```
LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE('ASSEMBLY LANGUAGE BOOTSTRAP PROGRAM')
                             2           NAME    BCOT1_A
                             3     ;CB$bootstrap:
                             4 +1  $include(:f1:propa.lit)
                       =1    5     ;
                       =1    6     ;   Intel Corporation Proprietary Information. This listing is
                       =1    7     ;   supplied under the terms of a license agreement with Intel
                       =1    8     ;   Corporation and may not be copied nor disclosed except in
                       =1    9     ;   accordance with the terms of the agreement.
                       =1   10     ;
                            11
                            12
                            13     DGROUP  GROUP   DATA
----                        14     DATA    SEGMENT PUBLIC 'DATA'
                            15
                            16             EXTRN   CQMIPIDSBASES:NEAR,CQMIPDEVICEINFO:NEAR,CQTHISDEVICE:BYTE
                            17             EXTRN   CMD:BYTE,CMDBLOCKPTR:NEAR,CMDBLOCKADDRESSES:NEAR
                            18             EXTRN   CQREMOTEWAITINGMB:NEAR,DLLBUFPTRO:NEAR
                            19             EXTRN   CQBOOTCMDMB:NEAR,LOCALBOOTCMDMSG:NEAR,CTRESULT:BYTE
                            20             EXTRN   NMLENTRY:NEAR,MIPDEFPTR:NEAR
                            21             EXTRN   CQDLLTXFREEMBX:NEAR,TRIES:BYTE,TEMPBUF:NEAR
                            22             EXTRN   CQMIPDEVCNT:BYTE,CQMIPDEVTOENTRY:NEAR
----                        23     DATA    ENDS
                            24
                            25     CGROUP  GROUP   CODE
----                        26     CODE    SEGMENT PUBLIC 'CODE'
                            27
                            28             EXTRN   CQRAMTEST:NEAR,CQDEVICETEST:NEAR,CQCBINIT:NEAR,CQHDWINIT:NEAR
                            29             EXTRN   CQMIPHALT:NEAR,CQMIPINIT:NEAR,CQMIPINTASK:NEAR,EDLSTART:NEAR
                            30             EXTRN   CQDLLCONNECT:NEAR,CQDLLTXSEND:NEAR,CQDLLRXRETBUF:NEAR
                            31             EXTRN   CQISEND:NEAR,CQRECEIVE:NEAR
                            32
                            33             ASSUME  CS:CGROUP,DS:DGROUP
                            34
0000 FD3F                   35     RANRTPTR DW     03FFDH,0F000H
0002 00F0
                            36
                            37     ;/*
                            38     ;  declare external variables. These will all be in other CB firmware
                            39     ;  modules
                            40     ;  for Mip first
                            41     ;*/
                            42     ;declare CQ$MIP$Ids$bases (Mip$idss) structure (
                            43     ;          base byte,
                            44     ;          length byte)  external,
                            45     ;
                            46     ;        CQ$MIP$Device$info (Mip$devices) structure(
                            47     ;          devid byte,
                            48     ;          Status byte,
                            49     ;          RQDin pointer,
```

```
LOC  OBJ                  LINE     SOURCE

                            50     ;           RQDout pointer,
                            51     ;           Int$type byte,
                            52     ;           Time$to$wait byte,
                            53     ;           Int$adr word ) external,
                            54     ;         CQ$Thisdevice byte external;
                            55
                            56     ;declare /*
                            57     ;           declare structures to get command block address
                            58     ;         */
                            59     ;         Cmd$block$ptr pointer,
                            60     ;         CBptr structure (off word, base word) at (@Cmd$block$ptr),
                            61     ;         Cmd$block$ptr$o word at (@CBptr.off),
                            62     ;         Cmd$block$ptr$b word at (@CBptr.base),
                            63     ;         Cmd$block$addresses (8) word public
                            64     ;            data(0F69H,1F00H,2F00H,100H,800H,1000H,2000H,2F00H),
                            65     ;         /*
                            66     ;           declare variables for control purposes
                            67     ;         */
                            68     ;         First$boot$cmd byte  at (0F3FFFH),
                            69     ;         Miprunning boolean  at (0F3FFEH),
                            70     ;         Ran$RT boolean  at (0F3FFDH),
                            71     ;         /*
                            72     ;           declare local variables for various purposes
                            73     ;         */
                            74     ;         CTresult byte public ,
                            75     ;         Action byte,
                            76     ;         Cmd byte public,
                            77     ;         NML$entry word,
                            78     ;         Execution$SA word,
                            79     ;         Mip$def$ptr pointer,
                            80     ;         DLL$bufptr pointer,
                            81     ;         DLL$bufptr$o word at (@DLLbufptr),
                            82     ;         /*
                            83     ;           structure for Series IV reporting
                            84     ;         */
                            85     ;         S4 based Cmd$block$ptr structure(Deviceid byte,Result byte ),
                            86     ;         Status$report structure (
                            87     ;           done byte,
                            88     ;           semaphore byte,
                            89     ;           result$blk$ptr word) at (2F000H);
                            90 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                            91     ;       /*
                            92     ;         declare based structures for local commands
                            93     ;       */
                            94     ;declare LL$area based Cmd$block$ptr structure
                            95     ;       ( Cmd byte,
                            96     ;         Response byte,
                            97     ;         From$area pointer,
                            98     ;         To$area pointer,
                            99     ;         Length word,
                           100     ;         Exec$SA word,
                           101     ;         Mip$def$area byte ),
                           102
                           103     ;       Mip$sizes based Mip$def$ptr structure
                           104     ;       ( Devcnt byte,
                           105     ;         Ids$cnt byte,
                           106     ;         This$dev byte,
                           107     ;         Rsrved byte,
                           108     ;         Mip$bases (8) word),
                           109
                           110     ;       Mip$dev$def based Mip$def$ptr (1) structure
                           111     ;       ( Dev$id byte,
                           112     ;         Status byte,
                           113     ;         RQD$to$CB pointer,
                           114     ;         RQD$from$CB pointer,
                           115     ;         Int$type byte,
                           116     ;         Time$to$wait byte,
                           117     ;         Int$adr word  );
                           118
                           119
                           120 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           121     ;/*
                           122     ;  declare some utility routines
                           123
                           124     ;  declare the routine to get the Cmd block address
                           125     ;*/
                           126     ;Load$Cmd$block$ptr:
                           127             PUBLIC  LOADCMDBLOCKPTR
C004                       128     LOADCMDBLOCKPTR:
                           129     ;  procedure public;
                           130     ;  Cmd$block$ptr$o = 0;
C004 33F6                  131             XOR     SI,SI
0006 8936000C        E     132             MOV     WORD PTR CMDBLOCKPTR,SI
                           133     ;  Cmd$block$ptr$b = Cmd$block$addresses(SHR(Input(PIO$PC) and 70H,4));
000A E4E2                  134             IN      AL,0E2H  check jumpers
000C 2470                  135             AND     AL,70H
000E B104                  136             MOV     CL,4H
0010 D2E8                  137             SHR     AL,CL  - right justify in byte
0012 B400                  138             MOV     AH,0H
0014 D1E0                  139             SHL     AX,1 (*2) [word offset]
0016 8BD8                  140             MOV     BX,AX
0018 2E8E870C00      E     141             MOV     ES,WORD PTR CS:CMDBLOCKADDRESSES[BX]  grab address from PL/M boot
001D 8C06020C        E     142             MOV     WORD PTR CMDBLOCKPTR+2,ES
                           143     ;  Cmd = LL$area.Cmd;
0021 268A04                144             MOV     AL,BYTE PTR ES:[SI]
0024 A20000          E     145             MOV     CMD,AL
                           146     ;end Load$Cmd$block$ptr;
0027 C3                    147             RET
                           148     ;/*
                           149     ;  declare procedures to load comm memory to/from host
                           150     ;*/
                           151     ;Local$move:
                           152             PUBLIC  LOCALMOVE
0028                       153     LOCALMOVE:
                           154     ;  procedure public;
                           155     ;  if LL$area.length = 0 then return;
                           156     ;  call Movb(LL$area.from$area,LL$area.To$area,LL$Area.Length);
0028 C41E0000        E     157             LES     BX,DWORD PTR CMDBLOCKPTR
002C 268B4F0A              158             MOV     CX,ES:[BX+0AH]          ; GET LENGTH
0030 0BC9                  159             OR      CX,CX
0032 7501                  160             JNZ     LM1
0034 C3                    161             RET
0035 06                    162     LM1:    PUSH    ES              ; SAVE FOR LATER
0036 26C47702              163             LES     SI,DWORD PTR ES:[BX+2H]
003A 8CDA                  164             MOV     DX,DS
003C 06                    165             PUSH    ES
003D 1F                    166             POP     DS
003E 07                    167             POP     ES
003F 26C47F06              168             LES     DI,DWORD PTR ES:[BX+6H]
0043 52                    169             PUSH    DX
0044 FC                    170             CLD
0045 F3                    171             REP MOVSB
0046 A4
0047 1F                    172             POP     DS      ; 1
                           173     ;end Local$move;
0048 C3                    174             RET
```

```
LOC  OBJ                  LINE     SOURCE

                           175     ;/*
                           176     ;  short form for DLL Send
                           177     ;*/
                           178     ;DLLsend:
                           179             PUBLIC  DLLSEND
0049                       180     DLLSEND:
                           181     ;  procedure;
                           182     ;  declare I byte;
                           183     ;  I =  CQ$DLL$connect(NML$type,.CQ$remote$waiting$mb);
0049 B80050                184             MOV     AX,5000H
004C 50                    185             PUSH    AX      ; 1
004D B80000          E     186             MOV     AX,OFFSET DGROUP:CQREMOTEWAITINGMB
0050 50                    187             PUSH    AX      ; 2
0051 E80000          E     188             CALL    CQDLLCONNECT
                           189     ;  call CQDLL$tx$send(DLL$bufptr$o);
0054 FF360000        E     190             PUSH    WORD PTR DLLBUFPTR0
0058 E80000          E     191             CALL    CQDLLTXSEND
                           192     ;end DLLsend;
005B C3                    193             RET
                           194     ;/*
                           195     ;  routine to save src adr and other info in a msg
                           196     ;*/
                           197             PUBLIC  SAVERCVDINFO
005C                       198     SAVERCVDINFO:
005C 8B1E0000        E     199             MOV     BX,WORD PTR DLLBUFPTR0
0060 8D770C                200             LEA     SI,WORD PTR [BX+0CH]
0063 BF0000          E     201             MOV     DI,OFFSET DGROUP:TEMPBUF
0066 B90F00                202             MOV     CX,0FH
0069 1E                    203             PUSH    DS
006A 07                    204             POP     ES
006B FC                    205             CLD
006C F3                    206             REP MOVSB
006D A4
                           207     ;       JMP     SHORT DLLRETBUF
                           208     ;/*
                           209     ;  short form for dll return buffer
                           210     ;*/
                           211     ;DLLretbuf:
                           212             PUBLIC  DLLRETBUF
006E                       213     DLLRETBUF:
                           214     ;  procedure;
                           215     ;  if DLLbufptr$0 <> 0 then call CQ$DLL$rx$retbuf(DLL$bufptr$o);
                           216     ;  DLL$bufptr$o = C;
006E 33C0                  217             XOR     AX,AX
0070 8706000C        E     218             XCHG    AX,WORD PTR DLLBUFPTR0
0074 08C0                  219             OR      AX,AX
0076 7404                  220             JZ      DLR81
0078 50                    221             PUSH    AX
0079 E80000          E     222             CALL    CQDLLRXRETBUF
                           223     ;end DLLretbuf;
007C C3                    224     DLR81:  RET
                           225     ;/*
                           226     ;  boot interrupt routine
                           227     ;*/
                           228     ;CQ$CA$int$routine:
```

```
LOC  OBJ                  LINE     SOURCE

                           229             PUBLIC  CQCAINTROUTINE
007D                       230     CQCAintroutine:
                           231     ;  procedure public;
                           232     ;  if Miprunning then call CQMIP$intask;
007D 2EC41E0C00      R     233             LES     BX,DWORD PTR RANRTPTR
0082 26807F01FF            234             CMP     BYTE PTR ES:[BX+1],0FFH
0087 7504                  235             JNE     @1
0089 E80000          E     236             CALL    CQMIPINTASK
008C C3                    237             RET
                           238     ;  else call CQ$Isend(.CQ$boot$cmd$mb,@Local$bootcmd$msg);
008D B80000          E     239     @1:     MOV     AX,OFFSET DGROUP:CQBOOTCMDMB
0090 50                    240             PUSH    AX      ; 1
0091 8D06000C        E     241             LEA     AX,WORD PTR LOCALBOOTCMDMSG
0095 1E                    242             PUSH    DS      ; 2
0096 50                    243             PUSH    AX      ; 3
0097 E80000          E     244             CALL    CQISEND
                           245     ;end CQ$CA$int$routine;
009A C3                    246             RET
                           247     ;/*
                           248     ;  set LED on or off depending on results of CT
                           249     ;*/
                           250     ;Set$LED:
                           251             PUBLIC  SETLED
009B                       252     SETLED:
                           253     ;  procedure;
                           254     ;  Output(PIOPC) = ((Ctresult <> 0) and 2H) or 5H;
009B 803E000C00      E     255             CMP     CTRESULT,0H
00A0 B007                  256             MOV     AL,07H
00A2 7502                  257             JNE     SETLED1
00A4 B005                  258             MOV     AL,5H
00A6                       259     SETLED1:
00A6 E6E2                  260             OUT     0E2H,AL
                           261     ;end Set$LED;
00A8 C3                    262             RET
                           263     ;/*
                           264     ;  routine for NML to use when present
                           265     ;*/
                           266     ;boot$register:
                           267             PUBLIC  BOOTREGISTER
00A9                       268     BOOTREGISTER:
                           269     ;  procedure(LW$o) public;
                           270     ;  declare LW$o word;
                           271     ;  NMLentry = LW$o;
00A9 5F                    272             POP     DI
00AA 58                    273             POP     AX
00AB A30000          E     274             MOV     WORD PTR NMLENTRY,AX
                           275     ;end boot$register;
00AE FFE7                  276             JMP     SHORT DI
                           277
                           278             PUBLIC  BUMPANDCHECKTRIES
00B0                       279     BUMPANDCHECKTRIES:
00B0 B0FF                  280             MOV     AL,0FFH
00B2 FE06000C        E     281             INC     TRIES
00B6 803E000003      E     282             CMP     TRIES,3
00BB 7401                  283             JE      $+3
```

only accept remote boot

accept local boot

LED ON

LED OFF (bit 1 is switch)

```
LOC  OBJ                  LINE      SOURCE

00BD 40                    284              INC     AX
00BE C3                    285              RET
                           286
                           287 +1   $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           288     ;/*
                           289     ;  first define DLL interface routine
                           290     ;*/
                           291             PUBLIC  TRANSMIT
00BF                       292     TRANSMIT:
                           293     ;Transmit:
                           294     ;  procedure(DA$ptr,Cmd,Info) public;
                           295     ;  declare DAptr pointer,
                           296     ;          Cmd byte,
                           297     ;          Info word,
                           298     ;          Msg based DLL$buf$ptr$o structure(
                           299     ;            Link pointer,
                           300     ;            P$length word,
                           301     ;            DA$1 pointer,
                           302     ;            DA$2 word,
                           303     ;            SA$1 pointer,
                           304     ;            SA$2 word,
                           305     ;            Type word,
                           306     ;            Cmd byte,
                           307     ;            Info word);
                           308     ;
                           309     ;  DLL$buf$ptr = CQ$Receive(CQ$DLL$tx$free$mbx);
00BF FF360000        E     310             PUSH    WORD PTR CQDLLTXFREEMBX
00C3 E80000          E     311             CALL    CQRECEIVE
00C6 891E0000        E     312             MOV     WORD PTR DLLBUFPTR0,BX
00CA 8C060200        E     313             MOV     WORD PTR DLLBUFPTR0+2,ES
                           314     ;  call movb(DAptr,@Msg.DA$1,6);
                           315     ;  Msg.P$length = 60;
                           316     ;  Msg.type = MNL$type;
                           317     ;  Msg.cmd = Cmd;
                           318     ;  Msg.Info = Info;
00CE 5A                    319             POP     DX                      ; RET ADR
00CF 58                    320             POP     AX                      ; INFO
00D0 894715                321             MOV     [BX+15H],AX
00D3 58                    322             POP     AX                      ; CMD
00D4 884714                323             MOV     BYTE PTR [BX+14H],AL
00D7 C747043C00            324             MOV     WORD PTR [BX+4],3CH             ; P$LENGTH
00DC C747125C00            325             MOV     WORD PTR [BX+12H],005CH         ; TYPE
00E1 5F                    326             POP     DI
00E2 07                    327             POP     ES                      ; DAPTR
00E3 268B05                328             MOV     AX,ES:[DI]
00E6 894706                329             MOV     [BX+6],AX
00E9 26C44502              330             LES     AX,DWORD PTR ES:[DI+2]
00ED 894708                331             MOV     [BX+8],AX
00F0 8C470A                332             MOV     [BX+10],ES
00F3 FFE2                  333             JMP     DX
                           334     ;end Transmit;
                           335 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           336     ;/*
                           337     ;  declare routine tto handle MIP things
                           338     ;*/
                           339     ;Handle$mip:
                           340             PUBLIC  HANDLEMIP
00F5                       341     HANDLEMIP:
                           342     ;  procedure;
                           343     ;  declare (I,J,K) byte;
                           344
                           345     ;    /*
                           346     ;      first load ids info
                           347     ;    */
                           348     ;    CQ$this$device = Mip$sizes.This$dev;
00F5 C41E000C        E     349     HMIP1:  LES     BX,DWORD PTR MIPDEFPTR
00F9 268A4702              350             MOV     AL,BYTE PTR ES:[BX+2]
00FD A20000          E     351             MOV     CQTHISDEVICE,AL
                           352     ;    K = Mipsizes.idscnt;
                           353     ;    J = Mip$sizes.devcnt;
0100 268B17                354             MOV     DX,ES:[BX]
                           355     ;    CQMIPdevcnt = J;
0103 88160000        E     356             MOV     CQMIPDEVCNT,DL
0107 B000                  357             MOV     AL,0H
                           358     ;    call Movb(@Mip$sizes.Mip$bases,@CQ$Mip$ids$bases,K*2);
0109 8D3E000C        E     359             LEA     DI,WORD PTR CQMIPIDSBASES
010D 8D5F04                360             LEA     BX,WORD PTR ES:[BX+4]
0110 8ACE                  361             MOV     CL,DH
0112 B500                  362             MOV     CH,0            ; GET K*2
0114 03C9                  363             ADD     CX,CX
0116 8BF3                  364             MOV     SI,BX
0118 06                    365             PUSH    ES
0119 1E                    366             PUSH    DS
011A 06                    367             PUSH    ES
011B 1E                    368             PUSH    DS
011C 07                    369             POP     ES
011D 1F                    370             POP     DS
011E FC                    371             CLD
011F F3                    372             REP     MOVSB
0120 A4
0121 1F                    373             POP     DS
0122 07                    374             POP     ES
                           375     ;    /*
                           376     ;      now load device info
                           377     ;    */
                           378     ;    Mip$def$ptr = @Mip$sizes.Mip$bases(K);
                           379     ;    I=0;
0123 B600                  380             MOV     DH,0
0125 B700                  381             MOV     BH,0
0127 8D3E000C        E     382             LEA     DI,WORD PTR CQMIPDEVICEINFO
                           383     ;    do while (I < J);
012B 3AD6                  384     @56:    CMP     DL,DH
012D 7419                  385             JE      @57
                           386     ;    CQMIPdevtoentry(Mipdevdef(i).devid) = I;
012F 268A1C                387             MOV     BL,ES:[SI]
0132 88870000        E     388             MOV     BYTE PTR CQMIPDEVTOENTRY[BX],DH
                           389     ;      call Movb(@Mip$dev$def(I).Status, @CQ$Mip$device$info(I),14);
```

```
LOC  OBJ                  LINE     SOURCE

0136 B90E00                390              MOV     CX,14
0139 06                    391              PUSH    ES
013A 1E                    392              PUSH    DS
013B 06                    393              PUSH    ES
013C 1E                    394              PUSH    DS
013D 07                    395              POP     ES
013E 1F                    396              POP     DS
013F FC                    397              CLD
0140 F3                    398              REP     MOVSB
0141 A4
0142 1F                    399              POP     DS
0143 07                    400              POP     ES
                           401      ;       I = I + 1;
0144 FEC6                  402              INC     DH
                           403      ;    end;
0146 EBE3                  404              JMP     @56
0148                       405      @57:
                           406      ;    call CQ$Mip$init;
0148 E80000          E     407              CALL    CQMIPINIT
                           408      ;    Miprunning = TRUE;
014B 2EC41E0C00      R     409              LES     BX,DWORD PTR RANRTPTR
0150 26C64701FF            410              MOV     BYTE PTR ES:[BX+1],0FFH
                           411      ;    call EDLSTART;
0155 E80000          E     412              CALL    EDLSTART
                           413      ;  end;
                           414      ;end Handle$mip;
0158 C3                    415              RET
                           416
                           417 +1   $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           418     ;/*
                           419     ;  boot main program
                           420     ;*/
                           421     ;declare  Boot$DMT$entry label public,
                           422     ;          Boot$restart label;
                           423     ;  disable;
                           424     ;  call CQ$Hdw$init;
                           425     ;  /*
                           426     ;    if TP 10 is grounded then run confidence tests forever
                           427     ;  */
                           428             PUBLIC  BOOTSTARTENTRY
0159                       429     BOOTSTARTENTRY:
0159 FA                    430             CLI
                           431     ;
                           432     ; NOW INITIALIZE DS,SS, AND SP
                           433     ;
015A B8----          R     434             MOV     AX,DGROUP
015D 8ED8                  435             MOV     DS,AX
015F B8F0F9                436             MOV     AX,0F9F0H
0162 8ED0                  437             MOV     SS,AX
0164 BC7000                438             MOV     SP,70H          ; FOR INIT, USE AREA IN STATIC RAM
0167 C606000C00      E     439             MOV     CTRESULT,0      ; ASSUME HARDWARE HAS NO PROBLEMS
                           440
                           441     ;  do while (input(PIO$PC) and 80H) = 0;
016C                       442     @61:
016C E4E2                  443             IN      AL,0E2H
016E E80000          E     444             CALL    CQDEVICETEST    do PUCONF tests on devices
0171 A20000          E     445             MOV     CTRESULT,AL     ; result is in both AL and AH
0174 E4E2                  446             IN      AL,0E2H
0176 A880                  447             TEST    AL,80H          if jumper plug, loop infinitely (or until jumper removed)
0178 750C                  448             JNZ     @62
                           449     ;    CTresult = CQdevice$test or CQ$ramtest;
017A 0AE4                  450             OR      AH,AH           ; IF PROBLEM WITH DEVICE TEST, SKIP RAM TEST
017C 75EE                  451             JNZ     @61
017E E80000          E     452             CALL    CQRAMTEST       do PUCONF on RAM
0181 A20000          E     453             MOV     CTRESULT,AL
                           454     ;    call Set$LED;
                           455     ;  end;
0184 EBE6                  456             JMP     @61
0186                       457     @62:
                           458     ;  CTresult = CQ$device$test;
                           459     ;  /*
                           460     ;    if in Series IV then report the result
                           461     ;  */
                           462     ;  call CQhdwinit;
0186 E80000          E     463             CALL    CQHDWINIT
                           464     ;  if (input(PIO$PC) and 70H) = 10H then
0189 E4E2                  465             IN      AL,0E2H
018B 2470                  466             AND     AL,70H
018D 3C20                  467             CMP     AL,20H          (E26-E27 unjumped) [reserved in 550 book]
018F 7543                  468             JNE     @49                 series IV?
                           469     ;  do;
                           470     ;    do while Status$report.done <> 99H; ; end;
0191                       471     @63:
0191 BB00F0                472             MOV     BX,0F000H
```

figure out where command block is

```
LOC  OBJ                  LINE     SOURCE

0194 B80020                473              MOV     AX,200CH
0197 8EC0                  474              MOV     ES,AX
0199 26803F99              475              CMP     BYTE PTR ES:[BX],99H
019D 75F2                  476              JNE     @63
                           477      ;L1: do while Status$report.semaphore <> 0; ; end;
019F                       478      L1:
019F B000                  479              MOV     AL,0
01A1 8D7F01                480              LEA     DI,WORD PTR ES:[BX+1]
01A4 263805                481              CMP     BYTE PTR ES:[DI],AL
01A7 75F6                  482              JNE     L1
                           483      ;       output(PIO$PC) = 0H;
01A9 E6E2                  484              OUT     0E2H,AL
                           485      ;       if Status$report.semaphore <> 0 then
01AB 263805                486              CMP     BYTE PTR ES:[DI],AL
01AE B004                  487              MOV     AL,4            ; TO LOWER BUS LOCK
01B0 7404                  488              JE      @50
                           489      ;       do;
                           490      ;         output(PIO$PC) = 4;
01B2 E6E2                  491              OUT     0E2H,AL         ;
                           492      ;         go to L1;
01B4 EBE9                  493              JMP     L1
                           494      ;       end;
01B6                       495      @50:
                           496      ;       Status$report.semaphore = 1;
01B6 26C60501              497              MOV     BYTE PTR ES:[DI],1
                           498      ;       output(PIO$PC) = 4;
01BA E6E2                  499              OUT     0E2H,AL
                           500      ;       Cmd$block$ptr$o = Status$report.result$blk$ptr;
                           501      ;       Cmd$block$ptr$b = 2000H;
01BC 268B7702              502              MOV     SI,WORD PTR ES:[BX+2]
                           503      ;       S4.deviceid = 2;
01C0 26C60402              504              MOV     BYTE PTR ES:[SI],2
01C4 46                    505              INC     SI
                           506      ;       S4.result = CTresult;
01C5 A00000          E     507              MOV     AL,CTRESULT
01C8 268804                508              MOV     BYTE PTR ES:[SI],AL
                           509      ;       Status$report.result$blk$ptr = Status$report.result$blk$ptr + 2;
01CB 46                    510              INC     SI
01CC 26897702              511              MOV     WORD PTR ES:[BX+2],SI
                           512      ;       Status$report.semaphore = 0;
01D0 26C60500              513              MOV     BYTE PTR ES:[DI],0
                           514      ;     end;
                           515      ;     /*
                           516      ;       set the mask for the CA interrupt only
01D4                       517      @49:
                           518      ;     */
                           519      ;     call Setled;
01D4 E8C4FE                520              CALL    SETLED
                           521      ;     Output(PIC$MASK) = not (10H);
                           522      ;     /*
                           523      ;       now poll and wait for a CA
                           524      ;     */
01D7 B0EF                  525              MOV     AL,0EFH
01D9 E6F1                  526              OUT     0F1H,AL
                           527      ;     output(7)=1;
```

```
LOC  OBJ                  LINE     SOURCE

01DB E607                  528              OUT     7H,AL           ; DONE WITH RESET, SET DONE FLAG
                           529      ;  L$poll:
01DD                       530      LPOLL:
                           531      ;  Output(PIC$CMD) = POLL$PIC;
01DD B00C                  532              MOV     AL,0CH
01DF E6F0                  533              OUT     0F0H,AL
                           534      ;  if (Input(PIC$CMD) and 80H) = 0 then goto L$poll;
01E1 E4F0                  535              IN      AL,0F0H
01E3 A880                  536              TEST    AL,80H
01E5 74F6                  537              JZ      LPOLL
                           538      ;  /*
                           539      ;    have interrupt (and cmd), get pointer to cmd block
                           540      ;  */
                           541      ;  call Load$cmd$block$ptr;
                           542      ;  /*
                           543      ;    if cmd is dump, execute it now, else start boot task
                           544      ;  */
01E7 E81AFE                545              CALL    LOADCMDBLOCKPTR
                           546      ;  if Cmd = C$local$dump then
01EA 803E000C06      E     547              CMP     CMD,6H
01EF 750E                  548              JNE     @52
                           549      ;  do;
                           550      ;    /*
                           551      ;      do local dump. note that we use the local load
                           552      ;      structure, so terms are reversed
                           553      ;    */
                           554      ;    call Local$move;
01F1 E834FE                555              CALL    LOCALMOVE
                           556      ;    LL$area.response = CMD$OK;
                           557      ;    /*
                           558      ;      do not leave this state, go back to polling
                           559      ;    */
01F4 C41E0000        E     560              LES     BX,DWORD PTR CMDBLOCKPTR
01F8 26C6470101            561              MOV     BYTE PTR ES:[BX+1H],1H
                           562      ;    goto L$poll;
01FD EBDE                  563              JMP     LPOLL
                           564      ;  end;
                           565      ;  /*
                           566      ;    have non-local dump command, set flag that we have
                           567      ;    a command and go start KAOS
01FF                       568      @52:
                           569      ;  */
                           570      ;  First$boot$cmd = TRUE;
01FF 2EC41E0000      R     571              LES     BX,DWORD PTR RANRTPTR
0204 26C64702FF            572              MOV     BYTE PTR ES:[BX+2],0FFH
                           573      ;  Mip$running = FALSE;
                           574      ;  Ran$RT = FALSE;
0209 26C7070000            575              MOV     WORD PTR ES:[BX],0H
                           576      ;  goto Boot$restart;
020E EB1890                577              JMP     BOOTRESTART
                           578      ;  /*
                           579      ;    come here on Dead man timer expiration.
                           580      ;  */
                           581      ;Boot$DMT$entry: First$Boot$cmd = FALSE;
                           582              PUBLIC  BOOTDMTENTRY
```

```
LOC  OBJ                  LINE    SOURCE

0211                       583    BOOTDMTENTRY:
                           584    ;  /*
                           585    ;    come here to restart KAOS
                           586    ;  */
0211 FA                    587           CLI
                           588    ;  call CQmiphalt;
0212 E80000          E     589           CALL CQMIPHALT;
0215 33C9                  590           XOR     CX,CX
0217 2EC41E0C00      R     591    BOOTD1: LES     BX,DWORD PTR RANRTPTR
                           592    ;  First$boot$cmd = FALSE;
                           593    ;  Miprunning = FALSE;
                           594    ;  Ran$RT = TRUE;
021C 26C607FF              595           MOV     BYTE PTR ES:[BX],0FFH   ; RANRT
0220 26C747010000          596           MOV     WORD PTR ES:[BX+1],0    ; FIRSTBOOTCMD AND MIPRUNNING
0226 E0EF                  597           LOOPNZ  BOOTD1                  ; DELAY TO LET OTHER MIPS NOTICE
                           598    ;Boot$restart:
                           599    ;  /*
                           600    ;    now init hardware and start KAOS
                           601    ;  */
0228                       602    BOOTRESTART:
                           603    ;  call CQ$CB$init;
0228 E90000          E     604           JMP     CQCBINIT
                           605    ;end CB$bootstrap;
----                       606    CODE    ENDS
                           607           END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

Start of boot - KAOS state