```
LOC  OBJ                  LINE     SOURCE

                             1 +1  $TITLE('KAOS-MIP INPUT ROUTINE ')
                             2           NAME    KAOS_INTASK_A
                             3 +1  $include(:f1:propa.lit)
                       =1    4     ;
                       =1    5     ;   Intel Corporation Proprietary Information. This listing is
                       =1    6     ;   supplied under the terms of a license agreement with Intel
                       =1    7     ;   Corporation and may not be copied nor disclosed except in
                       =1    8     ;   accordance with the terms of the agreement.
                       =1    9     ;
                            10     ;declare Request$cueue$descriptor literally
                            11     ;         'RQ$flag byte,
                            12     ;          RQ$flag2 byte,
                            13     ;          RQsize byte,
                            14     ;          RQE$size byte,
                            15     ;          Give$index byte,
                            16     ;          Give$state byte,
                            17     ;          Take$index byte,
                            18     ;          Take$state byte ';
                            19
                            20     ;declare RQEntry$f1 literally '
                            21     ;        Request$id byte,
                            22     ;        Src$request$id byte,
                            23     ;        Dest$dev$id byte,
                            24     ;        Dest$port$id byte,
                            25     ;        Src$dev$id byte,
                            26     ;        Offset word,
                            27     ;        Offset2 word,
                            28     ;        Length word,
                            29     ;        IDS$id byte,
                            30     ;        Owner$dev byte ';
                            31
                            32
                            33     ;declare CQ$MIP$ids$bases (Max$no$segments) structure(
                            34     ;        base byte,
                            35     ;        length byte) external;
                            36
                            37     ;declare CQ$mipDevice$info(Max$no$devices) structure (
                            38     ;  status byte,
                            39     ;  RQDin pointer,
                            40     ;  RQDout pointer,
                            41     ;  Int$type byte,
                            42     ;  Time$to$wait byte,
                            43     ;  Int$adr word ) external;
                            44
                            45     ;declare CQ$MIP$use$permit (8) byte external,
                            46     ;        CQ$MIP$send$wtmbx (16) byte external,
                            47     ;        CQ$MIP$remote$mbx (16) byte external,
                            48     ;        CQ$MIP$sendacb (16) byte external,
                            49     ;        CQ$Thisdevice byte external,
                            50
```

```
LOC  OBJ                  LINE     SOURCE

                            51     ;         Port$to$mailbox (Max$no$ports) word external,
                            52     ;         Send$Msg (2) word external,
                            53     ;         Send$result byte external,
                            54     ;         Send$state byte external,
                            55     ;         Send$device byte external,
                            56     ;         Reply$waiting literally '2',
                            57     ;         Non$full$waiting literally '1';
                            58
                            59     ;declare No$subsystems byte external,
                            60     ;         Subsystemlist (5) word external;
                            61
                            62     ;  declare MIP$msg$format literally
                            63     ;    'Link pointer,
                            64     ;     Offset pointer,
                            65     ;     Length word,
                            66     ;     IDS$id byte,
                            67     ;     Owner$dev byte ' ;
                            68
                            69     DGROUP  GROUP   DATA
----                        70     DATA    SEGMENT BYTE PUBLIC 'DATA'
                            71
                            72             EXTRN   CQMIPDEVICEINFO:NEAR,SENDSTATE:BYTE,SENDDEVICE:BYTE
                            73             EXTRN   CQMIPSENDWTMBX:NEAR,SENDMSG:NEAR,SENDRESULT:BYTE
                            74             EXTRN   CQTHISDEVICE:NEAR
                            75             EXTRN   PORTTOMAILBOX:NEAR,CQMIPDEVCNT:BYTE,SEQ_NO:BYTE
0000 00                     76     TRESULT DB      0
0001 00                     77     SREQID  DB      0               ; MUST FOLLOW TRESULT
----                        78     DATA    ENDS
                            79
                            80     CGROUP  GROUP   CODE
----                        81     CODE    SEGMENT BYTE PUBLIC 'CODE'
                            82             ASSUME  CS:CGROUP,DS:DGROUP
                            83
                            84             EXTRN   REQUESTGIVEPTR:NEAR,REQUESTTAKEPTR:NEAR
                            85             EXTRN   RELEASEGIVEPTR:NEAR,RELEASETAKEPTR:NEAR
                            86             EXTRN   CALCDEVPTR:NEAR,CQISEND:NEAR
                            87
                            88             PUBLIC  CQMIPINTASK
                            89 +1  $eject
```

```
LOC  OBJ                  LINE     SOURCE

                           90      ;
                           91      ; THIS ROUTINE IS CALLED WHEN A INTERRUPT IS RECEIVED. LOOK IN
                           92      ; EACH ACTIVE RQD TO SEE IF ANYTHING IS THERE.
                           93      ;
0000                       94      CQMIPINTASK     PROC NEAR
                           95
                           96      ;  do Device = 0 to Max$no$devices-1;
0000 B2FF                  97              MOV     DL,0FFH
0002 52                    98              PUSH    DX
0003                       99      @20:
0003 5A                   100              POP     DX
0004 FEC2                 101              INC     DL
0006 3A160000      E      102              CMP     DL,CQMIPDEVCNT
000A 7501                 103              JNE     @2CA
000C C3                   104              RET                             ; ALL RQDS LOOKED IN
                          105      ;
                          106      ;       look at RQs for each device. first at the one from the device.
                          107      ;
000D 52                   108      @20A:   PUSH    DX
000E 8AC2                 109              MOV     AL,DL
0010 E80000        E      110              CALL    CALCDEVPTR              ; CALC DEV POINTER
0013 75EE                 111              JNE     @20
0015 43                   112              INC     BX                      ; INDEX PAST DEVID
0016 C47701               113              LES     SI,DWORD PTR [BX+1H]
0019 268B04               114              MOV     AX,ES:[SI]      ; GET RQFLAG AND RQFLAG2
001C 0BC0                 115              OR      AX,AX           ; SET FLAGS
001E 74E3                 116              JZ      @20             ; JMP IF BOTH FLAGS ZERO
0020 53                   117              PUSH    BX              ; SAVE DEVPTR
0021 792C                 118              JNS     @19A            ; IF HIGH BIT SET THEN ITS FLAG2
0023 50                   119              PUSH    AX              ; SAVE FLAG
                          120      ;
                          121      ;     We have a full to not full transition. see if anyone was waiting
                          122      ;
0024 26C6440100           123              MOV     BYTE PTR ES:[SI+1],0
0029 8D1E000C      E      124              LEA     BX,DGROUP:SENDDEVICE
002D B601                 125              MOV     DH,1
002F 3B17                 126              CMP     DX,[BX]         ; COMPARES SENDSTATE AND SENDDEVICE
0031 7514                 127              JNZ     @19
0033 C6470201             128              MOV     BYTE PTR [BX+2],1  ; SET SENDRESULT TO 1
                          129      ;
                          130      ; SOMEONE WAS WAITING,SO ACTIVATE THEM
                          131      ;
0037 06                   132              PUSH    ES
0038 56                   133              PUSH    SI
0039 B80000        E      134              MOV     AX,OFFSET DGROUP:CQMIPSENDWTMBX
003C 50                   135              PUSH    AX      ; 1
003D 83C303               136              ADD     BX,3    ; BX IS NOW ADDRESS OF SENDMSG
0040 1E                   137              PUSH    DS      ; 2
0041 53                   138              PUSH    BX      ; 3
0042 E80000        E      139              CALL    CQISEND
0045 5E                   140              POP     SI              ; GET BACK ES,SI
0046 07                   141              POP     ES
                          142      ;
                          143      ; now look for an empty to not-empty transition
                          144      ;
```

```
LOC  OBJ                  LINE     SOURCE

0047 58                    145     @19:    POP     AX      ; GET FLAGS BACK
0048 A801                  146             TEST    AL,1
004A 7503                  147             JNZ     @19A    ; JMP IF SET
004C 5B                    148             POP     BX      ; NO SUCH TRANSITION, GO ON TO NEXT DEVICE
004D EBB4                  149             JMP     SHORT @20
                           150     ;
                           151     ;        now take all things from this RQ until an error occurs. The
                           152     ;        most likely error is that it is empty.
                           153     ;
004F 26C60400              154     @19A:   MOV     BYTE PTR ES:[SI],0
0053 5F                    155     @22:    POP     DI
0054 57                    156             PUSH    DI
0055 E80000          E     157             CALL    REQUESTTAKEPTR
0058 740A                  158             JZ      @22A
                           159     ;
                           160     ;    the take req returned with an error. That may mean:
                           161     ;    (1) it was empty, or (2) it was disabled.
                           162     ;
005A A810                  163             TEST    AL,10H
005C 5E                    164             POP     SI              ; DEV PTR
005D 74A4                  165             JZ      @20             ; IF NOT ZERO, THEN WAS DISABLED
                           166                                     ; IT WAS EMPTY, GO CHECK ON FULL TO NOT FULL
005F C60400                167     @2:     MOV     BYTE PTR [SI],0         ; QUEUE WAS DISABLED
0062 EB9F                  168             JMP     SHORT @20               ; GO WORK ON NEXT DEVICE
                           169     ;
                           170     ;    there is something in the queue, take it
                           171     ;
0064 268B07                172     @22A:   MOV     AX,WORD PTR ES:[BX]     ; SEE IF IT A CMD OR A RESPONSE
0067 88260100        R     173             MOV     SREQID,AH
006B 3C70                  174             CMP     AL,70H
006D 754F                  175             JNE     @3                      ; JMP IF RESPONSE
                           176     ;
                           177     ;   See if socket is open
                           178     ;
006F 268A4703              179             MOV     AL,ES:[BX+3H]
0073 3C17                  180             CMP     AL,23           ; CHECK FOR PORT AREA
0075 7740                  181             JA      @4
0077 B400                  182             MOV     AH,0H           ; CALC OFFSET WITHIN PORT TO MAILBOX TABLE
0079 D1E0                  183             SHL     AX,1
007B 8BF8                  184             MOV     DI,AX           ; IF PORT HAS MAILBOX SEND MSG TO IT
007D 8B850000        E     185             MOV     AX,WORD PTR PORTTOMAILBOX [DI]
0081 0BC0                  186             OR      AX,AX
0083 7432                  187             JZ      @4              ; JMP IF PORT CLOSED
0085 50                    188             PUSH    AX              ; PUT MB ON STACK FOR LATER USE
0086 06                    189             PUSH    ES              ; SAVE TREQPTR
0087 53                    190             PUSH    BX
                           191     ;
                           192     ;   the socket is open
                           193     ;
0088 C606000C80      R     194             MOV     TRESULT,80H
008D 268B4F05              195             MOV     CX,ES:[BX+5]            ; CONVERT MIP ADDRESS TO REAL ADDRESS
0091 268B5707              196             MOV     DX,ES:[BX+7H]           ; LOAD MIP ADR AND IDS ID
0095 268A770B              197             MOV     DH,ES:[BX+0BH]
0099 52                    198             PUSH    DX
009A 51                    199             PUSH    CX
```

```
LOC  OBJ                  LINE     SOURCE

009B E87000                200              CALL    MIPGETADDRESS           ; GET REAL ADDRESS
                           201      ;
                           202      ;       NOW MAKE COPY OF RQE ENTRY WITHIN THE MSG ITSELF
                           203      ;
009E 8D7F04                204              LEA     DI,WORD PTR [BX+4]
00A1 5E                    205              POP     SI                      ; TRQEPTR
00A2 8D7405                206              LEA     SI,[SI+5H]
00A5 8CDA                  207              MOV     DX,DS
00A7 1F                    208              POP     DS
00A8 52                    209              PUSH    DX
00A9 B90400                210              MOV     CX,4H
00AC FC                    211              CLD
00AD F3                    212              REP MOVSW
00AE A5
00AF 1F                    213              POP     DS      ; 1
                           214      ;
                           215      ;     send it off to the user
                           216      ;
00B0 06                    217              PUSH    ES      ; 2
00B1 53                    218              PUSH    BX      ; 3
00B2 E80000          E     219              CALL    CQISEND
00B5 EB26                  220              JMP     SHORT @8
                           221      ;
                           222      ;       the port is closed, return that info
                           223      ;
00B7 C606000C87      R     224      @4:     MOV     TRESULT,87H
                           225      ;
                           226      ;     we have processed the command and have a response. now
                           227      ;     send it back to the source device.
                           228      ;
00BC EB1F                  229      @5:       JMP     SHORT @8
                           230      ;
                           231      ;     the received item is a response, so do what needs to
                           232      ;     be done.
                           233      ;
                           234      ;     give waiting task the status and then restart it
                           235      ;
00BE 8D1E000C        E     236      @3:       LEA     BX,DGROUP:SENDDEVICE
00C2 884702                237                MOV     [BX+2],AL               ; STORE RESULT
                           238      ;
                           239      ;     make sure this is the ack we were waiting for
                           240      ;
                           241      ;           if Send$state = Reply$waiting then
00C5 807F0102              242                CMP     BYTE PTR [BX+1],2H
00C9 7510                  243                JNE     @7
00CB FE060000        E     244                INC     SEQ_NO
00CF B80000          E     245                MOV     AX,OFFSET DGROUP:CQMIPSENDWTMBX
00D2 50                    246                PUSH    AX      ; 1
00D3 83C303                247                ADD     BX,3            ; CALC SENDMSG OFFSET
00D6 1E                    248                PUSH    DS      ; 2
00D7 53                    249                PUSH    BX      ; 3
00D8 E80000          E     250                CALL    CQISEND         ; send msg off to restart mipsend task
00DB EB22                  251      @7:       JMP     SHORT @11
                           252      ;
                           253      ;          the question now is whether we need to generate a reply.
```

```
LOC  OBJ                  LINE     SOURCE

                           254     ;          Replies are necessary only for received commands.
                           255     ;
                           256     ;           if need to generate reply, do it now
                           257     ;          */
00DD 33C9                  258     @8:     XOR     CX,CX
00DF                       259     TRYINGTOGIVE:
                           260     ;          /*
                           261     ;            see if we can put the response into the RQ
                           262     ;          */
00DF 5F                    263             POP     DI              ; GET DEVICE PTR
00E0 57                    264             PUSH    DI
00E1 41                    265             INC     CX
00E2 7418                  266             JZ      @24             ; IF ZERO THEN TIMEOUT
00E4 51                    267             PUSH    CX
00E5 E80C00          E     268             CALL    REQUESTGIVEPTR
00E8 59                    269             POP     CX
00E9 750D                  270             JNZ     @12
                           271     ;            /* there is space in the RQ */
                           272     ;            G$RQEntry.Request$id = T$result;
00EB A10000          R     273             MOV     AX,WORD PTR TRESULT
                           274     ;            G$RQEntry.Src$request$id = T$RQEntry.Src$Request$id;
                           275     ;            /*
                           276     ;              check for empty to not empty transition
                           277     ;            */
00EE 268907                278             MOV     ES:[BX],AX
00F1 5F                    279             POP     DI
00F2 57                    280             PUSH    DI
00F3 E80000          E     281             CALL    RELEASEGIVEPTR
00F6 EB07                  282             JMP     SHORT @11
                           283     ;
                           284     ; come here on full queue
                           285     ;
00F8 A810                  286     @12:    TEST    AL,10H
00FA 74E3                  287             JZ      TRYINGTOGIVE
00FC E960FF                288     @24:    JMP     @2                      ; DEVICE DISABLED, FORGET IT
                           289     ;        /*
                           290     ;          if we are done processing the taken item, then release
                           291     ;          the RQ and see if there was a full to not-full transition.
                           292     ;          If so, then signal the device. In any case complete loop
                           293     ;          and see if there is anything else in the RQ.
                           294     ;        */
00FF 5F                    295     @11:    POP     DI
0100 57                    296             PUSH    DI
0101 E80000          E     297             CALL    RELEASETAKEPTR
0104 E94CFF                298             JMP     @22
                           299     CQMIPINTASK     ENDP
                           300
----                       301     CODE    ENDS
                           302
                           303
                           304 +1  $TITLE('KAOS-MIP INITIALIZATION ROUTINE')
                           305 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           306
----                       307     DATA    SEGMENT PUBLIC 'DATA'
                           308             EXTRN   CQMIPREMOTEMBX:NEAR
----                       309     DATA    ENDS
                           310
----                       311     CODE    SEGMENT PUBLIC  'CODE'
                           312             ASSUME  CS:CGROUP,DS:DGROUP
                           313             PUBLIC  CQMIPINIT
                           314
0107                       315     CQMIPINIT       PROC NEAR
                           316     ;  procedure public;
                           317     ;  declare I byte;
                           318     ;  /*
                           319     ;    Initialize things for MIP
                           320     ;  */
                           321     ;  I = CQ$MIP$connect(0,.CQ$MIP$remote$mbx);
0107 B80000          E     322             MOV     AX,OFFSET DGROUP:CQMIPREMOTEMBX
010A A30000          E     323             MOV     WORD PTR PORTTOMAILBOX,AX               ; LOAD IT DIRECTLY
                           324     ;end CQ$Mipinit;
010D C3                    325             RET
                           326     CQMIPINIT       ENDP
----                       327     CODE    ENDS
                           328     ;end CMX$Mip$init;
                           329
                           330
                           331
                           332 +1  $TITLE('MIP POINTER ARITHMETIC ROUTINES')
                           333 +1  $EJECT
```

```
LOC  OBJ                  LINE     SOURCE

                           334
----                       335     DATA    SEGMENT BYTE PUBLIC 'DATA'
                           336
                           337             EXTRN   CQMIPIDSBASES:NEAR
                           338
----                       339     DATA    ENDS
                           340
----                       341     CODE    SEGMENT BYTE PUBLIC     'CODE'
                           342             ASSUME  CS:CGROUP,DS:DGROUP
                           343     ;
                           344     ;
                           345     ; THIS ROUTINE CONVERTS AN IDS AND A 24 BIT POINTER INTO
                           346     ; A 8086 LONG POINTER. THE FORMAT OF THE INPUT IS AS FOLLOWS:
                           347     ;
                           348     ;
                           349             PUBLIC  MIPGETADDRESS
C10E                       350     MIPGETADDRESS:
C10E 5F                    351             POP     DI      ; RET ADR
C10F 5B                    352             POP     BX      ; LOW ORDER 16 BITS
0110 59                    353             POP     CX      ; IDS AND HIGH ORDER 8 BITS
0111 57                    354             PUSH    DI      ; RET ADR
0112 8BC1                  355             MOV     AX,CX
                           356     ;
                           357     ; FORM PLM PTR FROM 24 BIT OFFSET
                           358     ;
0114 8BD3                  359             MOV     DX,BX   ; MAKE COPY OF LOWER 16 BITS
0116 81E2F0FF              360             AND     DX,0FFF0H ; GET RID OF LOWER 4 BITS
C11A 81E10F00              361             AND     CX,0FH
011E 0BD1                  362             OR      DX,CX   ; OR IN LOWER 4 BITS OF HIGH ORDER 8 BITS
0120 B104                  363             MOV     CL,4
0122 D3CA                  364             ROR     DX,CL   ; FORM BASE
0124 81E30F0C              365             AND     BX,0FH  ; FORM OFFSET
                           366     ;
                           367     ; NOW GET IDSID BASE
                           368     ;
0128 33C9                  369             XOR     CX,CX
012A 8AC4                  370             MOV     AL,AH
012C 3C07                  371             CMP     AL,7    ; CHECK FOR IDS > 7. (IE AN INVALID VALUE)
012E 773D                  372             JA      S2A     ; RETURN OFFFFH IN ES IF IDSIS > 7 -SORRY FOR LONG JUMP
0130 32E4                  373             XOR     AH,AH   ; MAKE IDS UPPER BYTE ZERO
0132 D1E0                  374             SHL     AX,1    ; MULT BY 2
0134 8BF0                  375             MOV     SI,AX
0136 3AAC0100        E     376             CMP     CH,BYTE PTR DGROUP:CQMIPIDSBASES[SI+1]  ; IF LEN = 0, EXIT
013A 7431                  377             JZ      S2A     ; IF IDS LENGTH = 0, RETURN OFFFFH - SORRY AGAIN !!
013C 8AAC0000        E     378             MOV     CH, BYTE PTR DGROUP:CQMIPIDSBASES[SI]
                           379                             ; HAVE BASE IN AX AS A PARAGRAPH ADDRESS
                           380     ;
                           381     ; NOW ADD IN IDS BASE TO PTR BASE
                           382     ;
0140 03D1                  383     MGA1:   ADD     DX,CX
0142 8EC2                  384             MOV     ES,DX
0144 C3                    385             RET             ; RETURN
                           386
                           387
                           388     ; THIS ROUTINE TAKES A 32 BIT PLM86 POINTER AND RETURNS THE
```

```
LOC  OBJ                  LINE     SOURCE

                           389     ; IDS IT IS IN AND A 24 BIT OFFSET INTO THAT IDS CORRESPONDING
                           390     ; TO THE POINTER VALUE. IF IT IS NOT IN ANY IDS, IT RETURNS
                           391     ; IDSID = OFFH.
                           392     ;
                           393             PUBLIC  MIPGETMIPFORM
0145                       394     MIPGETMIPFORM:
0145 5B                    395             POP     BX      ; GET RETURN ADDRESS
0146 58                    396             POP     AX      ; OFFSET PART OF POINTER
0147 5A                    397             POP     DX      ; BASE PART OF POINTER
0148 53                    398             PUSH    BX      ; RESTORE RETURN ADDRESS
                           399     ;
                           400     ; NOW FORM THE PARAGRAPH NUMBER OF THE POINTER
                           401     ;
0149 8BF8                  402             MOV     DI,AX   ; MAKE COPY OF OFFSET
014B B104                  403             MOV     CL,4
014D D3E8                  404             SHR     AX,CL   ; GET RID OF LOWER 4 BITS OF OFFSET
014F 03D0                  405             ADD     DX,AX   ; HAVE PARAGRAPH NUMBER IN DX
0151 81E70F00              406             AND     DI,0FH  ; ISOLATE 4 LOWER BITS OF OFFSET
                           407     ;
                           408     ; NOW SEE WHAT IDS IT IS IN
                           409     ;
0155 BE0000          E     410             MOV     SI,OFFSET DGROUP:CQMIPIDSBASES
0158 B90800                411             MOV     CX,8    ; COUNTER FOR IDSS
015B 33DB                  412             XOR     BX,BX   ; IDS NUMBER BEING LOOKED AT
                           413     ;
                           414     ; DO SEARCH
                           415     ;
015D 8B00                  416     S1:     MOV     AX,[SI+BX] ; GET BASE AND LENGTH OF CURRENT IDS
015F 02E0                  417             ADD     AH,AL   ; FORM BASE (AL) AND BASE+LENGTH (AH)
0161 3AF0                  418             CMP     DH,AL
0163 7204                  419             JB      S2      ; IF BH IS BELOW BASE, THEN NOT IN THIS IDS
0165 3AF4                  420             CMP     DH,AH   ;
0167 720A                  421             JB      S3      ; IF BH IS NOT BELOW BASE+LENGHT, THEN NOT IN THIS IDS
0169                       422     S2:
0169 43                    423             INC     BX      ; NOT IN THIS IDS, GO LOOK AT NEXT ONE
016A 43                    424             INC     BX
016B E2F0                  425             LOOP    S1
                           426     ;
                           427     ; NOT IN ANY IDS, RETURN IDSID = OFFH
                           428     ;
                           429     ; COME HERE ALSO FROM MIPGETADDRESS IN A VER-R-R-Y UNSRUCTURED TECHNIQUE
                           430     ;
                           431     ;
016D B8FFFF                432     S2A:    MOV     AX,0FFFFH       ; RETURNS ES=0FFFFH
0170 8EC0                  433             MOV     ES,AX
0172 C3                    434             RET
0173                       435     S3:
0173 2AF0                  436             SUB     DH,AL   ; IN THIS IDS, MAKE AN OFFSET
                           437     ;
                           438     ; NOW FORM 24 BIT OFFSET FROM WHAT WE HAVE IN DX AND DI
                           439     ;
0175 B104                  440             MOV     CL,4    ; GET HIGH ORDER BITS INTO LOW ORDER PLACES
0177 D3C2                  441             ROL     DX,CL
0179 8BC2                  442             MOV     AX,DX   ; MAKE COPY OF ROTATED DATA
017B 25F0FF                443             AND     AX,0FFF0H ; THIS BECOMES HIGH 12 ORDER BITS OF LOWER 16
```

```
LOC  OBJ                  LINE     SOURCE

017E 0BC7                  444              OR      AX,DI   ; BRING IN 4 LOWER BITS OF LOWER ORDER 16
0180 81E20F00              445              AND     DX,0FH  ; ONLY 4 OF 8 BITS OF HIGHER ORDER 8 BITS
0184 D0EB                  446              SHR     BL,1
0186 8AF3                  447              MOV     DH,BL   ; BRING IN IDS ID
0188 8BD8                  448              MOV     BX,AX   ; LOWER ORDER 16 BITS
018A 8EC2                  449              MOV     ES,DX   ; HIGH ORDER 8 BITS AND IDS
018C C3                    450              RET
                           451
                           452
                           453              PUBLIC  SUBSYSTEMCALL
018D                       454      SUBSYSTEMCALL:
018D 5B                    455              POP     BX      ; RET ADR
018E 58                    456              POP     AX      ; ADDRESS OF ROUTINE TO CALL
018F 53                    457              PUSH    BX      ; RESTORE RET ADR
0190 FFE0                  458              JMP     AX      ; GO TO IT
                           459
----                       460      CODE    ENDS
                           461
                           462              END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```