```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE EDLLIB
OBJECT MODULE PLACED IN :F1:EDLLIB.OBJ
COMPILER INVOKED BY:  PLM80 :F1:EDLLIB.SRC


   1          EDLLIB: DO;

   2   1      MIPSEN: PROCEDURE (SOC, MSG$P, LEN) BYTE EXTERNAL;
   3   2        DECLARE (SOC, MSG$P, LEN) ADDRESS;
   4   2      END MIPSEN;

   5   1      MIPRCV: PROCEDURE (PORT) ADDRESS EXTERNAL;
   6   2        DECLARE PORT BYTE;
   7   2      END MIPRCV;


   8   1      DECLARE EDL$SOCKET LITERALLY '0001H';

   9   1      DECLARE REQ STRUCTURE (
                MIP(12) BYTE,
                PROC    ADDRESS,
                CMD     BYTE,
                RESULT  BYTE,
                RESPSOC ADDRESS,
                DAT(4)  ADDRESS);

  10   1      DECLARE REQ$DATB(2) BYTE AT (.REQ.DAT);

  11   1      DECLARE MIP$STATUS BYTE;


  12   1      TRANSACT: PROCEDURE (CMD);
  13   2        DECLARE CMD BYTE;
  14   2        REQ.CMD = CMD;
  15   2        REQ.PROC = 0;
  16   2        REQ.RESPSOC = 0304H;
  17   2        MIP$STATUS = MIPSEN (EDL$SOCKET, .REQ, 26);
  18   2        IF MIP$STATUS = 80H OR MIP$STATUS = 82H
                  THEN MIP$STATUS = 0;
  20   2          ELSE RETURN;
  21   2        DO WHILE MIP$RCV (4) <> .REQ;
  22   3        END;
  23   2      END TRANSACT;


  24   1      DECLARE POINTER   LITERALLY '(2) ADDRESS';
  25   1      DECLARE WORD      LITERALLY 'ADDRESS';
  26   1      DECLARE COMM$PORT LITERALLY '0A4H';
  27   1      DECLARE (OUTRQD, INRQD) BYTE EXTERNAL;

  28   1      DECLARE PRES STRUCTURE (
                  CMD     BYTE,
                  RESP    BYTE,
                  DIAG    BYTE) AT (0F690H);

  29   1      DECLARE LGO STRUCTURE (
```

EDL

```
                CMD         BYTE,
                RESP        BYTE,
                FROM        POINTER,
                DEST        POINTER,
                LEN         WORD,
                START       WORD,

                DEVCNT      BYTE,
                IDSCNT      BYTE,
                THISDEV     BYTE,
                RSRD        BYTE,

                IDSBASE     BYTE,
                IDSSIZE     BYTE,

                DEVID       BYTE,
                STATUS      BYTE,
                RQDtoCB     POINTER,
                RQCfromCB   POINTER,
                INTTYPE     BYTE,
                WAITTIME    BYTE,
                INTADR      WORD)  AT (0F690H);

30   1      DECLARE LGO$DATA STRUCTURE (
                CMD         BYTE,
                RESP        BYTE,
                FROM        POINTER,
                DEST        POINTER,
                LEN         WORD,
                START       WORD,

                DEVCNT      BYTE,
                IDSCNT      BYTE,
                THISDEV     BYTE,
                RSRD        BYTE,

                IDSBASE     BYTE,
                IDSSIZE     BYTE,

                DEVID       BYTE,
                STATUS      BYTE,
                RQDtoCB     POINTER,
                RQCfromCB   POINTER,
                INTTYPE     BYTE,
                WAITTIME    BYTE,
                INTADR      WORD) DATA (
                2, 0, .NULLCODE,0, 3000H,0FC00H, 1, 3000H,
                1, 1, 0, 0,
                0, 16,
                3, 0FFH, .OUTRQD,0, .INRQD,0, 0, 0FFH, 2324H);

31   1      DECLARE NULLCODE BYTE DATA (0C3H); /* 8086 RETURN INSTRUCTION */

32   1      CQ$STRT: PROCEDURE (STATUS$P) PUBLIC;
33   2        DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
34   2        DECLARE (I,J) ADDRESS;
```

```
                /* RESET COMM BOARD */

  35    2       OUTPUT (COMM$PORT) = 1;
  36    2       DO I = 0 TO 65000; J=J+I; END;  /* TIME MUST BE 1.5 SECONDS OR MORE */

                /* ISSUE PRESENCE COMMAND */

  39    2       PRES.CMD = 1;             /* INITIALIZE CONTROL BLOCK */
  40    2       PRES.RESP = 0;
  41    2       OUTPUT (COMM$PORT) = 2;  /* GENERATE INTERRUPT */
  42    2       DO I = 0 TO 100;
  43    3         DO J = 0 TO 65000;
  44    4           IF PRES.RESP <> 0 THEN GOTO L1;
  46    4         END;
  47    3       END;
  48    2       STATUS = 1 /* NOT PRESENT */;
  49    2       RETURN;
  50    2     L1:

                /* ISSUE LOAD AND GO COMMAND WITH JUST A RETURN */

                CALL MOVE (SIZE(LGO), .LGO$DATA, .LGO);  /* INITIALIZE CONTROL BLOCK */
  51    2       OUTPUT (COMM$PORT) = 2;
  52    2       DO I = 0 TO 100;        /* WAIT FOR REPLY */
  53    3         DO J = 0 TO 65000;
  54    4           IF LGO.RESP <> 0 THEN GOTO L2;
  56    4         END;
  57    3         END;
  58    2         STATUS = 1 /* NOT PRESENT */;
  59    2         RETURN;
  60    2     L2:

                  STATUS = 0; /* WAIT ABOUT OTHER STATUSES? */
  61    2     END CQ$STRT;


  62    1     CQ$CONN: PROCEDURE (TYPE, STATUS$P) PUBLIC;
  63    2       DECLARE TYPE ADDRESS;
  64    2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
  65    2       REQ$DATB(0) = HIGH (TYPE);
  66    2       REQ$DATB(1) = LOW(TYPE);
  67    2       CALL TRANSACT (1);
  68    2       IF MIP$STATUS <> 0
                    THEN STATUS = MIP$STATUS;
  70    2         ELSE STATUS = REQ.RESULT;
  71    2     END CQ$CONN;


  72    1     CQ$DISC: PROCEDURE (TYPE, STATUS$P) PUBLIC;
  73    2       DECLARE TYPE ADDRESS;
  74    2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
  75    2       REQ$DATB(0) = HIGH (TYPE);
  76    2       REQ$DATB(1) = LOW (TYPE);
  77    2       CALL TRANSACT (2);
  78    2       STATUS = MIP$STATUS;
  79    2     END CQ$DISC;
```

```
 80   1      CQ$AMID: PROCEDURE (MCID$P, STATUS$P) PUBLIC;
 81   2        DECLARE MCID$P ADDRESS;
 82   2        DECLARE MCID BASED MCID$P (3) ADDRESS;
 83   2        DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
 84   2        REQ.CAT(0) = MCID(0);
 85   2        REQ.DAT(1) = MCID(1);
 86   2        REQ.DAT(2) = MCID(2);
 87   2        CALL TRANSACT (3);
 88   2        IF MIP$STATUS <> 0
                  THEN STATUS = MIP$STATUS;
 90   2          ELSE STATUS = REQ.RESULT;
 91   2      END CQ$AMID;


 92   1      CQ$DMID: PROCEDURE (MCID$P, STATUS$P) PUBLIC;
 93   2        DECLARE MCID$P ADDRESS;
 94   2        DECLARE MCID BASED MCID$P (3) ADDRESS;
 95   2        DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
 96   2        REQ.DAT(0) = MCID(0);
 97   2        REQ.DAT(1) = MCID(1);
 98   2        REQ.DAT(2) = MCID(2);
 99   2        CALL TRANSACT (4);
100   2        STATUS = MIP$STATUS;
101   2      END CQ$DMID;


102   1      CQ$XMIT: PROCEDURE (BUF$P, STATUS$P) PUBLIC;
103   2        DECLARE BUF$P ADDRESS;
104   2        DECLARE BUF BASED BUF$P STRUCTURE (
                  MIP(12) BYTE,
                  PROC    ADDRESS,
                  CMD     BYTE,
                  RESULT  BYTE,
                  RESPSOC ADDRESS,
                  LEN     ADDRESS);
105   2        DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
106   2        BUF.PROC = 0;
107   2        BUF.CMD = 5;
108   2        BUF.RESPSOC = 03C5H;
109   2        STATUS = MIPSEN (EDL$SOCKET, BUF$P, BUF.LEN+26);
110   2        IF STATUS = 80H OR STATUS = 82H THEN STATUS = 0;
112   2      END CQ$XMIT;


113   1      CQ$CKTX: PROCEDURE (STATUS$P) ADDRESS PUBLIC;
114   2        DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
115   2        DECLARE BUF$P ADDRESS;
116   2        DECLARE BUF BASED BUF$P STRUCTURE (
                  MIP(12) BYTE,
                  PROC    ADDRESS,
                  CMD     BYTE,
                  RESULT  BYTE,
                  RESPSOC ADDRESS,
                  LEN     ADDRESS);
117   2        BUF$P = MIPRCV (C5H);
118   2        IF BUF$P <> 0 THEN STATUS = BUF.RESULT;
```

```
120   2       RETURN BUF$P;
121   2     END CQ$CKTX;


122   1     CQ$SBUF: PROCEDURE (BUF$P, STATUS$P) PUBLIC;
123   2       DECLARE BUF$P ADDRESS;
124   2       DECLARE BUF BASED BUF$P STRUCTURE (
                  MIP(12) BYTE,
                  PROC    ADDRESS,
                  CMD     BYTE,
                  RESULT  BYTE,
                  RESPSOC ADDRESS);
125   2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
126   2       BUF.PROC = 0;
127   2       BUF.CMD = 6;
128   2       BUF.RESPSOC = 0306H;
129   2       STATUS = MIPSEN (EDL$SOCKET, BUF$P, 1514+20);
130   2       IF STATUS = 80H OR STATUS = 82H THEN STATUS = 0;
132   2     END CQ$SBUF;


133   1     CQ$CKRX: PROCEDURE (STATUS$P) ADDRESS PUBLIC;
134   2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
135   2       STATUS = 0;
136   2       RETURN MIPRCV (06H);
137   2     END CQ$CKRX;


138   1     CQ$READ: PROCEDURE (OBJECT, RET$P, STATUS$P) PUBLIC;
139   2       DECLARE (OBJECT, RET$P) ADDRESS;
140   2       DECLARE RET BASED RET$P (3) ADDRESS;
141   2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
142   2       REQ.DAT(0) = OBJECT;
143   2       CALL TRANSACT (8);
144   2       STATUS = MIP$STATUS;
145   2       RET(0) = REQ.DAT(1);
146   2       RET(1) = REQ.DAT(2);
147   2       RET(2) = REQ.DAT(3);
148   2     END CQ$READ;


149   1     CQ$RDCL: PROCEDURE (OBJECT, RET$P, STATUS$P) PUBLIC;
150   2       DECLARE (OBJECT, RET$P) ADDRESS;
151   2       DECLARE RET BASED RET$P (3) ADDRESS;
152   2       DECLARE STATUS$P ADDRESS, STATUS BASED STATUS$P ADDRESS;
153   2       REQ.DAT(0) = OBJECT;
154   2       CALL TRANSACT (9);
155   2       STATUS = MIP$STATUS;
156   2       RET(0) = REQ.DAT(1);
157   2       RET(1) = REQ.DAT(2);
158   2       RET(2) = REQ.DAT(3);
159   2     END CQ$RDCL;


160   1     END EDLLIB;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 043FH   1087D
     VARIABLE AREA SIZE = 004CH     76D
     MAXIMUM STACK SIZE = 0006H      6D
     277 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION