```
LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE ('KAOS-MIP MIPSEND AND MISC ROUTINES')
                             2           NAME    MIP_SEND_A
                             3 +1  $include(:f1:propa.lit)
                     =1      4     ;
                     =1      5     ;   Intel Corporation Proprietary Information. This listing is
                     =1      6     ;   supplied under the terms of a license agreement with Intel
                     =1      7     ;   Corporation and may not be copied nor disclosed except in
                     =1      8     ;   accordance with the terms of the agreement.
                     =1      9     ;
                            10     ;CMX$Mipsend:
                            11     ;do;
                            12     ;declare Request$queue$descriptor literally
                            13     ;         'RQ$flag byte,
                            14     ;          RQflag2 byte,
                            15     ;          RQsize byte,
                            16     ;          RQE$size byte,
                            17     ;          Give$index byte,
                            18     ;          Give$state byte,
                            19     ;          Take$index byte,
                            20     ;          Take$state byte ';
                            21
                            22     ;declare RQEntry$f1 literally '
                            23     ;        Request$id byte,
                            24     ;        Src$request$id byte,
                            25     ;        Dest$dev$id byte,
                            26     ;        Dest$port$id byte,
                            27     ;        Src$dev$id byte, ' ;
                            28
                            29     ;declare RQEntry$f2 literally '
                            30     ;        Offset word,
                            31     ;        Offset2 word,
                            32     ;        Length word,
                            33     ;        IDS$id byte,
                            34     ;        Owner$dev byte ';
                            35
                            36     ;declare RQEntry$format literally 'RQEntry$f1 RQEntry$f2';
                            37
                            38     ;  declare MIP$msg$format literally
                            39     ;    'Link pointer,
                            40     ;     Offset pointer,
                            41     ;     Length word,
                            42     ;     IDS$id byte,
                            43     ;     Owner$dev byte ' ;
                            44
                            45     ;declare CQ$MIP$ids$bases (Max$no$segments) structure(
                            46     ;        base byte,
                            47     ;        length byte) external;
                            48
                            49     ;declare CQ$mipDevice$info(Max$no$devices) structure (
                            50     ;  devid byte,
```

```
LOC  OBJ                  LINE     SOURCE

                            51     ;  status byte,
                            52     ;  RQDin pointer,
                            53     ;  RQDout pointer,
                            54     ;  Int$type byte,
                            55     ;  Time$to$wait byte,
                            56     ;  Int$adr word ) external;
                            57
                            58     ;declare DI literally 'CQ$MIP$device$info(Device)';
                            59
                            60     ;declare CQ$MIP$use$permit (8) byte external,
                            61     ;        CQ$MIP$send$wtmbx (16) byte external,
                            62     ;        CQ$MIP$remote$mbx (16) byte external,
                            63     ;        CQ$MIP$sendacb (16) byte external,
                            64     ;        CQ$Thisdevice byte external,
                            65
                            66     ;        Port$to$mailbox (Max$no$ports) word external,
                            67     ;        Send$Msg (2) word external,
                            68     ;        Send$result byte external,
                            69     ;        Send$state byte external,
                            70     ;        Send$device byte external,
                            71     ;        Reply$waiting literally '2',
                            72     ;        Non$full$waiting literally '1';
                            73
                            74     ;declare No$subsystems byte external,
                            75     ;        Subsystemlist (5) word external;
                            76
                            77 +1  $INCLUDE(:F1:MIP.EQU)
                 =1         78     ;
                 =1         79     ; DEFINE RQD RESULTS
                 =1         80     ;
0001             =1         81     GERROR   EQU     1H
0004             =1         82     GBUSY    EQU     4H
0008             =1         83     FIRSTG   EQU     8H
0010             =1         84     GDISAB   EQU     10H
0020             =1         85     GFULL    EQU     20H
0040             =1         86     DISABT   EQU     40H
0080             =1         87     FULLF    EQU     80H
                 =1         88
0001             =1         89     TERROR   EQU     1H
0004             =1         90     TBUSY    EQU     4H
0008             =1         91     FIRSTT   EQU     8H
0010             =1         92     TDISAB   EQU     10H
0020             =1         93     TEMPTY   EQU     20H
0040             =1         94     DISABG   EQU     40H
0080             =1         95     EMPTYF   EQU     80H
                 =1         96     ;
                 =1         97     ; DEFINE MIP CMDS AND RESPONSES
                 =1         98     ;
0070             =1         99     CSEND    EQU     70H
0080             =1        100     SENTOK   EQU     8CH
0081             =1        101     UNKNP    EQU     81H
0083             =1        102     ACTIVP   EQU     83H
0085             =1        103     INSUFM   EQU     85H
0087             =1        104     INACTP   EQU     87H
0089             =1        105     DEADP    EQU     89H
```

```
LOC  OBJ                  LINE     SOURCE

                     =1    106     ;
                     =1    107     ; DEFINE MIP-ISIS PARAMETERS
                     =1    108     ;
  0000               =1    109     MYIDS   EQU     0
  0003               =1    110     THIDEV  EQU     3
                           111
                           112     DGROUP  GROUP   DATA
----                       113     DATA    SEGMENT BYTE PUBLIC 'DATA'
                           114             EXTRN   CQMIPDEVICEINFO:NEAR,CQTHISDEVICE:BYTE,CQMIPSENDACB:NEAR
                           115             EXTRN   CQMIPSENDWTMBX:NEAR,SENDRESULT:BYTE,SENDSTATE:BYTE
                           116             EXTRN   CQMIPUSEPERMIT:NEAR,PORTTOMAILBOX:NEAR,SENDDEVICE:BYTE
                           117             EXTRN   CQMIPDEVCNT:BYTE,CQMIPDEVTOENTRY:NEAR
                           118             PUBLIC  SEQ_NO
0000 00                    119     SEQ_NO  DB      0               ; INIT VALUE DOES NOT MATTER
----                       120     DATA    ENDS
                           121
                           122     CGROUP  GROUP   CODE
----                       123     CODE    SEGMENT BYTE PUBLIC 'CODE'
                           124             ASSUME  CS:CGROUP,DS:DGROUP
                           125
                           126             PUBLIC  MIPSEND,CALCDEVPTR,CALCDEVPTR2
                           127             EXTRN   REQUESTGIVEPTR:NEAR,RELEASEGIVEPTR:NEAR,CQSEND:NEAR
                           128             EXTRN   CQRECEIVE:NEAR,CQSETALARM:NEAR,CQCLEARALARM:NEAR
                           129             EXTRN   CQWAITSEM:NEAR,CQSIGNAL:NEAR
                           130
                           131 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           132
                           133     ;declare   RQDptr pointer,
                           134     ;          Rqd based RQDptr structure(Requestqueuedescriptor),
                           135     ;          RQEptr pointer,
                           136     ;          RQEntry based RQEptr structure(Rqentry$format),
                           137     ;          Give$state byte,
                           138     ;          K pointer;
                           139
                           140     ;MIP$send:
0000                       141     MIPSEND      PROC NEAR
0000 5F                    142              POP     DI      ; RETURN ADR
0001 5E                    143              POP     SI
0002 07                    144              POP     ES      ; MSGPTR
0003 58                    145              POP     AX      ; SOCKET
0004 57                    146              PUSH    DI
                           147     ;  procedure(Dsocket,Msgptr) byte public  reentrant ;
                           148     ;  declare Dsocket word,
                           149     ;          Msgptr pointer,
                           150     ;          Msg based Msgptr structure(Mip$msg$format),
                           151     ;          Device byte,
                           152     ;          Mailboxptr word,
                           153     ;          Result byte;
                           154     ;  /*
                           155     ;    if destination is for a local port, then call the os send routine.
                           156     ;  */
                           157     ;  Device = High(Dsocket);
                           158     ;  if Device = CQ$This$Device then
0005 3A26000C        E     159              CMP     AH,CQTHISDEVICE
0009 751A                  160              JNE     @7
                           161     ;  do;
                           162     ;    if (Mailbox$ptr:=Port$to$mailbox(Low(Dsocket))) <> 0 then
000B B400                  163              MOV     AH,0    ; FORM INDEX INTO PTOMB
000D D3C0                  164              ADD     AX,AX
000F 8BD8                  165              MOV     BX,AX
0011 8B87000C        E     166              MOV     AX,WORD PTR PORTTOMAILBOX[BX]
0015 0BC0                  167              OR      AX,AX
0017 7409                  168              JZ      @8
                           169     ;    do;
                           170     ;      call CQ$Send(Mailboxptr,Msgptr);
0019 50                    171              PUSH    AX
001A 06                    172              PUSH    ES
001B 56                    173              PUSH    SI
001C E80000          E     174              CALL    CQSEND
                           175     ;      return SENTOK;
001F B000                  176              MOV     AL,0H
0021 C3                    177              RET
                           178     ;    end;
                           179     ;    else return INACTP;
0022 B007                  180     @8:      MOV     AL,7H
0024 C3                    181              RET
                           182     ;  end;
                           183     ;  /*
                           184     ;    proceed if not busy, else block til not
0025 06                    185     @7:      PUSH    ES
0026 56                    186              PUSH    SI              ; MSGPTR
```

```
LOC  OBJ                  LINE     SOURCE

0027 86C4                  187              XCHG    AL,AH
0029 50                    188              PUSH    AX                  ; SOCKET
                           189      ;  */
                           190      ;  call CQ$Wait$sem(.CQ$MIP$use$permit);
002A B80000          E     191              MOV     AX,OFFSET DGROUP:CQMIPUSEPERMIT
002D 50                    192              PUSH    AX      ; 1
002E E80000          E     193              CALL    CQWAITSEM
                           194      ;  /*
                           195      ;    now get entry in the Request queue to the right device
                           196      ;    first get the RQD for the Request queue.
                           197      ;  */
                           198      ;  RQDptr = Di.RQDout;
                           199      ;  /*
                           200      ;    now loop until we have put the item into the RQ or an error
                           201      ;    (fatal type) occcurs.
                           202      ;  */
0031 5B                    203              POP     BX
0032 53                    204              PUSH    BX                  ; GET SOCKET
                           205      ;  Result = 0FFH;
                           206      ;  do while Result = 0FFH;
                           207      ;    /*
                           208      ;      check device status
                           209      ;    */
                           210      ;    if not Di.status then Result = DEADP;
                           211
0033 E88D00                212              CALL    CALCDEVPTR2     ; THIS ROUTINE CALCULATES THE DEV PTR
0036 7403                  213      @20:    JE      @21A
0038 E98D00                214              JMP     @2              ; IF NOT EQUAL, THEN STATUS IS FALSE
                           215      ;    else
                           216      ;    do;
                           217      ;      /*
                           218      ;        port is active, so try to get a RQE
                           219      ;      */
                           220      ;      disable;
                           221      ;                    TOS => SOCKET,MSGPTR (2)
003B 43                    222      @21A:   INC     BX      ; INCR DEV PTR TO POINT TO STATUS BYTE
003C 53                    223      @20A:   PUSH    BX      ; SAVE DEVPTR
003D FA                    224              CLI
                           225      ;      Give$state = Request$give$ptr(.RQEptr,DEVPTR);
003E 8BFB                  226              MOV     DI,BX
0040 E80000          E     227              CALL    REQUESTGIVEPTR
                           228      ;      if (Give$state and GERROR) = 0 then
0043 753F                  229              JNZ     @12
                           230      ;      do;
                           231      ;        /*
                           232      ;          there is an free RQE and we have. now fill it in
                           233      ;        */
                           234      ;        RQEntry.Request$id = CSEND;
0045 26C60770              235              MOV     BYTE PTR ES:[BX],70H
0049 A00000          R     236              MOV     AL,SEQ_NO               ; INTASK WILL BUMP IT
004C 26884701              237              MOV     ES:[BX+1],AL
                           238      ;        RQEntry.Dest$dev$id = Device;
0050 59                    239              POP     CX                      ; DEVINFO
0051 58                    240              POP     AX                      ; SOCKET
0052 5E                    241              POP     SI                      ; MSGPTR -OFFSET
```

```
LOC  OBJ                  LINE     SOURCE

0053 5F                    242             POP     DI                      ; MSGPTR -BASE
0054 57                    243             PUSH    DI
0055 56                    244             PUSH    SI
0056 50                    245             PUSH    AX
0057 51                    246             PUSH    CX
                           247     ;        RQEntry.Dest$port$id = Low(Dsocket);
0058 26894702              248             MOV     WORD PTR ES:[BX+2H],AX          ; DO BOTH
                           249     ;        RQEntry.Src$dev$id = CQ$This$device;
005C 8A0E0000        E     250             MOV     CL,CQTHISDEVICE
0060 26884F04              251             MOV     BYTE PTR ES:[BX+4H],CL
                           252     ;        Senddevice = Device;
                           253     ;        Send$result = 0;
                           254     ;        Send$state = Reply$waiting;
0064 A20000           E    255             MOV     SENDDEVICE,AL
0067 C70600000200     E    256             MOV     WORD PTR SENDSTATE,0002H        ; STORE RESULT AND STATE
                           257     ;        call Movb(@Msg.Offset,@RQEntry.Offset,8);
006D 1E                    258             PUSH    DS
006E 8EDF                  259             MOV     DS,DI
0070 8D7F05                260             LEA     DI,WORD PTR [BX+5]
0073 83C604                261             ADD     SI,4            ; GET MSG OFFSET RIGHT
0076 B90400                262             MOV     CX,4
0079 FC                    263             CLD
007A F3                    264             REP MOVSW
007B A5
007C 1F                    265             POP     DS      ; 1
                           266     ;        /*
                           267     ;          now release the request queue
                           268     ;        */
                           269     ;        Give$state = Release$give$ptr(DEVPTR);
                           270     ;        /*
                           271     ;          signal the destination device all the time
                           272     ;        */
007D 5F                    273              POP     DI
007E 57                    274              PUSH    DI
007F E80000           E    275              CALL    RELEASEGIVEPTR
                           276     ;       end;
0082 EB0B                  277              JMP     SHORT @13
                           278     ;       else
                           279     ;       if (Give$state and GFULL) <> 0 then Send$state = Nonfullwaiting;
0084 A820                  280     @12:     TEST    AL,20H
0086 FB                    281              STI
0087 743F                  282              JZ      @2
0089 C7060000C100     E    283              MOV     WORD PTR SENDSTATE,0001H
                           284     ;       else
                           285     ;       do;
                           286     ;         /*
                           287     ;           device must be dead, so leave
                           288     ;         */
                           289     ;         Di.status = FALSE;
                           290     ;         Result = DEADP;
                           291     ;         goto Exit;
                           292     ;       end;
                           293     ;       /*
                           294     ;         wait for either reply or a full to not full transition
                           295     ;       */
```

```
LOC  OBJ                  LINE     SOURCE

                           296     ;       enable
C08F FB                    297     @13:     STI
                           298     ;       if Di.Timetowait <> OFFH then call CQ$Set$alarm(@CQMIPsendacb,
0090 58                    299              POP     BX
0091 53                    300              PUSH    BX
0092 8A570A                301              MOV     DL,[BX+0AH]
0095 80FAFF                302              CMP     DL,0FFH
0098 7413                  303              JE      NOLIMIT
009A B600                  304              MOV     DH,0
009C 8D0E000C        E     305              LEA     CX,WORD PTR CQMIPSENDACB
00A0 1E                    306              PUSH    DS      ; 1
00A1 51                    307              PUSH    CX      ; 2
00A2 B90000          E     308              MOV     CX,OFFSET DGROUP:CQMIPSENDWTMBX
00A5 51                    309              PUSH    CX      ; 3
00A6 52                    310              PUSH    DX              ; LENGTH
00A7 B200                  311              MOV     DL,0H
00A9 52                    312              PUSH    DX
00AA E80000          E     313              CALL    CQSETALARM
                           314     ;                               .CQ$MIP$send$wtmbx,double(Di.Time$to$wait),0);
                           315     ;       /*
                           316     ;         wait for result
                           317     ;       */
                           318     ;       K = CQ$Receive(.CQ$MIP$send$wtmbx);
00AD                       319     NOLIMIT:
00AD B80000          E     320              MOV     AX,OFFSET DGROUP:CQMIPSENDWTMBX
00B0 50                    321              PUSH    AX      ; 1
00B1 E80000          E     322              CALL    CQRECEIVE
                           323     ;       Call CQ$Clear$alarm(@CQMIPsendacb);
00B4 8D060000C       E     324              LEA     AX,WORD PTR CQMIPSENDACB
00B8 1E                    325              PUSH    DS      ; 1
00B9 50                    326              PUSH    AX      ; 2
00BA E80000          E     327              CALL    CQCLEARALARM
                           328     ;       if Send$Result = 0 then
00BD 8D3E0000C       E     329              LEA     DI,DGROUP:SENDDEVICE
00C1 807D020C              330              CMP     BYTE PTR [DI+2],0       ; CHECK IF RESULT = 00
00C5 5B                    331              POP     BX              ; DEV PTR
00C6 750B                  332              JNZ     @16
                           333     ;       do;  /* timeout */
                           334     ;         Di.status =  FALSE;
00C8 83C406                335     @2:      ADD     SP,6            ; EQUAL TO 3 POPS
00CB C60700                336              MOV     BYTE PTR [BX],0H
                           337     ;         Result = 0EADP;
00CE B089                  338     @9:      MOV     AL,89H
                           339     ;       end;
00D0 EB1090                340              JMP     @21
                           341     ;       else
                           342     ;       do;
                           343     ;         /*
                           344     ;           something is on the mailbox. If waiting for a reply then
                           345     ;           it is the reply, else it is the full to not full transition
                           346     ;         */
                           347     ;         if Send$state = Reply$waiting then Result = Send$Result;
00D3 807D0102              348     @16:     CMP     BYTE PTR [DI+1],2
00D7 7403                  349              JE      @16A
00D9 E95AFF                350              JMP     @20
```

```
LOC  OBJ                  LINE     SOURCE

00DC 83C406                351     @16A:   ADD     SP,6            ; EQUALS 3 POPS
00DF 8A4502                352             MOV     AL,BYTE PTR [DI+2]  ; SEND RESULT
                           353     ;      end;
                           354     ;    end;
                           355     ;Exit:
                           356     ;  end;
                           357     ;  /*
                           358     ;    now return permit to system and return to user
                           359     ;  */
                           360     ;  Send$state = 0;
00E2 C606000C00      E     361     @21:    MOV     SENDSTATE,0H
                           362     ;  call CQ$Signal(.CQ$MIP$use$permit);
00E7 50                    363             PUSH    AX              ; SAVE STATUS
00E8 B80000          E     364             MOV     AX,OFFSET DGROUP:CQMIPUSEPERMIT
00EB 50                    365             PUSH    AX      ; 1
00EC E80000          E     366             CALL    CQSIGNAL
                           367     ;  return Result and 07FH;
                           368
00EF 58                    369             POP     AX
00F0 247F                  370             AND     AL,7FH
00F2 C3                    371             RET
                           372     ;end MIP$send;
                           373     MIPSEND      ENDP
                           374
                           375     ;
                           376     ; THESE ROUTINES CALCULATE THE DEVICE PTR. CALCDEVPTR2 DOES IT BASED ON
                           377     ; AN INDEX INTO CQMIPDEVTOENTRY IN BL. CALCDEVPTR DOES IT BASED ON
                           378     ; THE INDEX INTO CQMIPDEVINFO  IN AL. BOTH  USE AX,CX, BX, AND SI
                           379     ;
00F3                       380     CALCDEVPTR2:
00F3 B700                  381             MOV     BH,0
00F5 8A87000C        E     382             MOV     AL,BYTE PTR CQMIPDEVTOENTRY[BX]
00F9                       383     CALCDEVPTR:
00F9 B10E                  384             MOV     CL,0EH
00FB F6E1                  385             MUL     CL
00FD 8BD8                  386             MOV     BX,AX
00FF 8D9F000C        E     387             LEA     BX,WORD PTR CQMIPDEVICEINFO[BX]
0103 807F01FF              388             CMP     BYTE PTR [BX+1],0FFH
0107 C3                    389             RET
                           390
----                       391     CODE    ENDS
                           392     ;end CMX$Mipsend;
                           393
                           394
                           395 +1  $TITLE ('KAOS MIP MISC FUNCTIONS')
                           396 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           397
                           398     ;Subsystem$call:
                           399     ;  procedure(Devid, Processid, Procptr) external;
                           400     ;  declare Devid byte,
                           401     ;           Processid word,
                           402     ;           Procptr word;
                           403     ;end Subsystem$call;
                           404
                           405
----                       406     DATA    SEGMENT BYTE PUBLIC 'DATA'
                           407
                           408             EXTRN   NOSUBSYSTEMS:NEAR,SUBSYSTEMLIST:NEAR
                           409             EXTRN   CQMIPREMOTEMBX:NEAR
                           410
----                       411     DATA    ENDS
                           412
----                       413     CODE    SEGMENT BYTE PUBLIC     'CODE'
                           414             ASSUME  CS:CGROUP,DS:DGROUP
                           415
                           416             EXTRN   SUBSYSTEMCALL:NEAR
                           417
                           418
                           419     ;Mip$Connect:
                           420             PUBLIC  MIPCONNECT
0108                       421     MIPCONNECT      PROC NEAR
0108 5F                    422             POP     DI      ; RETURN ADDRESS
0109 58                    423             POP     AX      ; MBptr
010A 5B                    424             POP     BX      ; PORTT
010B 57                    425             PUSH    DI
                           426     ;  procedure(Port,Mbptr) byte public;
                           427     ;  declare Port byte, Mbptr word;
                           428
                           429     ;  if Port >= Max$no$ports then return 1;
010C 80FB18                430             CMP     BL,24
010F 7203                  431             JB      @X1
0111 B001                  432             MOV     AL,1H
0113 C3                    433             RET
0114                       434     @X1:
                           435     ;  Port$to$mailbox(Port) = Mbptr;
0114 B700                  436             MOV     BH,0
0116 03DB                  437             ADD     BX,BX
0118 89870000      E       438             MOV     WORD PTR PORTTOMAILBOX[BX],AX
                           439     ;  return 0;
011C 33C0                  440             XOR     AX,AX
011E C3                    441             RET
                           442     ;end Mip$Connect;
                           443     MIPCONNECT      ENDP
                           444
                           445
                           446     ;Mip$register:
                           447             PUBLIC  MIPREGISTER
011F                       448     MIPREGISTER      PROC NEAR
011F 5F                    449             POP     DI
0120 58                    450             POP     AX
0121 57                    451             PUSH    DI
```

```
LOC  OBJ                  LINE     SOURCE

                           452     ;  procedure (Proc$ptr) byte public;
                           453     ;  declare Proc$ptr word;
                           454
                           455     ;  if No$subsystems = 5 then return 1;
0122 BE0000          E     456             MOV     SI,OFFSET DGROUP:NOSUBSYSTEMS
0125 803C05                457             CMP     BYTE PTR [SI],5
0128 7503                  458             JNE     @X2
012A B001                  459             MOV     AL,1
012C C3                    460             RET
012D                       461     @X2:
                           462     ;  Subsystem$list(No$subsystems) = Proc$ptr;
012D 8A1C                  463             MOV     BL,BYTE PTR [SI]
012F B700                  464             MOV     BH,0H
0131 03DB                  465             ADD     BX,BX
0133 89870000        E     466             MOV     WORD PTR SUBSYSTEMLIST[BX],AX
                           467     ;  No$subsystems = No$subsystems + 1;
0137 FE04                  468             INC     BYTE PTR [SI]
                           469     ;  return 0;
0139 33C0                  470             XOR     AX,AX
013B C3                    471             RET
                           472     ;end Mip$register;
                           473     MIPREGISTER      ENDP
                           474
                           475 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           476     ;MIP$set$device$status:
                           477             PUBLIC  MIPSETDEVICESTATUS
C13C                       478     MIPSETDEVICESTATUS      PROC NEAR
013C 5F                    479             POP     DI
013D 5A                    480             POP     DX      ; STATE
013E 5B                    481             POP     BX      ; PORT
013F 57                    482             PUSH    DI
                           483     ;  procedure (Deviceid, State) byte public;
                           484     ;  declare Deviceid byte,
                           485     ;          State byte;
                           486
                           487     ;  if Deviceid >= Max$no$devices then return 1;
0140 80FB0A                488             CMP     BL,10
0143 7203                  489             JB      @X3
0145 B001                  490             MOV     AL,1H
0147 C3                    491             RET
0148                       492     @X3:
                           493     ;  CQ$MIP$device$info(CQ$mipdev$to$entry(Deviceid)).status = State;
0148 E8A8FF                494             CALL    CALCDEVPTR2
014B 885701                495             MOV     BYTE PTR [BX+1],DL
                           496     ;  return 0;
014E B000                  497             MOV     AL,0H
0150 C3                    498             RET
                           499     ;end Mip$set$device$status;
                           500     MIPSETDEVICESTATUS      ENDP
                           501
                           502
                           503     ;MIP$halt:
                           504             PUBLIC  MIPHALT
0151                       505     MIPHALT         PROC NEAR
                           506     ;  procedure public;
                           507     ;  declare I byte,
                           508     ;          RQDptr pointer,
                           509     ;          Rqd based RQDptr structure(Request$queue$descriptor);
                           510     ;  do I = 0 to CQmipdevcnt-1;
0151 33D2                  511             XOR     DX,DX
0153                       512     @X9:
0153 3A160000      E       513             CMP     DL,CQmipdevcnt
0157 7422                  514             JE      @X10
                           515     ;    if CQ$MIP$device$info(I).status then
0159 8AC2                  516             MOV     AL,DL
015B E893FF                517             CALL    CALCDEVPTR
015E 43                    518             INC     BX              ; POINT TO STATUS
015F B500                  519             MOV     CH,0
0161 862F                  520             XCHG    CH,BYTE PTR [BX]
0163 0AED                  521             OR      CH,CH
0165 7410                  522             JZ      @X11
                           523     ;    do;
                           524     ;      RQDptr = CQ$MIP$Device$info(I).RQDout;
0167 C47705                525              LES     SI,DWORD PTR [BX+5H]
                           526     ;      Rqd.Give$state = DISABT;
016A 26C6440540            527              MOV     BYTE PTR ES:[SI+5H],40H
                           528     ;      RQDptr = CQ$MIP$device$info(I).RQDin;
016F C47701                529              LES     SI,DWORD PTR [BX+1H]
                           530     ;      Rqd.Take$state = DISABG;
```

```
LOC  OBJ                  LINE      SOURCE

C172 26C6440740            531               MOV     BYTE PTR ES:[SI+7H],40H
                           532       ;     CQ$MIP$device$info(I).status = FALSE;
                           533       ;   end;
                           534       ; end;
C177                       535       @X11:
C177 FEC2                  536               INC     DL
C179 EBD8                  537               JMP     @X9
                           538       ;end Mip$halt;
C17B C3                    539       @X10:   RET
                           540       MIPHALT       ENDP
                           541
                           542
                           543
                           544 +1    $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           545
                           546     ;CQ$Mip$task:
                           547             PUBLIC CQMIPTASK
017C                       548     CQMIPTASK       PROC NEAR
                           549     ;  procedure public;
                           550     ;  declare Reqptr pointer,
                           551     ;          I byte,
                           552
                           553     ;          Req based Reqptr structure(
                           554     ;            Mip$msg$format,
                           555     ;            Processid word,
                           556     ;            Cmd byte,
                           557     ;            Response byte,
                           558     ;            Response$socket word,
                           559     ;            Deleted$process$dev byte,
                           560     ;            Deleted$process$id word ),
                           561
                           562     ;          Add$device based Reqptr structure(
                           563     ;            Mip$msg$format,
                           564     ;            Processid word,
                           565     ;            Cmd byte,
                           566     ;            Response byte,
                           567     ;            Response$socket word,
                           568     ;            Devid byte,
                           569     ;            Status byte,
                           570     ;            info (12) byte );
                           571
                           572
                           573     ;  do forever;
017C                       574     @X12:
                           575     ;     Reqptr = CQ$receive(.CQ$MIP$remote$mbx);
017C B80000          E     576             MOV     AX,OFFSET DGROUP:CQMIPREMOTEMBX
017F 50                    577             PUSH    AX      ; 1
0180 E80000          E     578             CALL    CQRECEIVE
0183 26FF7710              579             PUSH    WORD PTR ES:[BX+10H]
0187 06                    580             PUSH    ES      ; SAVE REQPTR
0188 53                    581             PUSH    BX
                           582     ;     Req.Response = 0;
0189 26C6470F00            583             MOV     BYTE PTR ES:[BX+0FH],0H
                           584     ;     if Req.Cmd = 1 then
018E 26807F0E01            585             CMP     BYTE PTR ES:[BX+0EH],1H
0193 752B                  586             JNE     @X5
                           587     ;     do; /* Add device */
                           588     ;       if (I:=Add$device.devid) >= Highest$device then Add$device.response = 1;
0195 268A4712              589             MOV     AL,ES:[BX+12H]
0199 3C0A                  590             CMP     AL,10
019B 7208                  591             JB      @X6
019D 26C6470F01            592             MOV     BYTE PTR ES:[BX+0FH],1H
01A2 EB4590                593             JMP     @X8
01A5                       594     @X6:
                           595     ;       else
                           596     ;       do;
                           597     ;         I = CQMipdevtoentry(I);
                           598     ;         call Movb(@Add$device.devid ,@CQ$MIP$device$info(I),14);
01A5 53                    599             PUSH    BX              ; SAVE PTR TO DEVID
```

await msg

```
LOC  OBJ                  LINE     SOURCE

01A6 8AD8                  600              MOV     BL,AL
01A8 E848FF                601              CALL    CALCDEVPTR2
01AB 8BFB                  602              MOV     DI,BX           ; DEST PTR INTO DEVINFO TABLE
01AD B90E00                603              MOV     CX,14           ; BYTES TO MOVE
01B0 5E                    604              POP     SI
01B1 83C612                605              ADD     SI,12H
01B4 1E                    606              PUSH    DS
01B5 1E                    607              PUSH    DS
01B6 06                    608              PUSH    ES
01B7 1F                    609              POP     DS
01B8 07                    610              POP     ES
01B9 FC                    611              CLD
01BA F3                    612              REP     MOVSB
01BB A4
01BC 1F                    613              POP     DS
                           614      ;     end;
                           615      ;   end;
01BD EB2A90                616              JMP     @X8
01C0                       617      @X5:
                           618      ;    else
                           619      ;    do; /* delete process */
                           620      ;      do I = 0 to No$subsystems-1;
01C0 B100                  621              MOV     CL,0
01C2 8A2E0000      E       622              MOV     CH,BYTE PTR NOSUBSYSTEMS
01C6                       623      @X14:
01C6 5B                    624              POP     BX              ; GET BACK REQPTR
01C7 07                    625              POP     ES
01C8 06                    626              PUSH    ES
01C9 53                    627              PUSH    BX
01CA 3AE9                  628              CMP     CH,CL
01CC 741B                  629              JE      @X8
                           630      ;         call Subsystem$call(Req.Deleted$process$dev,Req.Deleted$process$id,
01CE 51                    631              PUSH    CX
01CF 26FF7712              632              PUSH    WORD PTR ES:[BX+12H]
01D3 26FF7713              633              PUSH    WORD PTR ES:[BX+13H]
01D7 8AD9                  634              MOV     BL,CL
01D9 D1E3                  635              SHL     BX,1
01DB B700                  636              MOV     BH,0
01DD FFB70000      E       637              PUSH    WORD PTR SUBSYSTEMLIST[BX]; 3
01E1 E80000        E       638              CALL    SUBSYSTEMCALL
01E4 59                    639              POP     CX
                           640      ;              Subsystem$list(I));
                           641      ;       end;
01E5 FEC1                  642              INC     CL
01E7 EBDD                  643              JMP     @X14
                           644      ;    end;
01E9                       645      @X8:
                           646
                           647      ;    I = CQ$mip$send(Req.Response$socket,Reqptr);
01E9 E814FE                648              CALL    MIPSEND
                           649      ;  end;
01EC EB8E                  650              JMP     @X12
                           651
                           652      ;  end CQ$MIP$task;
                           653      CQMIPTASK       ENDP
```

```
LOC  OBJ                  LINE      SOURCE

                           654      ;end Mip$misc;
----                       655      CODE    ENDS
                           656
                           657
                           658
                           659              END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```