  equ  $1        ,step rate = 6 milliseconds
00000002 2993* FSTPRT10ms equ  $2        ,step rate = 10 milliseconds
00000003 2994* FSTPRT15ms equ  $3        ,step rate = 15 milliseconds
2995* ;
00000010 2996* FMFS      equ  $10     ,M=1 if multiple sectors else =0
00000004 2997* FDLT      equ  $4        ,E=1 if internal dly of 15 ms =0 no dly
00000008 2998* FSDCPM   equ  $8        ,F2=0 compare with side 0,=1 with side 1
00000002 2999* FSDCPEN  equ  $2        ,E1=1 enable side compare, =0 disable cmp
3000* ;
00000001 3001* FINTRDY   equ  $1        ,not ready to ready
00000002 3002* FINTNRDY  equ  $2        ,ready to not ready
00000004 3003* FINTIDXP  equ  $4        ,interrupt on index pulse
00000008 3004* FINTIMM   equ  $8        ,terminate command immediately and intrpt
3005* ;

```

```

3007* ;
3008* ; Status register equates -- all status reg bits have prefix of 5
3009* ;
3010* ;name      bit position
3011* ;
00000000 3012* SBUSY   equ 0      ;S0 busy
3013* ;
00000001 3014* SINDEXT equ 1      ;S1 index pulse encountered
00000001 3015* SDRQ   equ 1      ;S1 data request
3016* ;
00000002 3017* STRK0   equ 2      ;S2 track 00
00000002 3018* SDTOVER equ 2      ;S2 data over run
00000002 3019* SDTUNDR equ 2      ;S2 data under run
3020* ;
00000003 3021* SRCERR  equ 3      ;S3 crc error
3022* ;
00000004 3023* SSEEKERR equ 4      ;S4 seek error
00000004 3024* SRNF   equ 4      ;S4 record not found
3025* ;
00000005 3026* SHDLDD  equ 5      ;S5 head loaded
00000005 3027* SRECTYP equ 5      ;S5 record type
00000005 3028* SWRFAULT equ 5      ;S5 write fault
3029* ;
00000006 3030* SWRPROT equ 6      ;S6 floppy write protected
3031* ;
00000007 3032* SNOTRDY equ 7      ;S7 floppy not ready
3033* ;

```



```

3035* ;
3036* ; FDI8sssd -- Set up constants for Corvus 8" single side single density
3037* ;
1492      3038* FDI8sssd
1492 6100 014C      3039*      bsr      FDgetadr      ;set address registers
3040* ;A1 = ptr to device description info
3041* ;A2 = ptr to slot controller registers
3042* ;A3 = ptr to slot static RAM
1494 337C 01F4 0734 3043*      move.w #NBLK8SD,CPfdvss(A1) ;set device size in blocks
149C 337C 0080 0736 3044*      move.w #BPS81SD,CPfbps(A1) ;set bytes per sector
14A2 137C 001A 0738 3045*      move.b #SCPT8SD,CPfspt(A1) ;set sectors per track
14A8 137C 004D 0739 3046*      move.b #TKPS8SD,CPftps(A1) ;set tracks per side
14AE 601C      3047*      bra.s  FDI8ss ;set other values and return
3048*
3049* ;
3050* ; FDI8ssdd -- Set up constants for Corvus 8" single side double density
3051* ;
14B0      3052* FDI8ssdd
14B0 6100 012E      3053*      bsr      FDgetadr      ;set address registers
3054* ;A1 = ptr to device description info
3055* ;A2 = ptr to slot controller registers
3056* ;A3 = ptr to slot static RAM
14B4 337C 03E9 0734 3057*      move.w #NBLK8DD,CPfdvss(A1) ;set device size in blocks
14BA 337C 0100 0736 3058*      move.w #BPS81DD,CPfbps(A1) ;set bytes per sector
14C0 137C 001A 0738 3059*      move.b #SCPT8DD,CPfspt(A1) ;set sectors per track
14C4 137C 004D 0739 3060*      move.b #TKPS8DD,CPftps(A1) ;set tracks per side
14CC 137C 0001 073A 3061* FDI8ss move.b #1,CFispd(A1) ;set sides per disk
14D2 137C 0001 073B 3062*      move.b #1,CFifst(A1) ;set first track offset
14D8 4E75      3063*      rts ;return
3064*
3065* ;
3066* ; FDinit -- Initialize Corvus floppy disk drive
3067* ;
14DA 6100 0104      3068* FDIinit BSR      FDgetadr      ;set address registers
3069* ;A1 = ptr to device description info
3070* ;A2 = ptr to slot controller registers
3071* ;A3 = ptr to slot static RAM
14DE 4280      3072*      CLR.L  D0 ;
14E0 7201      3073*      MOVEQ #1,D1 ;A FAKE SECTOR LENGTH
14E2 6120      3074*      BSR.S  FDlcmd1 ;turn on motor and setup controller
14E4 6100 0230      3075*      BSR      FDrst ;restore to track 0
14E8 6104      3076*      BSR.S  FDMtrf ;turn motor off
14EA 4A07      3077*      TST.B  D7 ;set return condition code
14EC 4E75      3078*      RTS ;return
3079*

```

```

3081* ;
3082* ; D0.W -- FREE
3083* ; D1.W -- FREE
3084* ; D2.W -- BYTE CNT
3085* ; D3.W -- BASE BLK ADRS
3086* ; D4.W -- FREE
3087* ; D5.W -- USER CMD
3088* ; D6.W -- FREE
3089* ;
3090* ; A0.L -- USER BUFFER ADDRESS
3091* ; A1.L -- DEVICE DESCRIPTION AREA BASE ADDRESS
3092* ; A2.L -- FLOPPY CONTROLLER BASE ADDRESS
3093* ; A3.L -- STATIC RAM BASE ADDRESS
3094* ;
3095* ;
3096* ; FDMtrf -- Turn motor off
3097* ;
14EE 182E 0000 3098* FDMtrf move.b SVLCMD(A3),D4 ;get current local command
14F2 08C4 0005 3099* bset #LCMOTOROF,D4 ;set motor off flag
14F6 1544 0000 3100* move.b D4,LCMDR(A2) ;move command to command register
14FA 1744 0000 3101* move.b D4,SVLCMD(A3) ;save current local command
14FE 4E75 3102* rts ;return
3103*
3104* ;
3105* ; FDIcmd -- GET THE LOCAL COMMAND FOR THIS DRIVE INTO REGISTER D0.8
3106* ; RETURNS WITH D7 CLEAR LONG
3107* ;
1500 6100 010A 3108* FDIcmd bsr FDClcTS ;Calc first side trk sec
3109* ;
3110* ; form a local command in D0
3111* ; Entry used for Read/Write a sector
3112* ;
1504 0880 0005 3113* FDIcmdi beqr #LCMOTOROF,D0 ;clear motor off bit
1508 08C0 0006 3114* bset #LCFLP8IN,d0 ;indicate 8 inch flp
150C 08C0 0007 3115* bset #LCFMFM8,d0 ;indicate 8 inch flp
1510 1E29 0708 3116* move.b CPosdrr(a1),d7 ;sel drive
1514 E25F 3117* ror.w #1,d7 ;note DE0 is B0
1516 E50F 3118* lsl.b #2,d7 ;DE1 is B4
1518 E55F 3119* rol.w #2,d7 ;
151A 8007 3120* or.b d7,d0 ;set into d0
151C 1740 0000 3121* move.b d0,SVLCMD(a3) ;save then command
1520 1540 0000 3122* move.b d0,LCMDR(a2) ;set local command register
1524 4287 3123* clr.l d7 ;clear error register
1526 4E75 3124* rts ;return
3125*

```

```

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 FDEb1ck ;jump if first blk rqstd is invalid
1532 8669 0734 3133* cmp.w CPIdvex(A1),d3 ;is it in limit
1536 6C00 02A0 3134* bge FDEb1ck ;jump final block exceeds max
153A 61C4 3135* BSR.S FD1cmd ;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/WITES 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* FDrdwrl sub.w d1,d2 ;
1544 680E 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 FDrdwrl ;process next sector
3167* ;
1554 6100 FF98 3168* FDrdwr9 bsr FDMtrf ;turn off motor
1558 4E75 3169* rts ;return
3170*

```

```

3172* ;
3173* ; FDsecRW -- Read or write a sector of data to the floppy
3174* ;         It transfer the data to/from the adrs in A0
3175* ;         from / to the floppy
3176* ;
3177* ;         Enter: DS - DskRead or DskWrit
3178* ;
155A 3179* FDsecRW                                ;fall thru to FDsecR
3180* ;         cmp.w #DskWrit,DS                 ;see if it is a Unit write
3181* ;         beq.s  FDsecRW1
3182* ;         bsr.s  FDsecR
3183* ;         bra.s  FDsecRW9
3184* ;FDsecRW1 bsr.s  FDsecW
3185* ;FDsecRW9 rts
3186* ;
3187* ;
3188* ; FDsecR -- Read one sector of data
3189* ;
3190* ;         Enter: bytes per sec ---) D1.w
3191* ;         sector adrs ----) D4.w
3192* ;         buffer adrs ----) A0
3193* ;         floppy must be positioned on desired track
3194* ;
3195* ;         Exit: OS result code ---) D7
3196* ;         data to the adrs pointed by A0
3197* ;
          00000004 3198* FDrcRd equ 4 ;read sector retry count
          00000004 3199* FDrcDOr equ 4 ;data overrun retry count
3200* ;
155A 48E7 0400 3201* FDsecR movem.l d5-d6,--(SP) ;save
155E 3A3C 0004 3202* move.w #FDrcRd,d5 ;get read sector retry count
1562 2848 ; 3203* move.l a0,a4 ;save user buf ptr
          ; 3204* ;
1564 3C3C 0004 3205* FDsecR1 move.w #FDrcDOr,d6 ;get data overrun retry count
          ; 3206* ;
1568 204C 3207* FDsecR2 move.l a4,a0 ;get user buf adrs
156A 6100 0144 3208* BSR FDwRdy ;WAIT FOR READY OR TIMED OUT
156E 661C 3209* BNE.S FDsecR9 ;TIMED OUT ERR
1570 6100 0022 3210* BSR FDccRd ;
1574 0807 0002 3211* b1st #SDTOVER,d7 ;is ther data overrun
1578 57CE FFE2 3212* dbeg d6,FDsecR2 ;data over run, try again.
157C 0807 0003 3213* b1st #SCRCERR,d7 ;is ther crc error
1580 57CD FFE2 3214* DBEQ d5,FDsecR1 ; DO UNTIL (no crc error)
          ; 3215* ; or (no more retries left)
          ; 3216* ;
1584 6100 0206 3217* FDsecR3 BSR FRdSta ;check read status
1588 6702 3218* beq.s FDsecR9 ;
158A 204C 3219* move.l a4,a0 ;
          ; 3220* ;
158C 4CDF 0060 3221* FDsecR9 movem.l (sp)+,d5-d6 ;
1590 4A07 3222* tst.b d7 ;d7 contains result code
1592 4E75 3223* rts ;return
          ; 3224* ;

```

```

3226* ,
3227* , FDsecW -- Write one sector of data
3228* ;
3229* ; Enter: bytes per sec ---> D1.w
3230* ; sector adrs -----> D4.w
3231* ; buffer adrs -----> A0
3232* ; floppy must be positioned on desired track
3233* ,
3234* ; Exit: OS result code ---> D7
3235* ; data to the adrs pointed by A0
3236* ;
3237* ,FDrcWr equ 4 ;write sector retry count
3238* ,FDrcDOW equ 4 ;data overrun retry count
3239* ;
3240* ,FDsecW move.l d5-d6, -(SP) ;save registers
3241* ; move.w #FDrcWr, d5 ;get write retry count
3242* ; move.l a0, a4 ;save user buf ptr
3243* ;
3244* ,FDsecW1 move.w #FDrcDOW, d6 ;get data overrun retry count
3245* ;
3246* ,FDsecW2 move.l a4, a0 ;get user buf adrs
3247* ; BSR FDWrDy ;WAIT FOR READY OR TIMED OUT
3248* ; BNE.S FDsecW9 ;TIMED OUT ERR
3249* ; BSR FDccWr ;
3250* ; b1st %SDTOVER, d7 ;data overrun?
3251* ; dbeq d6, FDsecW2 ;yes, try again
3252* ; b1st %SCRCERR, d7 ;CRC error?
3253* ; dbeq d5, FDsecW1 ;yes, try again
3254* ;
3255* ; BSR FDWrSta ;check write status
3256* ; beq.S FDsecW9 ;if no error, return
3257* ; move.l a4, a0 ;
3258* ;
3259* ,FDsecW9 move.l (sp)+, d5-d6 ;restore registers
3260* ; tst.b d7 ;d7 contains result code
3261* ; rts ;return
3262*

```

```

3264* ; ***** TYPE II COMMANDS *****
3265*
3266* ;
3267* ; FDCcRd -- Read one sector of data
3268* ;
3269* ;      Enter: bytes per sec ---) D1.w
3270* ;      sector adrs ----) D4.w
3271* ;      buffer adrs ----) A0
3272* ;      floppy must be positioned on desired track
3273* ;
3274* ;      Exit: status ----) D7
3275* ;      data to the adrs pointed by A0
3276* ;
1594 3F01      3277* FDCcRd  move.w d1,-(SP)          ;save
1596 5341      3278*      subq.w #1,d1              ;byte count
1598 40E7      3279*      move.w sr,-(SP)          ;
159A 007C 0700 3280*      ori.w #50700,sr          ;disable interrupts
159E 1544 0014 3281*      move.b d4,FDCSECR(A2)      ;
15A2 157C 0080 0010 3282*      move.b #CRDSEC,FDCCMDR(A2)    ;issue command
15A8 3E3C 0019 3283*      move.w #25,d7          ;wait at least 28 micro-second
15AC 51CF FFEF 3284* FDCcRd1 dbf    d7,FDCcRd1      ;
3285* ;
15B0 0B2A 0000 0010 3286* FDCcRd2 btst   #SBUSY,FDCSTRR(A2) ;see if the ctrl is busy
15B6 67F8      3287*      beq.s FDCcRd1          ;jump if not busy
3288* ;
3289* ; ***** TIME CRITICAL LOOP
3290* ;
15B8 1E2A 0000 3291* FDCcRd3 move.b LSTRR(A2),D7      ;read status
15BC 0807 0000 3292*      btst   #LSDRQ,d7          ;is DRQ there
15C0 6608      3293*      bne.s FDCcRd5          ;yes , jump
3294* ;
15C2 0807 0001 3295* FDCcRd4 btst   #LSINT,D7          ;is FDC done
15C6 67F0      3296*      beq.s FDCcRd3          ;no, jump
15C8 600C      3297*      bra.s FDCcRd6          ;ctrl terminated too soon
3298* ;
15CA 10EA 0016 3299* FDCcRd5 move.b FDCDATR(A2),(a0)+ ;get a byte from FDC
15CE 51C9 FFE8 3300*      dbf    d1,FDCcRd3          ;read the remaining bytes
3301* ;
3302* ; a complete sector has been read.
3303* ;
15D2 6100 0194 3304*      bsr    FDCcRd5          ;
3305* ;
15D6 1E2A 0010 3306* FDCcRd6 move.b FDCSTRR(A2),D7      ;read the status
15DA 46DF      3307*      move.w (sp)+,sr          ;restore SR
15DC 321F      3308*      move.w (SP)+,d1          ;
15DE 4E75      3309*      rts
3310*

```

```

3312*
3313* , FDccWr -- Write one sector of data
3314*
3315*      Enter: bytes per sec ---) D1.w
3316*      sector adrs ----) D4.w
3317*      buffer adrs ----) A0
3318*      floppy must be poitioned on desired track
3319*
3320*      Exit: status ----) D7
3321*      data to the adrs pointed by A0
3322*
3323* ,FDccWr  move.w d1,-(SP)          ;save
3324*      subq.w #1,d1                ;byte count
3325*      move.w sr,-(SP)             ;
3326*      ori.w #0700,sr              ;disable interrupts
3327*      BSR.S FDccWr1               ;CALL time critical FDccWr PART
3328*      bsr FDnRdy                  ;
3329*      move.b FDCSTRR(A2),d7        ;read the status
3330*      move.w (sp)+,sr              ;restore SR
3331*      move.w (SP)+,d1              ;
3332*      rts                          ;return
3333*      ;
3334*      ;
3335* ,FDccWr1 move.b d4,FDCSECR(A2)    ;
3336*      move.b #CWRSEC,FDCMDR(A2)    ; issue command
3337*      ;
3338*      move.w #25,d7                ;wait at least 28 micro-second
3339* ,FDccWr2 dbf d7,FDCcWr2          ;
3340*      ;
3341* ,FDccWr3 btst #SBUST,FDCSTRR(A2)  ;
3342*      beq.s FDccWr3                ;
3343*      ;
3344*      ; ***** TIME CRITICAL LOOP
3345*      ;
3346* ,FDccWr4 move.b LSTRR(A2),D7        ; FDCSTRR(A2),d7 ;read status
3347*      btst #LSDRG,d7                ;is DRG there
3348*      bne.s FDccWr6                 ;yes , jump
3349*      ;
3350* ,FDccWr5 btst #LSINT,D7            ;is FDC done
3351*      beq.s FDccWr4                 ;no , jump
3352*      rts                            ;terminated too soon
3353*      ;
3354* ,FDccWr6 move.b (a0)+,FDCDATR(A2)  ;move a byte to FDC
3355*      dbf d1,FDCcWr4                ;write the reaminging bytes
3356*      RTS                            ;return
3357*

```

```

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 (CPossslot)
3366* ;
15E0 3367* FDgetadr
15E0 227C 0000 0000 3368* move.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 CPossslot(a1),d7 ;*
15F0 4887 3371* ext.w d7 ;*
15F2 CEFC 0020 3372* mulu.w #DEVADOEST,d7 ;*
15F6 D5C7 3373* adda.l d7,a2 ;*
15F8 47E8 0900 3374* lea CPsllram.w,A3 ;get pointer to static RAM
15FC 1E29 0706 3375* move.b CPossslot(a1),d7 ;*
1600 4887 3376* ext.w d7 ;*
1602 5347 3377* subq.w #1,d7 ;*
1604 CEFC 0100 3378* mulu #9100,d7 ;*
1608 D7C7 3379* adda.l d7,A3 ;*
160A 4E75 3380* rts ;return
3381*

```



```

3383* ;
3384* ; FDclcTS -- calculate the Side, Track address and sector
3385* ; address for the first block requested by the user
3386* ;
3387* ; Enter: A1 - device table address
3388* ; D3 - block address
3389* ;
3390* ; Exit: D0.bit - side flag
3391* ; D3.w - track address
3392* ; D4.w - sector address
3393* ;
160C 4280 3394* FDclcTS clr.l d0 ;
160E 3229 0736 3395* move.w CPfbps(A1),D1 ;get bytes per sector
1612 48C3 3396* ext.l d3 ;clear the upper 16 bits of d3
1614 2E3C 0000 0200 3397* move.l #BLKSZ,d7 ;
161A 8EC1 3398* divu d1,d7 ;
161C C6C7 3399* mulu d7,d3 ;absolute sector adrs to d3
161E 2E03 3400* move.l d3,d7 ;
1620 4244 3401* clr.w d4 ;make sure that upper byte is 00
1622 1829 0738 3402* move.b CPfspt(A1),d4 ;
1626 8EC4 3403* divu d4,d7 ;
1628 3607 3404* move.w d7,d3 ;absolute track adrs to d3
162A 4847 3405* swap d7 ;
162C 1829 073B 3406* move.b CPfofst(a1),d4 ;get first sector offset
1630 4884 3407* ext.w d4 ;
1632 D847 3408* add.w d7,d4 ;get sector address
3409* ; ---- bclr #LCFLPSD1,d0 ;select side 0 (already 0)
1634 1E29 0739 3410* move.b CPftps(A1),d7 ;get tracks per side
1638 4887 3411* ext.w d7 ;
163A B647 3412* cmp.w d7,d3 ;is track on side 0?
163C 6D16 3413* bit.s FDclcT9 ;yes, return
163E 0C29 0001 073A 3414* cmpi.b #1,CPfspd(a1) ;is there a side 1?
1644 470A 3415* beq.s FDclcT8 ;no, report error
1646 08C0 0000 3416* bset #LCFLPSD1,D0 ;select side 1
164A 9647 3417* sub.w d7,d3 ;update track address
164C B647 3418* cmp.w d7,d3 ;is track on side 1?
164E 6DD4 3419* bit.s FDclcT9 ;yes, return
3420* ;
1650 6000 0186 3421* FDclcT8 bra FDEbick ;indicate block number error
3422* ;
1654 4247 3423* FDclcT9 clr.w d7 ;indicate no error
1656 4E75 3424* rts ;return
3425* ;

```

```

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 0738 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 1029 0738 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 FDincTS ;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 6624 3456* bne.s FDincT9 ;return if timeout error
1686 6100 00BC 3457* bsr FDocSin ;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 #LCFLP5D1,D0 ;Select side 1
1694 1540 0000 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* FDincTS cmp.b d7,d3 ;are we past last track on side?
16A2 6C00 0134 3470* bge FDEbick ;yes, report error
16A6 5243 3471* addq.w #1,d3 ;increment track number
16A8 6100 071E 3472* bsr ADocSin ;step in 1 track
3473* ;
16AC 4A47 3474* FDincT9 tst.w d7 ;set return condition codes
16AE 4E75 3475* rts ;return
3476* ;

```

```

3478* ;
3479* ; FDWRdy -- WAIT UNTIL FDC SAYS DRIVE IS READY OR
3480* ; TIME OUT (NOT MORE THAN 1 SECOND)
3481* ;
3482* ; Exit: NE = timed out D7 has error result
3483* ; EQ = ready (D7 = 0)
3484* ;
00000004 * 3485* FDtmoHi equ 4 ; SHOULD BE AT LEAST 1 SECOND
00007FFF 3486* FDtmoLo equ $7FFF ; *
3487* ;
16B0 4287 3488* FDWRdy CLR.L D7 ;
16B1 48A7 0600 3489* MOVEM.W D5-D6,-(SP) ;SAVE D6 AND D5
16B6 3A3C 7FFF 3490* MOVE.W #FDtmoLo,D5 ;
16BA 3C3C 0004 3491* MOVE.W #FDtmoHi,D6 ;
3492* ;
16BE 082A 0007 0010 3493* FDWRdy1 BTST #SNOTRDY,FDCSTRR(A2) ;IS FLOPPY READY
16C4 57CD FFF8 3494* DBEQ D5,FDWRdy1 ;DO UNTIL (FLOPPY READY) OR (TIME OUT)
16C8 57CE FFF4 3495* DBEQ D6,FDWRdy1 ;
16CC 6704 3496* BEQ.S FDWRdy? ;DIDN'T TIME OUT
16CE 3E3C FFEC 3497* MOVE.W #RNOTRDY,D7 ;
3498* ;
16D2 4C9F 0040 3499* FDWRdy? MOVEM.W (SP)+,D5-D6 ;
16D4 4A47 3500* TST.W D7 ;SET CONDITION CODES - NE MEANS ERROR
16D8 4E75 3501* RTS ;return
3502* ;

```

```

3504* ;
3505* ; FDseek --
3506* ;
3507* ; It is assumed that TRACK REG contains the number of the track
3508* ; of the current position of the read write Head.
3509* ;
3510* ; Enter: D3.W - Seek track address
3511* ;
00000004
3512* FDrcSk equ 4 ;seek retry count
3513* ;
16DA 61D4 3514* FDseek BSR.S FDwRdy ;WAIT FOR READY OR TIMED OUT
16DC 642A 3515* BNE.S FDseek9 ;TIMED OUT ERR
16DE 6100 0074 3516* BSR FDccSk ;
16E2 6100 00D2 3517* BSR FDskSta ;check seek status
16E6 6720 3518* BEQ.S FDseek9 ;there is NO error
16E8 3F05 3519* move.w d5,-(sp) ;save d5
16EA 3A3C 0004 3520* move.w #FDrcSk,D5 ;
3521* ;
16EE 61C0 3522* FDseek1 BSR.S FDwRdy ;WAIT FOR READY OR TIMED OUT
16F0 6414 3523* BNE.S FDseek8 ;TIMED OUT ERR
16F2 6140 3524* BSR.S FDccRst ;
16F4 61BA 3525* BSR.S FDwRdy ;WAIT FOR READY OR TIMED OUT
16F6 660E 3526* BNE.S FDseek8 ;TIMED OUT ERR
16F8 615A 3527* BSR.S FDccSt ;
16FA 0807 0004 3528* btst #sseekerr,d7 ;
16FE 57CD FFEE 3529* DBEQ d5,FDseek1 ;try until no seek error or
3530* ;no more retries
1702 6100 00B2 3531* BSR FDskSta ;check seek status
3532* ;
1706 3A1F 3533* FDseek0 move.w (sp)+,d5 ;
3534* ;
1708 4A07 3535* FDseek9 TST.B D7 ;set return condition code
170A 4E75 3536* RTS ;return
3537* ;

```

```

3539* ;
3540* ; FDrstW -- Restore the floppy to track 0
3541* ; It exits when it has successfully restored the floppy to
3542* ; track 0 or when the retry count has exhausted.
3543* ; Then it calls the FDswSta routine to analyse status.
3544* ; Note W in FDswSta. It looks at WRprot bit of FDCSTRR.
3545* ;
170C 48E7 0202 3546* FDrstW movem.l a6/d6,-(sp) ;
1710 4DFA 0098+ 3547* lea FDswSta,a6 ;
1714 6008 3548* bra.s FDrst0 ;
3549* ;
3550* ;
3551* ; FDrst -- same as FDrstW except no W there.
3552* ; It does not look at Write protect status bit in FDCSTRR.
3553* ;
1716 48E7 0202 3554* FDrst movem.l a6/d6,-(sp) ;
171A 4DFA 009A+ 3555* lea FdskSta,a6 ;
3556* ;
171E 3C3C 0004 3557* FDrst0 move.w #4,d6 ;
3558* ;
1722 618C 3559* FDrst1 BSR.S FDWRdy ;WAIT FOR READY OR TIMED OUT
1724 6608 3560* BNE.S FDrst2 ;TIMED OUT ERR
1726 610C 3561* bsr.s FDocRst ;
1728 4E96 3562* jsr (a6) ;
172A 57CE FFF6 3563* dbeq d6,FDrst1 ;do until (successful) or (tried enough)
3564* ;
172E 4CDF 4040 3565* FDrst2 movem.l (sp)+,a6/d6 ;
1731 4E75 3566* rts ;return
3567* ;

```

```

3569* ;
3570* ; ***** TYPE I COMMANDS *****
3571* ;
00000007 3572* FDCrst equ CRESTORE+FSTPRT1&ms+VERIFY
00000035 3573* FDCstp equ CSTEP+FSTPRT6&ms+VERIFY+FUpdttrk
00000055 3574* FDCstpIn equ CSTEPIN+FSTPRT6&ms+VERIFY+FUpdttrk
00000075 3575* FDCstpOt equ CSTEPOUT+FSTPRT6&ms+VERIFY+FUpdttrk
00000015 3576* FDCseek equ CSEEK+FSTPRT6&ms+VERIFY
3577* ;
3578* ; FDecRst -- bring the floppy back to track 00
3579* ; Then set the FDCTRKR = 0
3580* ; Stepping pulses are given at the rate specified in cmd
3581* ;
1734 157C 0007 0010 3582* FDecRst move.b #FDCrst,FDCCMDR(A2) ;issue command
173A 602C 3583* bra.s FDrRdy ;wait for not ready
3584* ;
3585* ; FDecStp -- FDCTRKR+/-1 --> FDCTRKR
3586* ;
173C 157C 0035 0010 3587* FDecStp move.b #FDCstp,FDCCMDR(A2) ;issue command
1742 6024 3588* bra.s FDrRdy ;wait for not ready
3589* ;
3590* ; FDecSin -- FDCTRKR+1 --> FDCTRKR
3591* ;
1744 157C 0055 0010 3592* FDecSin move.b #FDCstpIn,FDCCMDR(A2) ;issue command
174A 601C 3593* bra.s FDrRdy ;wait for not ready
3594* ;
3595* ; FDecSot -- FDCTRKR-1 --> FDCTRKR
3596* ;
174C 157C 0075 0010 3597* FDecSot move.b #FDCstpOt,FDCCMDR(A2) ;issue command
1752 6014 3598* bra.s FDrRdy ;wait for not ready
3599* ;
3600* ; FDecSk -- issue a seek command. If there is a seek error,
3601* ; flip the density flag in D0 and try again
3602* ;
1754 3F05 3603* FDecSk move.w d5,-(sp) ;save d5
1756 1543 0016 3604* move.b D3,FDCCATR(A2) ;load the desired TRACK adrs
175A 7A02 3605* moveq #2,d5 ;
3606* ;
175C 157C 0015 0010 3607* FDecSk1 move.b #FDCseek,FDCCMDR(A2) ;issue command
1762 6104 3608* bsr.s FDrRdy ;wait for not ready
3609* ; ---- btsl #sseekerr,d7 ;
3610* ; ---- beq FDecSk2 ;no seek error
3611* ; ---- bchg #LSFMMEM,d7 ;flip the density bit
3612* ; ---- dbf d5,FDecSk1 ; $$$ WE MAY HAVE TO FLIP $$$
3613* ;
1764 3A1F 3614* FDecSk2 move.w (sp)+,d5 ;get back d5
1766 4E75 3615* rts ;return
3616* ;

```

```

3618* ;
3619* , FDrDy -- WAIT UNTIL FDC SAYS DRIVE IS NOT BUSY OR TIME OUT
3620* ;
3621* ; Exit: D7 = controller status register (FDCSTRR)
3622* ;
1768 48A7 6600 3623* FDrDy movem.w d5-d6,-(SP) ;
176C 3C3C 7FFF 3624* move.w #FDtmoLo,d6 ;
1770 3A3C 0002 3625* move.w #2,d5 , add 3*48000 iterations
3626* ;
1774 082A 0001 0000 3627* FDrDy1 btsl #LSINT,LSTRR(a2) ;is it busy
177A 56CE FFF8 3628* dbne d6,FDrDy1 ;DO UNTIL (not busy) or (no more retries)
177E 56CD FFF4 3629* dbne d5,FDrDy1 ;DO UNTIL (not busy) or (no more retries)
3630* ;
1783 4C9F 0060 3631* FDrDy2 movem.w (SP)+,d5-d6 ;
1786 1E2A 0010 3632* MOVE.B FDCSTRR(A2),D7 ;GET CONTROLLER STATUS
178A 4E75 3633* RTS ;return
3634* ;

```

```

3636* ;
3637* ; Check status subroutines
3638* ;
3639* ; Exit: D7 - IORESULT code
3640* ;
178C 1E2A 0010 3641* FDrdsSta move.b FDCSTRR(A2),d7 ;read the status
3642* ; bra.s FDrwSta ;
3643* ;
3644* ; FDrwSta move.b FDCSTRR(A2),d7 ;read the status
3645* ; btst #SWRPROT,d7 ;
3646* ; bon.s FDEprot ;
3647* ; btst #SWRFAULT,d7 ;
3648* ; bon.s FDEherr ;
3649* ;
1790 0807 0003 3650* FDrwSta btst #SCRCERR,d7 ;
1794 6642 3651* bon.s FDEorc ;
1796 0807 0004 3652* btst #SRNF,d7 ;
179A 6660 3653* bon.s FDErnf ;
3654* ;
179C 0807 0000 3655* FDrwSt1 btst #SBUSY,d7 ;
17A0 6660 3656* bon.s FDEbusy ;
17A2 0807 0007 3657* btst #SNOTRDY,d7 ;
17A6 6666 3658* bon.s FDEnrdy ;
17A8 6028 3659* bra.s FDokSta ;no error, return
3660* ;
17AA 1E2A 0010 3661* FDswSta move.b FDCSTRR(a2),d7 ;
17AE 0807 0006 3662* btst #SVRPROT,d7 ;
17B2 663C 3663* bon.s FDEprot ;
17B4 6004 3664* bra.s FDskSt1 ;
3665* ;
17B6 1E2A 0010 3666* FDskSta move.b FDCSTRR(a2),d7 ;
3667* ;
17BA 0807 0004 3668* FDskSt1 btst #SSEEKERR,d7 ;seek error?
17BE 6636 3669* bon.s FDEseek ;SEEK ERROR IN RSLT CODE
17C0 0807 0003 3670* btst #SCRCERR,d7 ;
17C4 6612 3671* bon.s FDEorc ;
17C6 0807 0000 3672* btst #sbusy,d7 ;
17CA 6636 3673* bon.s FDEbusy ;HARDWARE ERROR
17CC 0807 0007 3674* btst #SNOTRDY,d7 ;
17D0 663C 3675* bon.s FDEnrdy ;
3676* ;
17D2 4247 3677* FDokSta clr.w d7 ;indicate no error
3678* ;
17D4 4A47 3679* FDerSta tst.w d7 ;set return condition code
17D6 4E75 3680* rts ;return
3681* ;

```



17D8			3683*	FDEcrc		,error -- CRC	
17D8	3E3C	FFFF	3684*	FDEbick	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	#RHWERR,d7	,error -- hardware
17EE	60E4		3694*		bra.s	FDerSta	,set condition code and return
			3695*				,
17F0	3E3C	FFFF	3696*	FDEprot	move.w	#RWPROT,d7	,error -- write protect
17F4	60DE		3697*		bra.s	FDerSta	,set condition code and return
			3698*				,
17F6	3E3C	FFEF	3699*	FDEseek	move.w	#RSEKERR,d7	,error -- seek
17FA	60D6		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	0310	FDEbusy	move.b	#CFRCINT*FINTIMM,FDCCHDR(A2)	
1808	3E3C	FFEE	3706*		move.w	#RBSY,D7	,error -- busy
180C	60C6		3707*		bra.s	FDerSta	,set condition code and return
			3708*				,
180E	3E3C	FFEC	3709*	FDErny	move.w	#RNOTRDY,d7	,error -- not ready
1812	60C0		3710*		bra.s	FDerSta	,set condition code and return
			3711*				,

```

3713*      include 'CC.PROM.AD'      ,Apple floppy driver
3714*      ;
3715*      , File: CC.PROM.AD.TEXT
3716*      , Date: 03-Sep-82
3717*      , By:   Ravi Luthra
3718*      ,      Keith Ball
3719*      ;
3720*
3721*      ;
3722*      ,      Aboot -- Apple floppy disk boot processing
3723*      ;
1814 11C0 0700 3724* Aboot  move.b d0,CPbtslot.w      ,set boot slot number
1818 11C0 0706 3725*      move.b d0,CPosslot.w      ,set OS slot number
3726*      ; ---- cir.b CPbtsrvr.w      ,set boot server number (already 0)
181C 487A 0038+ 3727*      pea ADbkio      ,set boot disk blk i/o subr pointer
1820 21DF 0714 3728*      move.l (sp)+,CPbkio.w      ,*
1824 487A 0058+ 3729*      pea ADsecio      ,set boot disk sector i/o subr pointer
1828 21DF 0718 3730*      move.l (sp)+,CPdskio.w      ,*
3731*      ; ---- moveq #0,d0      , (already 0)
3732*      ; ---- move.b d0,CPossrvr.w      ,set OS server number (already 0)
3733*      ; ---- move.w d0,CPosblk+l.w      ,set OS volume block number (already 0)
3734*      ; ---- move.b d0,CPosdrv.w      ,set OS volume drive number (already 0)
182C 6100 0280 3735*      bsr ADISsssd      ,set up floppy constants
1830 6100 02C2 3736*      bsr ADinit      ,initialise floppy drive
1834 6D1E      3737*      bit.s Aboot90      ,just return if error
3738*      ;
1836 207C 0008 E000 3739* Abootl  move.l #USRbase,a0      ,get block buffer pointer
183C 7000      3740*      moveq #0,d0      ;
183E 3200      3741*      move.w d0,d1      ;
1840 7A32      3742*      moveq #DskRead,d5      ,get read block function code
3743*      ;
1842 6112      3744*      bsr.s ADbkio      ,read block 1 of boot code
1844 6D0E      3745*      bit.s Aboot90      ,just return if error
1846 610E      3746*      bsr.s ADbkio      ,read block 2 of boot code
1848 6D0A      3747*      bit.s Aboot90      ,just return if error
184A 207C 0008 E000 3748*      move.l #USRbase,a0      ,get block buffer pointer
1850 D0FC 800C      3749*      adda.w #12,a0      ,get pointer to boot code
3750*      ;
1854 4E75      3751* Aboot90 rts      ,return
3752*

```

```

3754* ;
3755* ; PHILOSOPHY: The user views floppy as a set of 512 byte blocks.
3756* ; The driver then translates this block to track address, sector
3757* ; address, side.
3758* ; It then makes the necessary number of request to read sectors.
3759* ; Partial sectors are not read or written, the excess is ignored.
3760* ; Sector length of an Apple floppy is 256 bytes.
3761* ;
3762* ; RESTRICTION: Bytes per sector must be exact divisor of 512 (block size).
3763* ; The block address must be less than (2**15)/bytes per sector,
3764* ; so that when sector is formed, it fits in the D3.W.
3765* ;
3766* ;
3767* ;
3768* ; ADbioKIO - Read/Write an Apple floppy disk block subroutine
3769* ;
3770* ; Enter: A0.L - Buffer address
3771* ;         D0.W - Block number
3772* ;         D1.W - Drive number
3773* ;         D5.W - Read (532) or Write (533) command
3774* ;
3775* ; Exit:  A0.L - Next free location in buffer
3776* ;         D0.W - Updated block number
3777* ;         D7.W - IORESULT
3778* ;
3779* ; All other registers are preserved.
3780* ;
1856 48E7 FE7E 3781* ADbioKIO MOVEM.L D0-D6/A1-A6,-(sp) ;save registers
185A 343C 0200 3782* MOVE.W #BLKSZ,D2 ;BLOCK SIZE IN BYTES
185E 3600 3783* MOVE.W D0,D3 ;
1860 0C45 0033 3784* CMPI.W #DskWrit,D5 ;
1864 6606 3785* BNE.S ADbioI ;
1866 6100 FF7C 3786* BSR FDEopcd ;only do reads
186A 6006 3787* BRA.S ADbio9 ;return
3788* ;
186C 6100 02A2 3789* ADbioI BSR ADrdwr ;
1870 6100 0292 3790* BSR ADmtrof ;TURN THE MOTOR OFF
3791* ;
1874 4CDF 7E7F 3792* ADbio9 MOVEM.L (sp)+,D0-D6/A1-A6 ;
1878 5240 3793* ADDQ.W #1,D0 ;INC BASE BLOCK
187A 4A47 3794* TST.W D7 ;SET CONDITION CODES *KB 8/24/82*
187C 4E75 3795* RTS ;return
3796* ;

```

```

3798* ;
3799* ; ADscfIO - Read/Write an Apple floppy disk sector
3800* ;
3801* ; Enter: A0.L - Buffer address
3802* ;         D1.W - Bytes per sector (256 for Apple)
3803* ;         D3.W - Track number (range: 0-35)
3804* ;         D4.W - Sector number (range: 0-15)
3805* ;         D5.W - Read ($32) or Write ($33) command
3806* ;
3807* ; Exit: D7.W - IORESULT
3808* ;
3809* ; All other registers are preserved.
3810* ;
187E 0C45 0033 3811* ADscfIO CHPI.W #DskWrit,DS ;make sure cmd is a read cmd
1882 6604 3812* BNE.S ADscfio ;it is
1884 6000 FF5E 3813* BRA FDEopcd ;it isn't, return error
3814*
1888 48E7 FEEF 3815* ADscfio MOVEM.L D0-D6/A0-A6,-(sp) ;save registers
188C 6100 FDS2 3816* BSR EDgetadr ;set address registers
3817* ;A1 = ptr to device description info
3818* ;A2 = ptr to slot controller registers
3819* ;A3 = ptr to slot static RAM
1890 6100 0270 3820* BSR ADmtron ;turn on motor
1894 6100 04D6 3821* BSR ADseek ;get to track specified by D3.W
1890 6604 3822* BNE.S ADscfio? ;if error, return
189A 6100 02B4 3823* BSR ADsecR ;read sector specified by D4.W
3824*
189E 6100 0244 3825* ADscfio? BSR ADmtrof ;turn off motor
18A2 4CDF 7F7F 3826* MOVEM.L (sp)+,D0-D6/A0-A6 ;restore registers
18A6 4A47 3827* TST.W D7 ;set condition codes *Eb 9/3/82*
18A8 4E75 3828* RTS ;return
3829*

```

```

3831* ,
3832* ; index into static RAM to routines
3833* ;
00000014 3834* SRAMIn4 equ  ADrd4E-ADrd4B ;
0000001C 3835* SRAMIn5 equ  ADrd5E-ADrd5B ;
00000020 3836* SRAMIn6 equ  ADrd6E-ADrd6B ;
00000022 3837* SRAMInw equ  ADwaitE-ADwaitB ;
3838* ;
3839* ,RAMrd4 equ  0 ;ADrd4 routine (READ)
3840* ,RAMrd5 equ  SRAMrd4+SRAMIn4 ;ADrd5 routine (READ)
3841* ,RAMrd6 equ  SRAMrd5+SRAMIn5 ;ADrd6 routine (READ)
3842* ,RAMwt equ  SRAMrd6+SRAMIn6 ;ADwaitB routine (SEEK)
3843* ,RAMend equ  SRAMwt+SRAMInw ;Must be (= 9100
3844* ;
00000000 3845* SRAMrd4 equ  500 ;ADrd4 routine (READ)
00000040 3846* SRAMrd5 equ  940 ;ADrd5 routine (READ)
00000080 3847* SRAMrd6 equ  980 ;ADrd6 routine (READ)
000000C0 3848* SRAMwt equ  5C0 ;ADwaitB routine (SEEK)
3849* ;
3850* , PRENIB16 routine niblixes the user data into 6/2 format
3851* , and places in Nbuf1 and Nbuf2 buffers
3852* ;
00000D00 3853* NBUF1 equ  CFIobuf ;MUST BE AT LEAST 256 BYTES
00000E02 3854* NBUF2 equ  NBUF1+256 ;,540 HEX BYTES
3855* ;
3856* , AREA FOR THE TEMPORARIES AND OTHER VARIABLES
3857* ;
00000E62 3858* APLSVAR equ  NBUF2+540
3859* ;
3860* , 1) for 'ADccSK' subroutine
3861* ;
00000000 3862* TRKCNT equ  0
00000001 3863* PRIOR equ  1
00000002 3864* TRKN equ  2
00000003 3865* CURTRK equ  3
3866* ;
3867* , 2) for 'ADccRd'
3868* ;
00000004 3869* csumrd equ  CURTRK+1
3870* ;
3871* , 3) FOR 'ADrdad'
3872* ;
00000E82 3873* AMBUF equ  APLSVAR+520 ;,NEED AT MAX. 32 BYTES FOR TEMP
3874* ;
3875* , Indices to fields in AMBUF
3876* ;
00000000 3877* AMvol equ  0 ;,volume name in adr mark
00000001 3878* AMtrk equ  AMvol+1 ;,track # in adr mark
00000002 3879* AMsec equ  AMtrk+1 ;,sector # in adr mark
00000003 3880* AMchksm equ  AMsec+1 ;,check sum in adr mark
3881* ;

```

```

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. PREMIBL16 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 niblixing. 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. DNIBL1, 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 03 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,$B2,3,3,0,$E3,0,0,2
18E4 03 03 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,$BB,3,1,2,$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 5ED,2,0,3,5EE,1,0,3,5EF,3,0,3,5F2,0,2,3
1974 00 03 EF 03 00 03
197A F2 00 02 03
197E F3 02 02 03 F4 01 3915*      data b 5F3,2,2,3,5F4,1,2,3,5F5,3,2,3,5F6,0,1,3
1984 02 03 F5 03 02 03
198A F6 00 01 03
198E F7 02 31 03 F9 01 3916*      data b 5F7,2,1,3,5F9,1,1,3,5FA,3,1,3,5FB,0,3,3
1994 31 03 FA 03 01 03
199A FB 03 03 03
199E FC 02 03 03 FD 01 3917*      data b 5FC,2,3,3,5FD,1,3,3,5FE,3,3,3,5FE,0,2,3
19A4 03 03 FE 03 03 03
19AA EF 00 02 03
                                     3918*
                                     3919*
                                     3920* ; NIBL table
                                     3921*
19AE 3000 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 0,0,0,0,0,0,0,0,0,0,0,0
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 000,004,598,599,008,50C,59C,510
1A4A 9C 10
1A4C 14 18 A0 A1 A2 A3 3929*      data b 514,518,5A0,5A1,5A2,5A3,5A4,5A5
1A52 A4 A5
1A54 1C 20 A8 A9 AA 24 3930*      data b 51C,520,5A8,5A9,5AA,524,528,52C
1A5A 2E 2C
1A5C 30 34 B0 B1 38 3C 3931*      data b 530,534,5B0,5B1,538,53C,540,544
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,\$6C,\$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,\$8C,\$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	data.b	\$F8,\$FC
	3941*		
	3942*		;it ends on even boundary



```

3944* ;
3945* ; Apple floppy controller equates
3946* ;
3947* ; These are the index values corresponding to the base address of the slot
3948* ; calculated by FDgetadr subroutine
3949* ;
3950* ; Commands to the floppy are issued by setting or resetting the bits in the
3951* ; addressable latch of the type 74LS259.
3952* ; These bits can be set or reset by making a READ/WRITE reference to these
3953* ; addresses. In general, we make a read reference to the addresses assigned
3954* ; to these bits (indexed w.r.t Device Select addresses).
3955* ; For write operations, these bits are set/reset by making a write reference.
3956* ;
3957* ; NOTE:
3958* ; The CONCEPT address bit A1 is tied to APPLE slot address bit A00 and
3959* ; further BIT0 of Apple slot is used to turn on or off a bit in 74LS259.
3960* ; Thus, Apple addresses are 4 * CONCEPT addresses.
3961* ;
3962* ;
3963* ;
3964* ; Address bits of Apple floppy controller to the address bits of CONCEPT
3965* ;
00000002 3966* ad0on equ 2
3967* ;
3968* ; The following are the equates for the bits of the latch
3969* ;
00000000 3970* PHASEOFF equ $0+0 ;turn phase 0 OFF
00000002 3971* PHASEON equ $0+ad0on ;turn phase 0 ON
3972* ;
00000000 3973* PHASE0OFF equ $0+0 ;phase 0 off
00000002 3974* PHASE0ON equ $0+ad0on ;phase 0 on
3975* ;
00000004 3976* PHASE1OFF equ $1*4+0 ;phase 1 off
00000006 3977* PHASE1ON equ $1*4+ad0on ;phase 1 on
3978* ;
00000008 3979* PHASE2OFF equ $2*4+0 ;phase 2 off
0000000A 3980* PHASE2ON equ $2*4+ad0on ;phase 2 on
3981* ;
0000000C 3982* PHASE3OFF equ $3*4+0 ;phase 3 off
0000000E 3983* PHASE3ON equ $3*4+ad0on ;phase 3 on
3984* ;
00000010 3985* MOTOROFF equ $4*4+0 ;motor off
00000012 3986* MOTORON equ $4*4+ad0on ;motor on
3987* ;
00000014 3988* DRV0EN equ $5*4+0 ;drv 0 enable
00000016 3989* DRV1EN equ $5*4+ad0on ;drv 1 enable
3990* ;

```

```

3992* ;
3993* ; Q6 and Q7 define the operation of controller
3994* ;
3995* ; ;Q7 Q6 OPERATION
00000018 3996* Q6L equ $6*4+0 ;L L read disk data
0000001A 3997* Q6H equ $6*4+ad00n ;L H sense write protect
0000001C 3998* Q7L equ $7*4+0 ;H L write disk data
0000001E 3999* Q7H equ $7*4+ad00n ;H H write store
4000* ;
4001* ; To write the disk data.
4002* ; set Q7 high (=1), Q6 low (=0)
4003* ; set Q6 to high then to low
4004* ;
4005* ; To read the disk data.
4006* ; set Q7 low and set Q6 low.
4007* ;
4008* ;
4009* ; GENERAL EQUATES
4010* ;
00000056 4011* LNBUF2 equ $56 ;length of buffer nbuf2
4012*
4013* ;
4014* ; AD15sssd -- Set up constants for Apple 5" single side single density
4015* ;
4016* AD15sssd
1AAE
1AAE 6100 FB30 4017* bsr FDgetadr ;set address registers
4018* ;A1 = ptr to device description info
4019* ;A2 = ptr to slot controller registers
4020* ;A3 = ptr to slot static RAM
1AB2 337C 0118 0734 4021* move.w #NBKSSD,CPfdvss(A1) ;set device size in blocks
1AB8 337C 0100 0736 4022* move.w #BPSS1SD,CPfbps(A1) ;set bytes per sector
1ABE 137C 0010 0738 4023* move.b #SCPTSSD,CPfisp(A1) ;set sectors per track
1AC4 137C 0023 0739 4024* move.b #TKPSSSD,CPftps(A1) ;set tracks per side
1ACA 137C 0001 073A 4025* move.b #1,CPfspd(A1) ;set sides per disk
1AD0 4229 073B 4026* clr.b CPfofst(A1) ;set first track offset
1AD4 137C 0005 073C 4027* move.b #DT4S,CPftyp(A1) ;set floppy type
1ADA 487A 0008+ 4028* pea ADilvtb ;set interleave table pointer
1ADE 235F 0730 4029* move.l (sp)+,CPfiniv(A1) ;*
1AE2 4E75 4030* rts ;return
4031* ;
4032* ; Interleave table for Apple floppy disk drives
4033* ;
1AE4 00 02 04 06 08 0A 4034* 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 07 0B 0D 0F
4035* ;----- data.b $0,$3,$6,$9,$C,$F,$2,$5,$8,$B,$E,$1,$4,$7,$A,$D ;CF/M
4036* ;----- data.b $0,$D,$B,$9,$7,$5,$3,$1,$E,$C,$A,$8,$6,$4,$2,$F ;DOS Basic
4037*

```

```

4039* ;
4040* ; ADinit -- Initialise Appie floppy disk drive
4041* ;
1AF4 6100 FAEA 4042* ADinit BSR FDgetadr ;set address registers
4043* ; ;A1 = pointer to device description info
4044* ; ;A2 = pointer to slot controller registers
4045* ; ;A3 = pointer to slot static RAM
4046* ; *KB 8/23/82* IN ROM DOESN'T NEED ROUTINES IN STATIC RAM
4047* ; ; MOVE WRITE CRITICAL CODE INTO THE STATIC RAM
4048* ;
4049* ; MOVEM.L A4-A6, -(sp) ;save registers
4050* ;
4051* ; LEA Adrd4B,a4 ;BEGINNING OF CRITICAL AREA
4052* ; LEA SRAMrd4(A3),a5 ;WHERE IT GOES
4053* ; LEA Adrd4E,a6 ;AFTER CRITICAL CODE AREA
4054* ; BSR S ADmov ;MOVE CODE
4055* ;
4056* ; LEA Adrd5B,a4 ;BEGINNING OF CRITICAL AREA
4057* ; LEA SRAMrd5(A3),a5 ;WHERE IT GOES
4058* ; LEA Adrd5E,a6 ;AFTER CRITICAL CODE AREA
4059* ; BSR S ADmov ;MOVE CODE
4060* ;
4061* ; LEA Adrd6B,a4 ;BEGINNING OF CRITICAL AREA
4062* ; LEA SRAMrd6(A3),a5 ;WHERE IT GOES
4063* ; LEA Adrd6E,a6 ;AFTER CRITICAL CODE AREA
4064* ; BSR S ADmov ;MOVE CODE
4065* ;
4066* ; LEA AdwaitB,a4 ;BEGINNING OF CRITICAL AREA
4067* ; LEA SRAMwt(A3),a5 ;WHERE IT GOES
4068* ; LEA AdwaitE,a6 ;AFTER CRITICAL CODE AREA
4069* ; BSR S ADmov ;MOVE CODE
4070* ;
4071* ; MOVEM.L (spi),A4-A6 ;restore registers
4072* ;
4073* ; ; do restore of drive
4074* ;
1AF8 6110 4075* BSR S ADmtron ;TURN ON MOTOR
1AFA 6100 02B8 4076* BSR ADrst ;restore to track 0
1AFE 6104 4077* BSR S ADmtrof ;TURN MOTOR OFF
1B00 4A47 4078* TST.W D7 ;set condition codes *kb 9/3/82*
1B02 4E75 4079* RTS ;return
4080* ;
4081* ;ADmov MOVE.W (A4)+,(A5)+ ;move code to static RAM *KB 8/23/82*
4082* ; ;CMPA.L A4,A6 ;finished moving code?
4083* ; ;BNE.S ADmov ;no, move next word
4084* ; ;rts ;return
4085* ;

```

```

4087* ;
4088* ; D0.W -- FREE
4089* ; D1.W -- FREE
4090* ; D2.W -- BYTE CNT
4091* ; D3.W -- BASE BLK ADRS
4092* ; D4.W -- FREE
4093* ; D5.W -- USER CMD
4094* ; D6.W -- FREE
4095* ;
4096* ; A0.L -- USER BUFFER ADDRESS
4097* ; A1.L -- DEVICE DESCRIPTION AREA BASE ADDRESS
4098* ; A2.L -- FLOPPY CONTROLLER BASE ADDRESS
4099* ; A3.L -- STATIC RAM BASE ADDRESS
4100* ;
4101* ;
4102* ;
4103* ; ADmtrof -- turn motor off
4104* ;
4105* ; ADmtrof MOVE.B MOTOROFF(A2),D0 ,TURN OFF FLOPPY DRIVE MOTOR
1B04 102A 0010
1B06 4E75
4106* ; RTS ;return
4107* ;
4108* ;
4109* ; ADmtron -- turn motor on
4110* ;
4111* ; ADmtron MOVE.B MOTORON(A2),D0 ,TURN ON FLOPPY DRIVE MOTOR
1B0A 102A 0012
1B0E 4E75
4112* ; rts ;return
4113* ;

```

```

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   FDEbick    ;jump if first blk rqstd is invalid
1B1A B669 0734      4121*         cmp.w CPIdvss(A1),d3 ;is it in limit
1B1E 6C00 FC88      4122*         bge   FDEbick    ;jump final block exceeds max
                   4123*                               ;
1B22 6100 FAE8      4124*         BSR   FDc1cTS    ;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* ADrdwr: 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 FE1E      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*

```

```

4160* ;
4161* ; ADInlv -- get the physical sector number for the logical sector number
4162* ;         specified in D4.W
4163* ;
4164* ;         Enter: D4.W = logical sector number
4165* ;         Exit:  D4.W = physical sector number
4166* ;
1B40 2F08 4167* ADInlv move.l A0,-(sp)           ,save register
1B42 2078 0730 4168*         move.l CPfinlv.w,A0       ,get interleave table pointer
1B46 6704 4169*         beq.s ADInlv9           ,just return if no table pointer
1B48 1830 4000 4170*         move.b 0(A0,D4.W),D4       ,get physical sector number
1B4C 205F 4171* ADInlv9 move.l (sp)+,A0         ,restore register
1B4E 4E75 4172*         rts                       ,return
4173* ;
4174* ;
4175* ; ADsecR -- (find the desired sector on the current track by using ADfsec
4176* ;         If the sector is found then call ADccRd to read the sector
4177* ;         into the buffer.
4178* ;
00007FFF 4179* ADccSc equ $7FFF           ,retry count *KB 8/23/82*
4180* ;
1B50 48E7 FA4E 4181* ADsecR MOVEM.L D0-D4/D6/A1/A4-A6,-(SP) ,save registers *KB 8/23/82*
1B54 61EA 4182*         bsr.s ADInlv           ,get physical sector number
4183* ;
1B56 40E7 4184*         MOVE.W SR, -(SP)         ,save sr *KB 8/23/82*
1B58 007C 0700 4185*         ORI.W #40700, SR        ,turn off interrupts *KB 8/23/82*
4186* ;
1B5C 3C3C 7FFF 4187*         MOVE.W #ADccSc,D6       ,retry count
1B60 2848 4188*         MOVEA.L A0,A4        ,SAVE A0 INTO A4
4189* ;
1B62 204C 4190* ADsecR1 MOVEA.L A4,A0
4191*         BSR ADfsec           ,find sector
1B64 6100 012A 4192*         BNE.S ADsecR2         ,sector not found - exit
1B68 6608 4193*         BSR.S ADccRd         ,read sector
1B6A 6112 4194*         DBEQ D6,ADsecR1    ,do until(sector read) or (no more retries)
1B6C 57CE FFF4 4195*         BEQ.S ADsecR9
1B70 6702 4196* ;
1B72 204C 4197* ADsecR2 movea.l a4,a0         ,error, so recover a0
4198* ;
1B74 4199* ADsecR9
1B74 46DF 4200*         MOVE.W (SP)+,SR        ,restore sr *KB 8/23/82*
1B76 4CDF 725F 4201*         MOVEM.L (SP)+,D0-D4/D6/A1/A4-A6 ,save registers *KB 8/23/82*
1B7A 4A47 4202*         TST.W D7             ,set return condition code
1B7C 4E75 4203*         RTS                       ,return
4204* ;

```

```

4206* ;
4207* ; ADccRd -- read 8 bit bytes of disk data, retranslates to data,
4208* ; then perform the Exclusive ors to get user data
4209* ;
4210* ; First 56 bytes of nbuf2 data are read in.
4211* ; Then the nbuf1 data is read in three groups
4212* ; first and second group of 86 (56H) data, the last group
4213* ; of 256-86-86 bytes i.e. left over bytes.
4214* ;
4215* ; Group1 is de-nibblized using denibblizing table DNIBL2
4216* ; Group2 is de-nibblized using denibblizing table DNIBL3
4217* ; Group3 is de-nibblized using denibblizing table DNIBL4
4218* ;
4219* ; disk data format
4220* ; -- prologue --- data field ----- cksum --- epilogue ---
4221* ; ; ; ; ;
4222* ; ; DS AA AD ; 342 bytes of disk data ; one byte ; DE AA EB ;
4223* ; ; ; ; ;
4224* ; -----
4225* ;
00000400 4226* ADccRd equ 5400 ;read sector retry count
4227* ;
1B7E 48E7 0268 4228* ADccRd movem.i d6/a4,-(sp) ;*KB 8/23/82*
4229* ;
4230* ; 1) read prologue of disk data field. Try 1024 times.
4231* ; If prologue not found, indicate error
4232* ;
1B82 3E3C 0400 4233* move.w #ADccRd,d7 ;RETRY COUNT *KB 8/24/82*
4234* ;
4235* ; SEARCH FOR PROLOGUE
4236* ;
1B86 51CF 0006 4237* ADccRd1 dbf d7,ADccRd2 ;
1B8A 6C00 0086 4238* bra ADccRd8 ;
4239* ;
1B8E 102A 0018 4240* ADccRd2 move.b Q6L(a2),d0 ; check for DS HEX
1B92 4AFA 4241* bpl.s ADccRd1 ;jmp if byte has not been asmbld
4242* ;
1B94 803C 00D5 4243* ADccRd3 cmp.b #6DS,d0 ;is a part of prologue
1B98 66EC 4244* bne.s ADccRd1 ;no
4245* ;
1B9A 4240 4246* clr.w d0 ;clear bit 8 to 15 of D reg
1B9C 4241 4247* clr.w d1 ;clear bit 8 to 15 of D reg
1B9E 4243 4248* clr.w d3 ;clear bit 8 to 15 of D reg

```

```

1BA0 102A 0018      4250* ADccRd4 move.b G6L(a2),d0      ;check for AA hex
1BA4 6AFA          4251* bpl.s ADccRd4                    ;jmp if byte has not been asmbld
1BA6 B03C 00AA      4252* cmp.b #5AA,d0                    ;is a part of prologue
1BA8 66E8          4253* bne.s ADccRd3                    ;no, see if start of prologue
                                4254*                                ;*KB 8/23/82* DELAY A LITTLE
1BAC 4242          4255* clr.w d2                          ;clear bit 8 to 15 of D reg
1BAE 49FA FDFF+    4256* LEA DNIBL, A4                     ;translate table
                                4257*
1BB2 102A 0018      4258* ADccRd5 move.b G6L(a2),d0      ;check for AD hex
1BB6 6AFA          4259* bpl.s ADccRd5                    ;jmp if byte has not been asmbld
1BB8 B03C 00AD      4260* cmp.b #5AD,d0                    ;is a part of prologue
1BBC 66D6          4261* bne.s ADccRd3                    ;no, see if start of prologue
                                4262*
                                4263* ; 2) prologue has been found, read 56H bytes of disk data into NBUF2
                                4264* ; This DATA is then used to get the LEAST SIGNIFICANT 2 BITS OF A BYTE
                                4265* ; NOTE THIS DATA IS OF THE FORM.
                                4266* ; N55 .eor. 0
                                4267* ; N54 .eor. N55
                                4268* ; N53 .eor. N54
                                4269* ; N52 .eor. N53
                                4270* ; .....
                                4271* ; N01 .eor. N02
                                4272* ; N00 .eor. N01
                                4273* ;
1BBE 4BF8 0E02      4274* LEA NBUF1.W, A5                    ;*KB 8/23/82*
1BC2 3C3C 0055      4275* MOVE.W #LNBUFF1-1, D6             ;*KB 8/23/82*
1BC4 6156          4276* BSR.S ADRd4                       ;Read into NBUF2,*KB 8/23/82*
                                4277* ;jsr SRAMrd4(A3) ;Read NBUF2 ,NOT MOVING TO STATIC RAM
                                4278* ;
                                4279* ; D2 is = N00
                                4280* ;
1BC8 43F8 0D00      4281* LEA NBUF1.W, A1                    ;*KB 8/23/82*
1BCC 3C38 0736      4282* MOVE.W CPfbps.w,D6                ;read 256 bytes of data one byte of chk sum
                                4283* ;and verify epilogue
1BD0 617C          4284* BSR.S ADRd6                       ;*KB 8/23/82*
                                4285* ;jsr SRAMrd6(A3) ;DO ADRd6 CODE (==== LOOK !!!!!)
1BD2 6642          4286* bne.s ADccRd9                    ;jmp if error
                                4287*
                                4288* ; TRANSLATE the data read from the disk.
                                4289* ;
1BD4 43F8 0D00      4290* LEA.L NBUF1.W,A1                  ;REPOSITION PTR TO BEGINNING
                                4291* ;
                                4292* ; 3) translate first group of 56H bytes i.e. 86 bytes
                                4293* ;
1BD8 4BF8 0E02      4294* lea.l NBUF2.W,A5                  ;RDS GROUP
1BDC 4DFA FCCD+    4295* lea.l DNIBL2,a6
1BE0 3C3C 0055      4296* move.w #LNBUFF1-1,d6
1BE4 614C          4297* BSR.S ADRd5                       ;*KB 8/23/82*

```



```

4299*
4300* ; 4) translate second group of 56H bytes      RD6 GROUP
4301* ; D2 contains ASS
4302*
1BEA 4BF8 0E02 4303* lea l NBUF1.W,a5
1BEA 4DFA FCC0+ 4304* lea l DNIBL3,a6
1BEE 3C3C 0055 4305* move w $LNBUF2-1,d6
1BF1 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-LNBUF1)-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 ; dnibsize disk data byte
1C0E 8302 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 1640 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

```

```

4333* ;
4334* ; ADrd4 -- moved to and executed at SRAMrd4(slot static RAM)
4335* ;
1C1E      4336* ADrd4B      ;start of SRAMrd4 code
1C1E 102A 0018 4337* ADrd4   move.b Q6L(a2),d0 ;
1C22 6AFA      4338*      bpl.s ADrd4   ;jmp if byte has not been asmbld
1C24 1234 0000 4339*      move.b 0(a4,d0),d1 ;dnblize disk data byte
1C28 1AC1      4340*      move.b d1,(a5)+ ;store it. it is N(I) EGR N(I-1)
1C2A B302      4341*      eor.b d1,d1 ;ULTIMATELY d2 = N00
1C2C 51CE FF00 4342*      dbf d6,ADrd4 ;
1C30 4E75      4343*      rts ;return
1C32      4344* ADrd4E      ;end of SRAMrd4 code
4345* ;
4346* ;
4347* ; ADrd5 -- Translate bytes **not done in time critical section**
4348* ;
1C32      4349* ADrd5B      ;start of SRAMrd5 code
1C32 1019      4350* ADrd5   move.b (a1)+,d0 ;.16
1C34 1234 0000 4351*      move.b 0(a4,d0),d1 ;.18 dnblize disk data byte
1C38 B302      4352*      eor.b d1,d1 ;.4
1C3A 161D      4353*      move.b (a5)+,d3 ;.12 get an entry of nbuf2
1C3C 1E36 3000 4354*      move.b 0(a6,d3),d7 ;.18 get low order bits b1,B0 of
1C40 BF02      4355*      eor.b d7,d2 ;.4 a byte and mask them into byte
1C42 10C2      4356*      move.b d2,(a0)+ ;.14 store the byte into user area
1C44 51CE FFEC 4357*      dbf d6,ADrd5 ;.10
4358* ;
4359* ;.TOTAL = 106 CYCLES
1C48 C43C 00FC 4359*      and.b #9FC,d2 ;mask of 2 LSBits. Now d2 contains A55
1C4C 4E75      4360*      rts ;return
1C4E      4361* ADrd5E      ;end of SRAMrd5 code
4362* ;
4363* ;
4364* ; ADrd6 -- moved to and executed at SRAMrd6(slot static RAM)
4365* ;
1C4E      4366* ADrd6B      ;start of SRAMrd6 code
1C4E      4367* ADrd6 ;
1C4E 102A 0018 4368* ADrd61  move.b Q6L(a2),d0 ;.read 256 bytes of data and one byte of CHK SUM
1C52 6AFA      4369*      bpl.s ADrd61 ;
1C54 12C0      4370*      move.b d0,(a1)+ ;.move bytes into BUFADRS
1C56 51CE FF64 4371*      dbf d6,ADrd61 ;
4372* ;
4373* ; READ epilogue
4374* ;
1C5A 102A 0018 4375* ADrd62  move.b Q6L(a2),d0 ;
1C5E 6AFA      4376*      bpl.s ADrd62 ;.jmp if byte has not been asmbld
1C60 4247      4377*      clr.w d7 ;.pre-set d7 =0 to indicate no error
1C62 B03C 00DE 4378*      cmp.b #9DE,d0 ;
1C66 6702      4379*      beq.s ADrd69 ;.jmp if ok
1C68 7EED      4380*      moveq #RRNF,d7 ;.else move error code to d7
4381* ;
1C6A 4A47      4382* ADrd69  tst.w d7 ;.set return condition code
1C6C 4E75      4383*      rts ;return
1C6E      4384* ADrd6E      ;end of SRAMrd6 code
4385* ;

```

```

4387* ;
4388* ; ADwaitB -- moved to and executed at SRAMwt(slot static RAM)
4389* ;          delay of units specified in D7
4390* ;          Each unit is 100 micro-seconds
4391* ;
0000000A 4392* dly100m equ    010          ;
4393* ;
1C6E      4394* ADwaitB          ;start of SRAMwt code
1C6E 48A7 9400 4395*      movem.w d5/d6,-(sp) ;save registers
1C72 3C3C 000A 4396* ADwait1 move.w #dly100m,d6 ;each count of d6 = 1100/112 = 10 micro second
1C76 4E71 4397* ADwait2 nop ;delay 100 micro-seconds
1C78 4E71 4398*      nop ;
1C7A 4E71 4399*      nop ;
1C7C 4E71 4400*      nop ;
1C7E 4E71 4401*      nop ;
1C80 4E71 4402*      nop ;
1C82 51CE FFF2 4403*      dbf    d6,ADwait2 ;
1C86 51CF FFEA 4404*      dbf    d7,ADwait1 ;
1C8A 4E9F 0040 4405*      movem.w (sp)+,d5/d6 ;restore registers
1C8E 4E75 4406*      rts ;return
1C90      4407* ADwaitE          ;end of SRAMwt code
4408* ;

```

```

4410* ;
4411* ; ADfsec -- call ADrdad until it finds the sector specified in D4.B or
4412* ;         it has no more retries left.
4413* ;         Calling routine must disable all the interrupts before making call
4414* ;
00000064 4415* ADfsnrv equ 100 ;number of revolutions until record not found *KB 8/23/82*
4416*
1C90 48E7 0208 4417* ADfsec MOVEM.L D6/A4,-(sp) ;save register *KB 8/23/82*
1C94 1C38 0738 4418* move.b CPfspt.w,d6 ;compute retry count
1C98 4886 4419* ext.w d6 ;
1C9A CCFC 0064 4420* mulu #ADfsnrv,D6 ;
4421*
1C9E 611C 4422* ADfsec1 BSR.S ADrdad ;get this sectors adr mark
1CA0 4A47 4423* tst.w d7 ;ERROR?
1CA2 6608 4424* bne.s ADfsec1 ;YES, RETRY
4425* ;
4426* ; see if this is the correct track and sector number
4427* ; *KB 8/23/82* removed track check
1CA4 4247 4428* clr.w d7 ;CLR error code to indicate no error
1CA6 B038 0E84 4429* CMP.B AMBUF+AMsec.W,D4 ;found sector?
1CAA 6708 4430* BEQ.S ADfsec9 ;successful ,so exit
1CAC 51CE FFF0 4431* ADfsec2 DBF D6,ADfsec1 ;do until(found sector) or (no more retries)
4432* ;
4433* ; RETRIES FAILED
4434* ;
1CB0 3E3C FFED 4435* ADfsec8 MOVE.W #RRNF,D7
4436* ;
1CB4 4CDF 1040 4437* ADfsec9 MOVEM.L (sp)+,D6/A4 ;save register *KB 8/23/82*
1CB8 4A47 4438* tst.w d7 ;set return condition code
1CBA 4E75 4439* RTS ;return
4440* ;

```

```

4442* ;
4443* ; ADrdad -- read the address field of the Appie floppy
4444* ;
4445* ; Format of the address field:
4446* ;
4447* ;-- prologue --+ vol name --+ track --+ sector --+ cksum --+ epilogue --+
4448* ; : : : : : : : : : : : :
4449* ; DS AA 96 : XX YY : XX YY : XX YY : DE AA EB :
4450* ; : : : : : : : : : : : :
4451* -----
4452* ;
4453* ; The data byte is split into 4 by 4 format, and then is written to disk.
4454* ; A data byte d7 d6 d5 d4 d3 d2 d1 d0 is split as follows:
4455* ;
4456* ; dsk byte1: 1 d7 1 d5 1 d3 1 d1
4457* ; dsk byte2: 1 d6 1 d4 1 d2 1 d0
4458* ;
4459* ; Shifting the first byte left by 1 and .AND. with next byte,
4460* ; the original data byte can be reconstructed.
4461* ;
4462* ; Calling routine must disable interrupts and save the registers D0-D2/A4.
4463* ; *KB 8/23/82*
4464* ;
00000400 4465* ADrcAd equ 5400 ;read address retry count
4466* ;
ICBC 3E3C 0400 4467* ADrdad move.w #ADrcAd,d7 ;set retry count
4468* ;
ICCG 51CF 0006 4469* ADrdad1 dbf d7,ADrdad2 ;if no more retries
ICCA 6000 00A0 4470* bra ADrdad8 ;then report error
4471* ;
4472* ; FIND PROLOGUE
4473* ;
ICCB 102A 0018 4474* ADrdad2 move.b Q6L(a2),d0 ;check for DS HEX
ICCC 6AFA 4475* bpl.s ADrdad2 ;wait until data ready
ICCE B03C 00D5 4476* ADrdad3 cmp.b #D5,d0 ;is a part of prologue
ICD1 66EC 4477* bne.s ADrdad1 ;no, try again
ICD4 4242 4478* clr.w d2 ;clear cksum
4479* ;
ICD6 4480* ADrdad3a
4481* ; move.b Q6L(a2),d0 ;check for AA hex
ICD8 102A 0018 4482* bpl.s ADrdad3a ;wait until data ready
ICDA 6AFA 4483* cmp.b #5AA,d0 ;is a part of prologue
ICDC B03C 00AA 4484* bne.s ADrdad3 ;no
ICDE 66EC 4485* lea AMBUF.W,a4 ;delay and get adr mark buf addr
ICDF 49F8 0E82 4486* ;
4487* ADrdad3b
ICD9 102A 0018 4488* ; move.b Q6L(a2),d0 ;check for 96 hex
ICDB 6AFA 4489* ; bpl.s ADrdad3b ;wait until data ready
ICDD B03C 0096 4490* ; cmp.b #596,d0 ;is a part of prologue
ICDF 66DC 4491* ; bne.s ADrdad3 ;no
4492* ;
4493* ; prologue has been found. Now read the vol header
4494* ; read 4 bytes: | vol name | track | sector | cksum |
4495* ;

```

```

ICF1 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*
ICF6 4500* ADrdad4A      ,VOL NAME
ICF6 102A 0018 4501*      move.b  Q6L(a2),d0    ,read first nibble
ICFA 6AFA 4502*      bpl.s  ADrdad4A      ,wait until data ready
ICFC E318 4503*      rol.b  #1,d0      ,rotate left by 1
ICFE 4504* ADrdad5A
ICFE 122A 0018 4505*      move.b  Q6L(a2),d1    ,read second nibble
ID02 6AFA 4506*      bpl.s  ADrdad5A      ,jump if byte has not been asmbid
ID04 C200 4507*      and.b  d0,d1        ,and the two to get actual byte
ID04 18C1 4508*      move.b  d1,(a4)+     ,store it in AMBUF
ID08 B302 4509*      eor.b  d1,d2        ,create chksum
4510*
ID0A 4511* ADrdad4B      ,TRACK NUMBER
ID0A 102A 0018 4512*      move.b  Q6L(a2),d0    ,read first nibble
ID0E 6AFA 4513*      bpl.s  ADrdad4B      ,wait until data ready
ID10 E318 4514*      rol.b  #1,d0      ,rotate left by 1
ID12 4515* ADrdad5B
ID12 122A 0018 4516*      move.b  Q6L(a2),d1    ,read second nibble
ID16 6AFA 4517*      bpl.s  ADrdad5B      ,jump if byte has not been asmbid
ID18 C200 4518*      and.b  d0,d1        ,and the two to get actual byte
ID1A 18C1 4519*      move.b  d1,(a4)+     ,store it in AMBUF
ID1C B302 4520*      eor.b  d1,d2        ,create chksum
4521*
ID1E 4522* ADrdad4C      ,SECTOR NUMBER
ID1E 102A 0018 4523*      move.b  Q6L(a2),d0    ,read first nibble
ID22 6AFA 4524*      bpl.s  ADrdad4C      ,wait until data ready
ID24 E318 4525*      rol.b  #1,d0      ,rotate left by 1
ID26 4526* ADrdad5C
ID26 122A 0018 4527*      move.b  Q6L(a2),d1    ,read second nibble
ID2A 6AFA 4528*      bpl.s  ADrdad5C      ,jump if byte has not been asmbid
ID2C C200 4529*      and.b  d0,d1        ,and the two to get actual byte
ID2E 18C1 4530*      move.b  d1,(a4)+     ,store it in AMBUF
ID30 B302 4531*      eor.b  d1,d2        ,create chksum
4532*
ID32 4533* ADrdad4D      ,CHECKSUM
ID32 102A 0018 4534*      move.b  Q6L(a2),d0    ,read first nibble
ID36 6AFA 4535*      bpl.s  ADrdad4D      ,wait until data ready
ID38 E318 4536*      rol.b  #1,d0      ,rotate left by 1
ID3A 4537* ADrdad5D
ID3A 122A 0018 4538*      move.b  Q6L(a2),d1    ,read second nibble
ID3E 6AFA 4539*      bpl.s  ADrdad5D      ,jump if byte has not been asmbid
ID40 C200 4540*      and.b  d0,d1        ,and the two to get actual byte
ID42 18C1 4541*      move.b  d1,(a4)+     ,store it in AMBUF
ID44 B302 4542*      eor.b  d1,d2        ,create chksum
4543*
ID46 4A02 4544*      tst.b  d2            ,compare chksum to 0
ID48 661C 4545*      bne.s  ADrdad8      ,return if chksum not = 0
4546*
4547*      , VERIFY EPILOGUE (DE AA EB)
4548*
ID4A 102A 0018 4549* ADrdad6 move.b  Q6L(a2),d0 ,get data byte

```

```
1D4E 6AFA          4550*      bpl.s  ADrdad6      ;jump if byte has not been asmbid
1D50 B03C 00DE    4551*      cmp.b  #5DE,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 asmbid
1D5E B03C 00AA    4558*      cmp.b  #5AA,d0      ;
1D62 6602          4559*      bne.s  ADrdad8      ;return error code if error
1D64 4E75          4560*      rts
                      4561*
1D66 3E3C FEED    4562* ADrdad8 move.w  #RRNF,d7      ;error code to d7
1D6A 4E75          4563*      rts                ;return
```

```

4565* ;
4566* ; ADseek -- seek a given track
4567* ;
4568* ; Enter: D3.W - desired track number
4569* ;
4570*
00000100 4571* ADrcSk equ $100 ;retry count
4572*
1D4C 48E7 F208 4573* ADseek movem.l d0-d3/d6/a4,-(sp)
4574*
1D70 3C3C 0100 4575* ADsk1 MOVE.W #ADrcSk,D6 ;retry count
1D74 4247 4576* clr.w d7
4577*
1D76 40E7 4578* ADsk2 move.w sr,-(sp)
1D78 007C 0700 4579* ori.w #0700,sr ;disable interrupts
1D7C 6100 FF3E 4580* BSR ADrdad ;get track number where head is
1D80 46DF 4581* move.w (sp)+,sr ;restore the interrupt
4582*
1D82 4A47 4583* tst.w d7 ;ERROR?
1D84 4706 4584* beq.s ADsk3 ;if no error then jump
4585*
4586* ; ; bsr.s ADrst ;go to track 0
1D86 51CE FFEE 4587* DBF D6,ADsk2 ;do until(found sector
1D8A 6020 4588* bra.s ADsk9 ;tried enough
4589* ;
4590* ; See if head is positioned on desired track.
4591* ; issue STEPIN or STEPOUT to get to the correct track.
4592* ;
1D8C 49E8 0E42 4593* ADsk3 lea.l APLSVAR.W,a4
4594*
1D90 4240 4595* clr.w d0 ;form the current half
1D92 1038 0E83 4596* move.b AMBUF+AMtrk.W,d0 ;track and save
1D96 E348 4597* lsl.w #1,d0 ;half track into d0
1D98 1940 0063 4598* move.b d0,curtrk(a4) ;half track number to
4599*
1D9C B638 0E83 4600* CMP.b AMBUF+AMtrk.W,d3
1DA0 670A 4601* beq.s ADsk9 ;exit if equal,found track
1DA2 6D04 4602* bit.s ADsk4 ;jump if track is ( current track
4603*
1DA4 6132 4604* bsr.s ADcc5in ;step in
1DA6 60C8 4605* bra.s ADsk1 ;see if another track
4606*
1DA8 612A 4607* ADsk4 bsr.s ADcc5out ;step out
1DAA 60C4 4608* bra.s ADsk1 ;see if another track
4609*
1DAC 4CDF 104F 4610* ADsk9 movem.l (sp)+,d0-d3/d6/a4
1DB0 4A47 4611* tst.w d7 ;set condition codes
1DB2 4E75 4612* rts

```



```

4614* ,
4615* ; ADrst -- restore floppy to track 0
4616* ,
1DB4 48E7 9008 4617* ADrst  movem.l d0/d3/a4,-(sp)
1DB8 49F8 0E42 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 1089 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* ,
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 0E42 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

```

```

4664* ;
4665* ; ADccSk -- seek the track desired by the caller
4666* ;
4667* ; Enter: D3.W - desired track number
4668* ; CURTRK location holds the current track number.
4669* ;
4670* ; Exit: The final track -----) curtrk
4671* ; The curtrk-1 -----) PRIOR
4672* ;
4673* ; APPLE DOS NAME CONVENTION
4674* ;
4675* ; (TRKN) = Desired track number
4676* ; (CURTRK) = The current track number where floppy heads are positioned
4677* ; (PRIOR) = The (CURTRK)-1
4678* ; (TRKCNT) = The number of track the floppy has been moved so far.
4679* ;

1E10 48E7 8608 4680* ADccSk movem.l d0/d5-d6/a4,-(sp) ;save registers
1E14 4245 4681* clr.w d5 ;clear bit 8 to 15
1E16 4246 4682* clr.w d6 ;clear bit 8 to 15
1E18 4247 4683* clr.w d7 ;clear bit 8 to 15
1E1A 49F8 0E62 4684* lea APLSVAR.W,a4 ;
1E1E B62C 0003 4685* cmp.b curtrk(a4),d3 ;is desired trk = current track
1E22 6764 4686* beq.s ADccSk9 ;exit if equal.
4687* ;

1E24 422C 0000 4688* clr.b trkcnt(a4) ;init track count
1E28 40E7 4689* move.w sr,-(sp) ;save interrupt level
1E2A 007C 0700 4690* ori.w #0700,sr ;disable interrupts
4691* ;

1E2E 196C 0003 0001 4692* ADccSk1 move.b curtrk(a4),prior(a4) ;curtrk to prior
1E34 B62C 0003 4693* cmp.b curtrk(a4),d3
1E38 673E 4694* beq.s ADccSk8 ;jump if current trk = desired trk
1E3A 6D0E 4695* bit.s ADccSk2 ;jump if desired trk( CURTRK
4696* ;

1E3C 1E03 4697* move.b d3,d7 ; Seek IN
1E3E 9E2C 0003 4698* sub.b curtrk(a4),d7 ;trkn - curtrk --) d7
1E42 5307 4699* subq.b #1,d7 ;trkn - curtrk -1 to d7
1E44 522C 0003 4700* addq.b #1,curtrk(a4) ;
1E48 600C 4701* bra.s ADccSk3 ;
4702* ;

1E4A 1E2C 0003 4703* ADccSk2 move.b curtrk(a4),d7 ;curtrk is ) desired track
1E4E 9E03 4704* sub.b d3,d7 ; Seek OUT
1E50 5307 4705* subq.b #1,d7 ;trkn - curtrk -1 to d7
1E52 532C 0003 4706* subq.b #1,curtrk(a4) ;
4707* ;

1E56 BE2C 0000 4708* ADccSk3 cmp.b trkcnt(a4),d7 ;calculate index to the delay table
1E5A 6D04 4709* bit.s ADccSk4 ;jump if d7 is less than trkcnt
1E5C 1E2C 0000 4710* move.b trkcnt(a4),d7 ;
4711* ;

1E60 BE3C 0008 4712* ADccSk4 cmp.b #08,d7 ; destination .cmp. source
1E64 6C02 4713* bge.s ADccSk5 ; leave d0 alone if d7 >= 08
4714* ; to use as index into turn ON delay OFFdelay
1E66 1007 4715* move.b d7,d0 ; do STEP
1E68 1A2C 0003 4716* ADccSk5 move.b curtrk(a4),d5
1E6C 1C2C 0001 4717* move.b prior(a4),d6 ;

```

```

1E70 611C          4718*      BSR.S  ADccStp          ;
1E72 522C 0000    4719*      addq.b  #1,tkcnt(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 2100    4725*      move.w  #100,d7         ;
1E84 6100 FDE8    4726*      BSR     ADwaitB         ;wait
4727* ,,        jsr     SRAMwt(A3)    ;ADwaitB
4728*
1E88 4CDF 1041    4729* ADccSk9 movem.l (sp)+,d0/d5-d6/a4    ;restore registers
1E8C 4E75          4730*      rts                    ;return

```

```

4732* ;
4733* ; ADccStp -- move the floppy in or out by one phase i.e. Half track
4734* ;
4735* ; Enter: the track num to use for setting a phase ----> d5
4736* ; the track num to use for clearing a phase ----> d6
4737* ; the index for ADtBION and ADtBIOF to fetch a dealy count --> d0
4738* ;
4739* ; Note: THE BIT 8 TO 15 OF D0,D5,D6 MUST BE ZERO.
4740* ;
4741* ; Exit: SETS one phase and clears a phase
4742* ;
4743* ; TO MOVE FLOPPY HALF TRACK INWARD.
4744* ; set phase i
4745* ; clear phase i-1
4746* ;
4747* ; TO MOVE FLOPPY HALF TRACK OUTWARD:
4748* ; set phase i-1
4749* ; clear phase i
4750* ;
1E8E 48E7 0802 4751* ADccStp movem.l d4/a6,-(sp) ;
1E92 6124 4752* bsr.s ADsetPh ;turn on phase
1E94 4247 4753* clr.w d7 ;clear bit 8 to 15 of D7
1E96 4DFA 0038+ 4754* lea ADtBION,a6 ;
1E9A 1E36 0000 4755* move.b 0(a6,d0),d7 ;
1E9E 6100 FDCE 4756* BSR ADwaitB ;
4757* ;; jsr SRAMwt(A3) ;ADwaitB
1EA2 6120 4758* bsr.s ADclrPh ;turn off phase
1EA4 4247 4759* clr.w d7 ;clear bit 8 to 15 of D7
1EA6 4DFA 0030+ 4760* lea ADtBIOF,a6 ;
1EAA 1E36 0000 4761* move.b 0(a6,d0),d7 ;
1EAE 6100 FDBE 4762* BSR ADwaitB ;
4763* ;; jsr SRAMwt(A3) ;ADwaitB
1EB2 4CDF 4010 4764* movem.l (sp)+,d4/a6 ;
1EB6 4E75 4765* rts ;
4766* ;
1EB8 0205 0003 4767* ADsetPh andi.b #03,d5 ;
1EBC E50D 4768* lsl.b #2,d5 ;GENERATE APPLE ADRS
1EBE 1E32 5002 4769* move.b phaseon(a2,d5),d7 ;
1EC1 4E75 4770* rts ;
4771* ;
1EC4 0206 0003 4772* ADclrPh andi.b #03,d6 ;
1EC8 E50E 4773* lsl.b #2,d6 ;GENERATE APPLE ADRS
1ECA 1E32 6006 4774* move.b phaseoff(a2,d6),d7 ;
1ECE 4E75 4775* rts ;
4776* ;
1ED0 00 2F 27 23 4777* ADtBION DATA.B 1-1,$30-1,$28-1,$24-1 ;
1ED4 1F 1D 1C 1B 4778* DATA.B $20-1,$1E-1,$1D-1,$1C-1 ;even number of bytes
4779* ;
1ED8 6F 2B 25 21 4780* ADtBIOF data.b $70-1,$2C-1,$24-1,$22-1 ;
1EDC 1E 1D 1C 1B 4781* DATA.B $1F-1,$1E-1,$1D-1,$1C-1 ;even number of bytes
4782* ;

```

00010084, 4784\* end setup

ABOOT	011814+	ADR2	011C5A+	AMTRK	00000001	CPIVEC6	00010084	CSTEP	00000020
ABOOT1	011836+	ADR69	011C6A+	AMVOL	00000000	CPIVEC7	00010088	CSTEPIN	00000040
ABOOT90	011854+	ADR68	011C4E+	APLSVAR	00000E62	CPKBCGCT	00010054	CSTEPPOUT	00000060
ADON	00000002	ADR6E	011C6E+	BADDEST	00000008	CPKBINIT	00010050	CSUMRD	00000004
ADBI01	01186C+	ADRAD	011C8C+	BADSOCK	00000008	CPLBLKIO	00010026	CURSON	00000002
ADBI09	011874+	ADRAD1	011CC0+	BASERAM	00000900	CPLBOOT	00010022	CURTRK	00000003
ADBLK10	011856+	ADRAD2	011CC8+	BLKSZ	00000200	CPLBOOTJ	00010020	CWRSEC	000000A0
ADCCRD	01187E+	ADRAD3	011CCE+	BPS1SD	00000100	CPLDSKIO	0001002A	CWRTRK	000000F0
ADCCRD1	011886+	ADRAD3A	011CD6+	BPS1DD	00000100	CPBLKIO	00010016	DCBLKH1	00000003
ADCCRD2	01188E+	ADRAD3B	011CE6+	BPS1SD	00000080	CPBOOT	00010012	DCBLKLO	00000002
ADCCRD3	011894+	ADRAD4	011CF6+	CFRCINT	000000D0	CPBOOTJ	00010010	DCDRV	00000001
ADCCRD4	0118A0+	ADRAD4B	011D0A+	CHEND	010188+	CPD0SKIO	0001001A	DCLEN	00000004
ADCCRD5	0118B2+	ADRAD4C	011D1E+	CHEND1	01018C+	CPOMNIBF	0008DFD0	DCMD	00000000
ADCCRD8	011C12+	ADRAD4D	011D32+	CHERR	010176+	CPOMNIRC	0000070F	DEBOP	00000006
ADCCRD9	011C16+	ADRAD5A	011CFE+	CHERR1	01017A+	CPOMNIRAM	00000880	DEVAD0F5	00000020
ADCCSIN	011DD8+	ADRAD5B	011D12+	CHDACFT	000000FE	CPOSBLK	00000709	DLY100M	000000DA
ADCCSK	011E10+	ADRAD5C	011D26+	CNSTSKT	000000B0	CPOSDRV	00000708	DN1BL	0119AE+
ADCCSK1	011E2E+	ADRAD5D	011D3A+	CPBLKIO	00010042	CPOSSL0T	00000706	DN1BL2	0118AB+
ADCCSK2	011E4A+	ADRAD6	011D4A+	CPAINIT	0001004A	CPOSSRVR	00000707	DN1BL3	0118AC+
ADCCSK3	011E56+	ADRAD7	011D58+	CPASCTIO	00010046	CPROMLVL	0001000D	DN1BL4	0118AD+
ADCCSK4	011E60+	ADRAD8	011D66+	CPBLKIO	00000714	CPROMVRS	0001000C	DRVOEN	00000014
ADCCSK5	011E68+	ADROWR	011E10+	CPBTSLOT	00000700	CPSCNOFS	00000744	DRV1EN	00000014
ADCCSK8	011E78+	ADROWR1	011E30+	CPBTSRVR	00000701	CPSL1RAM	00000900	D5ADDR	010CCC+
ADCCSK9	011E88+	ADROWR9	011E3E+	CPCKSUM	0001000E	CPSL1TYP	00000771	D5ADDRH	010CDB+
ADCCSOUT	011D04+	ADRS	011D84+	CPDISKRC	0000070E	CPSL2RAM	00000A00	D5ADDRV	010CFF+
ADCCSTP	011E8E+	ADSCIO1	011E88+	CPDSCVUC	0001006C	CPSL2TYP	00000772	D5CLBLK	00000020
ADCLRPH	011E4C+	ADSCIO9	011E9E+	CPDSINIT	00010060	CPSL3RAM	00000800	D5CCR	0000000D
ADFSAC	011C90+	ADSECI0	011E7E+	CPDSKIO	00000718	CPSL3TYP	00000773	D5CDDFF	00000020
ADFSAC1	011C9E+	ADSECR	011E50+	CPDSPFLG	00000766	CPSL4RAM	00000C00	D5CELLV	00000006
ADFSAC2	011CAC+	ADSECR1	011E62+	CPDSPUTC	00010064	CPSL4TYP	00000774	D5CELLY	0000000A
ADFSAC8	011CB0+	ADSECR2	011E72+	CPDSPUTS	00010068	CPSL5TYP	00000775	D5CESL	0000001B
ADFSAC9	011CB4+	ADSECR9	011E74+	CPEXTCRT	00000F00	CPSTACT	00000F00	D5CLAL	010BE0+
ADFSNRV	00000044	ADSEEK	011D6C+	CPFLKIO	00010036	CPSYSRST	00010004	D5CLCA	00000061
ADISSSD	011AAE+	ADSEEK1	011DFC+	CPFB00T	00010032	CPSYSST	00000F01	D5CLCZ	0000007A
ADILVT8	011AE4+	ADSETPH	011EB8+	CPFB00TJ	00010030	CPTPRNBR	0000070D	D5CLEL	010C1E+
ADINIT	011AF4+	ADSI01	011DDA+	CPFBPS	00000736	CPUNIQID	00010008	D5CLEL1	010C34+
ADINLV	011B40+	ADSI02	011DF0+	CPFDV5Z	00000734	CPUSERID	00000720	D5CLEL2	010C3A+
ADINLV9	011B4C+	ADSI03	011DF2+	CPFINIT	0001003E	CPUSERNM	00000726	D5CLES	010E4+
ADMTR0F	011B04+	ADSK1	011D70+	CPFINLV	00000730	CPWDRCD	00000740	D5CLES1	010E2+
ADMTR0N	011B0A+	ADSK2	011D76+	CPFOFST	0000073B	CRDAM	000000C0	D5CLES2	010C86+
ADRCAD	00000400	ADSK3	011D8C+	CPFSCCTIO	0001003A	CRDSEC	00000080	D5CLES3	010C0A+
ADRCRD	00000400	ADSK4	011DA8+	CPFSPD	0000073A	CRDTRK	000000E0	D5CLES9	010C1C+
ADRC5C	00007FFF	ADSK9	011DAC+	CPFSPT	00000738	CRESTORE	00000000	D5CLRH	010C5C+
ADRC5K	00000100	ADTBLOF	011ED8+	CPFTPS	00000739	CSATTR1	00000010	D5CLRH1	010C72+
ADR4	011C1E+	ADTBLOM	011ED0+	CPFTYP	0000073C	CSATTR2	00000011	D5CLRH2	010C84+
ADR4B	011C1E+	ADWAIT1	011C72+	CPIOBUF	00000D00	CSBPC	00000006	D5CLRH3	010C8C+
ADR4E	011C32+	ADWAIT2	011C76+	CPISSTACK	00000FFC	CSDATA	00000012	D5CLRH4	010C90+
ADR5	011C32+	ADWAITB	011C6E+	CPIVEC1	00010070	CSEEX	00000010	D5CLRH5	010C96+
ADR5B	011C32+	ADWAITE	011C90+	CPIVEC2	00010074	CSFRSTCH	00000008	D5CLRH6	010C9C+
ADR5E	011C4E+	AMBUF	00000E82	CPIVEC3	00010078	CSLASTCH	0000000A	D5CLRV	010C3E+
ADR6	011C4E+	AMCHKSM	00000003	CPIVEC4	0001007C	CSLPCB	00000004	D5CLRV1	010C52+
ADR61	011C4E+	AMSEC	00000002	CPIVEC5	00010080	CSMASK	0000000C	D5CLRAD	010BCC+

DSCRS0	010B4+	DSSHOW9	010B1A+	FDCSTPOT	00000075	FDSEEK1	0116EE+	IOBROTSW	00030F61
DSCRS0	010B6C+	DSSHW71	010ADA+	FDCSTRR	00000010	FDSEEK8	011706+	IOPBASE	00030000
DSCRS0	010B8E+	DSSHW72	010AE4+	FDCSTRK	00000012	FDSEEK9	011708+	IT01	01054A+
DSCRS1	010B5E+	DSSHW73	010AEE+	FDEBLCK	0117D8+	FD5101	01146E+	IT02	010558+
DSCRSR	010B1C+	DSSHW74	010AF8+	FDEBUSY	011802+	FD5102	011482+	IT99	010562+
DSCRSU	010B32+	DSSHW75	010B02+	FDECRC	0117D8+	FD5109	011486+	IVLVL1	00000064
DSCSETH	010D4E+	DSSHW76	010B0C+	FDEHERR	0117EA+	FD5KST1	0117BA+	IVLVL2	00000068
DSCSETV	010D84+	DSSHWCH	010A44+	FDENRST	01180E+	FDSKSTA	011786+	IVLVL3	0000004C
DSCTBL	010D08+	DSST0	010A0E+	FDEOPCD	0117E4+	FD5WSTA	0117AA+	IVLVL4	00000070
DSCTL	010A22+	DSSTBL	010D24+	FDEPROT	0117F0+	FDTMOH1	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+	DSWNH	010D2A+	FDESEK	0117F6+	FDWRDY1	0116BE+	JUMPTO	00004EF9
DSCURS2	010B8A+	DSWNV	010D60+	FDEUNIT	0117DE+	FDWRDY9	0116D2+	KBBBUFR	0000000C
DSCURS3	010BC0+	DSWRAP	010B8C+	FDGETADR	0115E0+	FHLD	00000008	KBBFLGS	00000000
DSCVTU1	010Y4+	DSWRAP1	010B78+	FD1855	0114CC+	FINTIDXP	00000004	KBBFRMT	00000002
DSCVTUC	010B84+	DTAS	00000005	FD1855DD	0114B0+	FINTIMM	00000008	KBBLEN	000000F4
DSDECI	010B60+	DTCS	00000004	FD1855SD	011492+	FINTNRDY	00000002	KBBREAR	00000004
DSDECY	010B7E+	DTCS	00000003	FDINCT1	01166A+	FINTROY	00000001	KBBRSV	00000004
DSDEF0F	00000060	DTLOCL	00000001	FDINCT2	011690+	FL1	0105B2+	KBC600	00000017
DSESC	010A3E+	DTNDEV	00000000	FDINCT5	0116A0+	FL2	0105B8+	KBCCBRK	00000008
DSESC1	010A44+	DTOMN1	00000002	FDINCT9	0116AC+	FL3	0105C4+	KBCCGO	00000009
DSESC2	010A54+	ESHOED	0000000C	FDINCTS	011658+	FL4	0105CE+	KBCCNTL	000000FD
DSETBL	010D14+	ECHOOP	00000002	FDINIT	0114DA+	FL5	0105DC+	KBCCOFF	00000002
DSEI1T	010A08+	ENDOP	00000010	FDLCMD	011500+	FLASH	010596+	KBCLCA	00000061
DSHOMEH	0008D55E	FB00T	01140E+	FDLCMD01	011504+	FMP5	00000010	KBCLCZ	0000007A
DSHOMEV	0008D506	FB00T1	011430+	FDLY	00000004	FSDCMPEN	00000002	KBCLOCK	000000FC
DSINCI	010B20+	FB00T90	01144A+	FDMTROF	0114EE+	FSDCPM	00000008	KBCNOCH	000000FE
DSINCY	010B48+	FD8LK10	01144C+	FDNRDY	011768+	FSTPRT10	00000002	KBCNMRK	0000003F
DSINIT	010Y6E+	FDAC	00000010	FDNRDY1	01177A+	FSTPRT15	00000003	KBCQAL	0000007F
DSINIT1	010Y92+	FDCCHDR	00000010	FDNRDY2	011792+	FSTPRT3M	00000000	KBCSHT	0000000E
DSINIT2	010Y9A+	FDCCRD	011594+	FDOKSTA	0117D2+	FSTPRT6M	00000001	KBDSINT	00000700
DSKREAD	00000033	FDCCRD1	0115AC+	FDRCDBR	00000004	FUPDTRK	00000010	KBFCLOS	00000002
DSKWRIT	00000032	FDCCRD2	0115B0+	FDRCRD	00000004	FVERIFY	00000004	KBFCNTL	00000064
DSMAXH	0000002CF	FDCCRD3	0115B8+	FDRCCK	00000004	GAVEUP	00000080	KBFEMTY	00000001
DSMAXIV	00000022D	FDCCRD4	0115C2+	FDRDSTA	01178C+	GDATA	00000016	KBFULL	00000000
DSMAXYH	00000022F	FDCCRD5	0115CA+	FDRDWR	011528+	GOT0BT	010496+	KBFLOCK	00000005
DSMAXYV	0000002CF	FDCCRD6	0115D6+	FDRDWR1	011542+	GOT0BT1	010498+	KBFSHFT	00000003
DSNITST	010YFE+	FDCCRST	011734+	FDRDWR9	011554+	GOT0BT2	0104A4+	KBGCRR1	0107CC+
DSPBASE	00080000	FDCCSIN	011744+	FDRST	011716+	GOT0BT3	0104B4+	KBGCRR2	0107E8+
DSPEND	0008E000	FDCCSK	011754+	FDRST0	01171E+	GRAPHIC	00000001	KBGCRR3	010800+
DSPLEN	0008E000	FDCCSK1	01175C+	FDRST1	011722+	HDRERR	00000083	KBGTCH	0107C0+
DSPST1	010YCA+	FDCCSK2	011764+	FDRST2	01172E+	INCTEST	010544+	KBGETKY	010808+
DSPST9	010YD2+	FDCCSOT	01174C+	FDRSTV	01170C+	INITOP	00000020	KBINIT	010722+
DSPUTCH	010YD8+	FDCCSTP	01173C+	FDRWST	01179C+	INTDC0	010604+	KBINIT1	01076E+
DSPUTST	010YCE+	FDGDATR	00000016	FDRWSTA	011790+	INTDC1	01061E+	KBINTR	010780+
DSRESET	010A04+	FDCLCT8	011650+	FDSECI0	011464+	INTKYBD	0105E8+	KBINTR1	0107A0+
DSRTRN	010B34+	FDCLCT9	011654+	FDSECR	01155A+	INTLVL7	0105E6+	KBINTR2	0107AE+
DSRTRN1	010B38+	FDCLCTS	01160C+	FDSECR1	011564+	INTOMN1	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+	FDCESEK	00000015	FDSECR9	01158C+	INUSE	00000085	KBPRO2	010856+
DSSHOW6	010AC0+	FDCESTP	00000035	FDSECRW	01155A+	INVCURS	00000003	KBPRO3	01085A+
DSSHOW7	010ACA+	FDCESTPIN	00000055	FDSEEK	0116DA+	IOBEEPQ	00030F71	KBPRO9	01085C+

KBPROXY	010818+	LDSYNC1	01108E+	NIRQ2	00000005	RAMKBBUF	00000300	SBSTROB	011220+
KBPUT1	010872+	LDSYNC2	011078+	NIRQ3	00000006	RAMKBLFN	00000100	SBUSER	010454+
KBPUT2	01087C+	LDSYNC3	0110AA+	NIRQ4	00000007	RAMLEN	00001000	SBUSY	00000000
KBPUTCH	01085E+	LDSYNC5	0110B6+	NNM11	00000000	RAMMBUC	00000400	SBW1	01122C+
KBQUAL	010884+	LDSYNC6	0116BA+	NNM12	00000001	RAMSIZ7	01058C+	SBWAIT	011228+
KBQUAL1	01088E+	LDSYNC9	0110C4+	NNM13	00000002	RAMSIZ8	010564+	SBWEIT	01123C+
KBQUAL2	010898+	LDWAIT	011016+	NNM14	00000003	RAMTST1	0101C2+	SC10	01128C+
KBQUAL3	0108A0+	LDWAIT1	01101A+	NORVFR	00000092	RAMTST2	0101E4+	SC12	01129C+
KBQUAL8	0108AA+	LDWAIT2	011024+	NOSCROLL	00000005	RAMWKSTA	00000700	SC20	0112CC+
KBQUAL9	0108AC+	LDW101	010FAE+	NOSOCKT	00000082	RBDCLK	FFFFFFFF	SC30	0112F6+
KBRCMD	0030F05	LNBUF2	00000054	NOTRANS	00000090	RBDOPCO	FFFFFFFF	SC32	0112FC+
KBRCNTL	0030F07	LONGCMS2	011354+	OBOOT	0110CC+	RBDUNT	FFFFFFFF	SC40	01130A+
KBRDATA	0030F01	LS152SD	00000004	ODBLK1	01115C+	REBUSY	FFFFFFFF	SC50	01131E+
KBRSTAT	0030F03	LS81NMN	00000005	ODBLK2	011170+	RDYADR	0000077F	SC60	01132C+
KBRTABLE	01090E+	LSDRQ	00000000	ODBLK3	011180+	RECVOP	00000000	SC70	01133A+
KBSTABLE	0108AE+	LSDSKCHG	00000004	ODBLK10	011130+	RERRUNW	FFFFFFFFC0	SCERR1	011344+
LB00T	010EFA+	LSFMFMF	00000007	ODCMD1	011114+	RETSKT	000000A0	SCERR2	01134A+
LB00T10	010F02+	LSJNT	00000001	ODCMD2	011126+	RG00D	00000000	SCERR3	01134E+
LB00T30	010F18+	LSSTR	00000000	ODCMD9	01112A+	RHDR	00000004	SCEXIT	011352+
LB00T80	010F30+	MARCH	010518+	ODCOMND	0110E6+	RHDSKLN	0000000A	SCM02	0113DC+
LB00T90	010FE+	MEMCLR	010214+	ODDSK1	0111A0+	RHDSKRC	0000000C	SCM03	0113EE+
LC81NMN	00000006	MEMTEST	010204+	ODDSK2	011186+	RHPKTLN	00000008	SCM04	011400+
LC0E0	00000001	MOTOROFF	00000010	ODDSK3	0111C4+	RHPKTRC	00000006	SCPTSSD	00000010
LC0E1	00000004	MOTORON	00000012	ODDSK4	0111E6+	RHSOR	00000007	SCPT8DD	0000001A
LC0FLP81N	00000006	MR1	01051C+	ODDSK5	0111EA+	RHWRRR	FFFFFFFFC0	SCPT8SD	0000001A
LC0FLPSD1	00000000	MR2	010526+	ODDSK6	0111F0+	RLOSTDEV	FFFFFFFFB0	SCRCERR	00000003
LC0FMFMF	00000007	MR3	010536+	ODDSK9	0111F2+	RNOTRDY	FFFFFFFFC0	SDRO	00000001
LCMD1	011362+	MRERR	010542+	ODDSK10	011190+	ROMBASE	00010000	SOTOVER	00000002
LCMD3	011370+	MSC1	01045A+	ODDW	00000024	ROMEND	00012000	SOTUNDR	00000002
LCMD4	01138A+	MSC10	0104B2+	ODDWH1	00000025	ROMLN	00002000	SELBOOT	0103C0+
LCMD5	011392+	MSC11	0104D1+	ODDWLO	00000026	ROMTST	0104D4+	SENDOP	00000040
LCMD6	01139E+	MSC12	0104D9+	ODVALID	0000002C	ROMTST1	0101A6+	SETG0	01123E+
LCMD7	0113AC+	MSC13	0104DF+	ODWRAD	00000028	RPTST1	010376+	SETG01	011262+
LCMDERR	0113BE+	MSC19	0104E6+	OFF	00000000	RPTST2	010396+	SETINTV	010244+
LCMDEX	0113C6+	MSC2	010480+	ON	00000001	RPTST8	01039E+	SETM8	010232+
LCMDOK	0113C2+	MSC3	010483+	PHASEOFF	00000000	RPTST9	0103B6+	SETRECV	011252+
LCMDR	00000000	MSC30	0104F2+	PHASEON	00000002	RPTSTAT	010366+	SETUP	0100B4+
LCMOTOR0	00000005	MSC31	0104FF+	PHASE1OF	00000004	RNF	FFFFFFFFD0	SETUP1	0100C2+
LDLKL10	010F8C+	MSC32	010708+	PHASE1ON	00000006	RSEKERR	FFFFFFFFE0	SETUP2	0100C6+
LDOSK1	01104E+	MSC4	010695+	PHASE2OF	00000008	RT1	0104D8+	SETUP3	0100CA+
LDOSK1A	011054+	MSCCPY	010090+	PHASE2ON	0000000A	RT1ERR	0101DA+	SHLDD	00000005
LDOSK2	01105C+	MIBBASE	00020000	PHASE3OF	0000000C	RT1ERR	0101FA+	SHDR	0000000E
LDOSK3	011064+	MIBEND	00022000	PHASE3ON	0000000E	RWRPROT	FFFFFFFFF0	SHFMLN	00000014
LDOSK9	01107A+	MIBENTRY	00020008	PHASEOFF	00000000	SBBOOT	010446+	SHORTCMD	0113CA+
LDOSK10	011036+	MIBINIT	00020004	PHASEON	00000002	SBDEBUC	0103DC+	SHPKTRC	0000000E
LDGETBB	010F70+	MIBLEN	00002000	PRIOR	00000001	SBERR	011214+	SHOLN	00000012
LDGETBX	010F8A+	NBLKSSD	00000118	PROMLEVL	00000006	SBEXIT	011218+	SINDEX	00000001
LDRI01	010FC6+	NBLK8DD	000003E9	PROMVERS	00000000	SBFLPY	010414+	SLOT1AD	00030001
LDRI03	010FC0+	NBLK8SD	000001F4	Q6H	0000001A	SBFLPY1	010422+	SLOTADR	010648+
LDSTRN	010FE2+	NBUF1	00000D00	Q6L	00000018	SBFLPY2	01043C+	SLOTID	010278+
LDSEND	010FEC+	NBUF2	00000E02	Q7H	0000001E	SBFLPY3	010442+	SLOTID1	010286+
LDSEND0	010FFA+	NDEVIAD	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	TCDTALM	00000020	USRBASE	0008E000	WRBITOFS	0000001A
SLOTIDA	010340+	SRAMWT	000000C0	TCHDRLM	00000022	VERT	00000000	WRCHARPT	00000000
SLOTIDB	01035A+	SRECTYP	00000005	TCMD	00000018	VIABASE	00030F00	WRCURADR	00000008
SLOTPTR	00030100	SRNF	00000004	TCOP	00000018	WAITING	000000FF	WRCURSI	00000016
SLTADDF5	00000200	SSEEXERR	00000004	TCRADHI	00000019	WALKBIT	0104E6+	WRCURSY	00000018
SLTSTAD	00030A01	STRADR	00030FA1	TCRADLO	0000001A	WB1	0104F0+	WRGRORCI	0000001C
SNDCMDS	0112D8+	STRK0	00000002	TC SOCK	0000001C	WB2	0104F4+	WRGRORCY	0000001E
SNDREST	0112AE+	STROBIT	0111FA+	TIMEOUT	00000091	WB3	010502+	WRHOMEOF	0000000C
SNOTRDY	00000007	SUI1	010252+	TKPSSSD	00000023	WBERR	010516+	WRHOMEPT	00000004
SRAMLN4	00000014	SVLCMD	00000000	TKPS8DD	0000004D	WHOOP	00000001	WRLENGTH	00000024
SRAMLN5	0000001C	SWRFAULT	00000005	TKPS8SD	0000004D	WRAPON	00000004	WRLENGTHI	00000012
SRAMLN6	00000020	SWRPROT	00000006	TOINTVL	0000FFFE	WRATTR1	00000020	WRLENGTHY	00000014
SRAMLN7	00000022	TCADAH1	0000001D	TOOLONG	00000081	WRATTR2	00000021	WRRCOLEN	00000023
SRAMRD4	00000000	TCADALO	0000001E	TRKCNT	00000000	WRBASEX	0000000E	WRSTATE	00000022
SRAMRD5	00000040	TCDEST	00000023	TRKN	00000002	WRBASEY	00000010	ZERORAM	01058E+

0 errors. 4785 lines.





## Cross-Reference Listing

File ID CC FROM TEXT

ABOOT	618	3724*								
ABOOT1	3739*									
ABOOT90	3737	3745	3747	3751*						
ADGON	3966*	3971	3974	3977	3980	3983	3986	3989	3997	3999
ADBIO1	3785	3789*								
ADBSIG	3787	3792*								
ADBLK10	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*							
ADPSEC	4191	4417*								
ADPSEC1	4422*	4431								
ADPSEC2	4424	4431*								
ADPSEC8	4435*									
ADPSEC9	4430	4437*								
ADFSNRV	4415*	4420								
ADISSSSD	3735	4016*								
AGILVTB	4028	4034*								
ADINIT	258	3736	4042*							
ADINLV	4167*	4182								
ADINLV9	4169	4171*								
ADMTR0F	3790	3825	4077	4105*						
ADMTRGN	3820	4075	4111*	4124						
ADRCAD	4465*	4467								
ADRCRD	4226*	4233								
ADRCSC	4179*	4187								
ADRCSK	4571*	4575								
ADDR4	4276	4337*	4338	4341						
ADDR4B	3834	4336*								
ADDR4E	3834	4344*								
ADDR5	4297	4306	4315	4350*	4357					
ADDR5B	3835	4349*								
ADDR5E	3835	4361*								
ADDR6	4284	4367*								
ADDR61	4368*	4369	4371							
ADDR62	4375*	4376								

ADDR69	4379	4382*				
ADDR6B	3836	4366*				
ADDR6E	3836	4384*				
ADDRAD	4422	4467*	4580			
ADDRAD1	4469*	4477				
ADDRAD2	4449	4474*	4475			
ADDRAD3	4476*	4484	4491			
ADDRAD3A	4480*	4482				
ADDRAD3B	4487*	4489				
ADDRAD4A	4500*	4502				
ADDRAD4B	4511*	4513				
ADDRAD4C	4522*	4524				
ADDRAD4D	4533*	4535				
ADDRAD5A	4504*	4506				
ADDRAD5B	4515*	4517				
ADDRAD5C	4524*	4528				
ADDRAD5D	4537*	4539				
ADDRAD6	4549*	4550				
ADDRAD7	4554*	4557				
ADDRAD8	4470	4545	4552	4559	4562*	
ADDRWR	3789	4115*				
ADDRWR1	4150*	4155				
ADDRWR9	4125	4148	4151	4153	4157*	
ADRST	4076	4617*				
ADSC101	3612	3815*				
ADSC109	3822	3825*				
ADSEC10	257	3729	3811*			
ADSECR	3823	4152	4181*			
ADSECR1	4190*	4194				
ADSECR2	4192	4197*				
ADSECR9	4195	4199*				
ADSEK	3821	4147	4573*			
ADSEK1	4622	4648	4657*			
ADSETPH	4752	4767*				
ADSI01	4633	4637*				
ADSI02	4643	4646*				
ADSI03	4645	4648*				
ADSK1	4575*	4605	4608			
ADSK2	4578*	4587				
ADSK3	4584	4593*				
ADSK4	4602	4607*				
ADSK9	4588	4601	4610*			
ADTBLOF	4760	4780*				
ADTBLOM	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	4600		
AMVOL	3877*	3878				
APLSVAR	3858*	3873	4593	4618	4639	4684
BADDEST	2280*					
BADSOCK	2278*					
BASERAM	2850*					
BLKSZ	2786	2853*	3397	3782		

BPSS1SD	2885*	4022						
BFS01DD	2884*	3058						
BFS01SD	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					
CPABLK10	179*							
CPAINIT	181*							
CPASCTIO	180*							
CPBLK10	113*	1951	2305	2724	3728			
CPBTSLOT	100*	680	687	1948	1967	2303	2720	3724
CPBTSVR	101*	529	689	1949	1964			
CPCKSUM	142*	368						
CPDISKRC	109*							
CPDSCVUC	189*							
CPDSINIT	186*							
CPDSK10	114*	1953	2307	2726	3730			
CPDSPFLG	131*	1598						
CPDSPUTC	187*							
CPDSPUTS	188*							
CPEITCRT	148*	311						
CPFBLK10	176*							
CPFBOOT	175*							
CPFBOOTJ	174*							
CPFBFS	122*	3044	3058	3395	4022	4282	4313	
CPFDV52	121*	3043	3057	3133	4021	4121		
CPFINIT	178*							
CPFINLV	120*	4029	4168					
CPFOFST	124*	3062	3406	3438	3444	4024		
CPFSCCTIO	177*							
CPFSPD	125*	3061	3414	4025				
CPFSPT	123*	3045	3059	3402	3437	4023	4418	
CPFTFS	124*	3046	3060	3410	3446	4024		
CPFTYP	127*	3447	4027					
CFIOBUF	146*	3853						
CP1STACK	150*	301						
CP1VEC1	192*	436						
CP1VEC2	193*							
CP1VEC3	194*							
CP1VEC4	195*							
CP1VECS	196*							
CP1VEC6	197*							
CP1VEC7	198*							
CPKBGETC	184*							
CPKBINIT	183*							
CPBLK10	171*							
CP1BOOT	170*							
CP1BOOTJ	169*							
CP1DSK10	172*							
CP1BLK10	166*							
CP1BOOT	165*							
CP1BOOTJ	164*							
CP1DSK10	167*							
CPOMNIBF	152*	2320	2359	2401				











FDCRD	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	5450*								
FDSECC10	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	3524	3533*						
FDSEEK9	3515	3518	3535*						
FDS101	2824	2827*							
FDS102	2837*								
FDS109	2835	2839*							
FDSKST1	3464	3668*							
FDSKSTA	3458	3517	3531	3555	3666*				
FDSWSTA	3547	3661*							
FDTMOH1	3485*	3491							
FDTMOLO	3486*	3490	3624						
FDWRDY	3208	3455	3488*	3514	3522	3525	3559		
FDWRDY1	3493*	3494	3495						
FDWRDY9	3494	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				
FUPDTRK	2990*	3573	3574	3575					
FVERIFY	2989*	3572	3573	3574	3575	3576			
GAVEUP	2274*								
GDATA	2239*	2487	2421	2423					
GOTOBT	616	619	671*						

GOTGBT1	610	673*				
GOTGBT2	671	677*				
GOTGBT3	679	682*				
GRAPHIC	1367*					
HCERRR	2277*					
INCTEST	406	785*				
INITOP	506	2250*				
INTDC6	275	876*				
INTDC1	273	889*				
INTKBC6	277	863*				
INTLVL7	278	858*				
INTOMNI	274	885*				
INTSLOT	272	857*				
INTTIMR	276	870*				
INUSE	2279*					
INVCURS	1369*					
IGBEEFFQ	204*					
IOBOOTSW	203*	570	1386			
IOBASE	76*	77				
IT01	788*	791				
IT02	796*	800				
IT99	797	802*				
IWLVL1	88*	437				
IWLVL2	89*					
IWLVL3	90*					
IWLVL4	91*					
IWLVL5	92*					
IWLVL6	93*	1025				
IWLVL7	94*					
JUMPTO	212*	239	245	251		
KBBBUPR	956*	957	1020	1112	1117	1213 1216
KBBFLGS	952*	953	1014	1059	1101	
KBBFRMT	953*	954	1021	1110	1226	
KBBLEN	957*	1112	1213			
KBBREAR	954*	955	1124	1267		
KBBSRSV	955*	956	1119			
KBCC400	998*	1031				
KBCCBRK	999*	1032				
KBCCGG	1000*	1033				
KBCCNTL	976*	1239				
KBCCOFF	997*	1013				
KBCLCA	982*	1177				
KBCLCZ	983*	1179				
KBCLOCK	977*	1244				
KBCNOCH	978*	1076				
KBCQMRK	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	
KBFULL	961*	1184	1222			
KBFLOCK	966*	1175	1246			
KBFSHFT	964*	1069	1236			
KBGCHR1	1105*	1106				
KBGCHR2	1116	1119*				
KBGCHR3	1125	1128*				

KBGETCH	161	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*											
KBPRO2	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												
KBADATA	990*	1030	1145											
KBSTAT	991*	1029	1144											
KBRTABLE	1071	1287*												
KBSTABLE	1068	1181	1264*											
LBBOOT	596	1936*												
LBBOOT10	1939*	1944												
LBBOOT30	1941	1948*												
LBBOOT80	1963*	2308												
LBBOOT90	1946	1965	1973	1977	1981	1985	1988*							
LCBINMIN	2947*													
LCDE0	2944*													
LCDE1	2945*													
LCFLF01N	2948*	3114												
LCFLFSD1	2943*	3416	3462											
LCFMFM	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
LCMDEI	2663	2667*												
LCNDOX	2657	2665*	2700	2703										
LCMDR	2931*	3100	3122	3463										
LCMOTORO	2946*	3099	3113											
LDBLK10	247	1950	2024*											
LDDSK1	2144*	2146												
LDDSK1A	2143	2146*												
LDDSK2	2137	2152*												
LDDSK3	2156*	2157	2161											
LDDSK9	2148	2159	2163*											
LDDSK10	248	1952	2132*											

LDGETBB	1972	1976	1980	1984	1990*
LDGETBI	1998	2001*			
LDRI01	2035	2052*			
LDRI03	2055*	2056	2060		
LDTRN	2046	2058	2062*		
LDSEND	2025	2031	2033	2077*	2078 2145
LDSEND0	2090*	2096			
LDSEND1	2027	2093*	2141		
LDSYNC	480	2176*			
LDSYNC1	2182*	2187	2190		
LDSYNC2	2184*	2184			
LDSYNC3	2186	2190*	2192		
LDSYNC5	2188	2194*			
LDSYNC6	2191	2197*			
LDSYNC9	2195	2201*			
LDWAIT	2044	2052	2106*	2151	
LDWAIT1	2108*	2108			
LDWAIT2	2110	2113*	2114	2116	
LDWIG1	2040*	2041	2043		
LNBUF1	4011*	4275	4296	4305	4314 4314
LONGCMDS	2373	2425	2599*	2622	
LS1SD2SD	2736*				
LS81NMIN	2937*				
LSORQ	2934*	3292			
LSBSKCHG	2938*				
LSFMFM	2939*				
LSINT	2935*	3295	3627		
LSTRR	2931*	3291	3627		
MARCH	384	397	751*		
MEMCLR	407	415*			
MEMTEST	398	404*			
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	936*			
MSG13	599	931*			
MSG19	622	932*			
MSG2	449	558	925*		
MSG3	673	924*			
MSG30	546	933*			
MSG31	551	934*			
MSG32	540	935*			
MSG4	584	927*			
MSGCPY	282*	447			
MXBASE	70*	72	73	74	428 581
MXBEND	72*				
MXBENTRY	74*	586			
MXBINIT	73*	427	580		
MXBLEM	71*	72			
NBLKSSD	2963*	4021			
NBLK0DD	2902*	3057			
NBLK0SD	2901*	3043			





SBMSG	590	595	600	621*
SBOMNI	240	576	589*	642
SBSTROB	2458	2460	2462	2473*
SBUSER	572	583	626*	675
SBUSY	3012*	3286	3455	3672
SBW1	2476*	2478		
SBWAIT	2464	2475*		
SBWEXIT	2477	2479*		
SC10	2511*	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*		
SCPTSSD	2891*	4023		
SCPT8DD	2890*	3059		
SCPT8SD	2889*	3045		
SCRCERR	3021*	3213	3650	3670
SDRG	3015*			
SDTGVER	3018*	3211		
SDTUNDR	3019*			
SELBOOT	570*			
SENDOP	2289*	2572		
SETGO	2485*	1603	2629	
SETGO1	2492	2505*		
SETINTV	436	436*		
SETMB	427*			
SETRECV	2497*	1637	2658	2675 2704
SETUP	233	300*	443	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	2136*	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*		





