/**

SPOOL
MNML.LST
05/04/82
15:55:01

**/

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE NML
OBJECT MODULE PLACED IN :F1:MNML.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:MNML.SRC OPTIMIZE(3) XREF SET(F1) DEBUG


            $COMPACT OPTIMIZE(3)
*** WARNING 10 IN 1 (LINE 1): RESPECIFIED PRIMARY CONTROL, IGNORED

                        /*************************************************************
                        ********** last update Time 13:00 Date 11:06:81 ***********
                        *************************************************************/

            $IF f7
            $INCLUDE (:F7:cpyrt.dcp)
            $ELSE
            $INCLUDE (:F1:cpyrt.dcp)

        =           /* Intel Corporation Proprietary Information.
        =           This listing is supplied under the terms of a
        =           license agrement with Intel Corporaton and
        =           may not be copied nor disclosed except in
        =           accordance with the terms of that agreement. */

            $ENDIF


  1         NML: DO;
            $nolist
            $IF f7
            $include (:f7:Mstrb.inc)
            $include (:f7:Mopnrb.inc)
            $ELSE
            $include (:f1:Mstrb.inc)
        =   /* type declaration for standard tcl request block : 041081 10:25 */

126   1   =   declare st$rb$type literally
        =   'structure(mippart(7) word,req BYTE,
        =   resp BYTE,rtn$mip$skt WORD,link POINTER,CID WORD,seq WORD,client$use WORD,
        =   buf$len WORD,num$blks BYTE,
        =   BLK1$ptr pointer,Blk1$len word)';

        =   /****  Request Block for TCL Standard requests  ***  03/01/81 ***

        =           mip$ptr                 POINTER,  **  0   **
        =           mip$buf$base            POINTER,  **  4   **
        =           mip$length              WORD,     **  8   **
        =           mip$ids$id              BYTE,     **  A = 10T   **
        =           mip$owner$dev$id        BYTE,     **  B = 11T   **
        =           internal$process$id     WORD,     **  C = 12T for failure handler, not SCL process ID   **
        =           req                     BYTE,     **  E = 14T  Code for type of request   **
        =           resp                    BYTE,     **  F = 15T  reponse code: ok or error type   **
        =           rtn$mip$skt             WORD,     ** 10 = 16T  return address: CMX mbx or MIP socket  **
        =           link                    POINTER,  ** 12 = 18T  optional chain to another RB  **
        =           CID                     WORD,     ** 16 = 22t  returned by open processing   **
        =           seq                     WORD,     ** 18 = 24t  reserved for TCL   **
        =           client$use              WORD,     ** 1A  reserved for client use  **
```

```
=           buf$len             WORD,     ** 1C = 28t  total no of client data bytes   **
=           num$blks            BYTE,     ** 1E = 30t  number of data blocks   **
=           Blk1$ptr            POINTER,  ** 1F = 31t  Block 1 pointer
=           Blk1$len            WORD,     ** 23 = 35t  Block 1 length
=                                            25 = 37t  standard request block size
=
=   */
    $include (:f1:Mopnrb.inc)
=   /* Type declaration for open request blocks : 041081 10:45 */

127  1  =   declare tclpart literally
=       'req BYTE,resp BYTE,rtn$mip$skt WORD,link POINTER,
=       CID WORD,loc$port WORD,rem$net WORD,rem$host(3)WORD,
=       rem$port WORD,persist WORD,abort$timeout WORD,seq WORD',

=       open$rb$type literally
=       'structure(mippart(7)word,tclpart)';

=       /******  Request Block for the TCL Open request  ******************************

=           cmx$ptr          POINTER,     **  0 =   0 for cmx/mip to use  **
=           mip$buf$base     POINTER,     **  4 = 04T :    **
=           mip$length       WORD,        **  8 = 08T :    **
=           mip$ids$id       BYTE,        **  A = 10T :  MIP fields: required for  **
=           mip$owner$dev$id BYTE,        **  B = 11T :  any comm subsystem  **
=           owner$process$id WORD,        **  C = 12T :    **
=           req              BYTE,        **  E = 14T Code for type of request  **
=           resp             BYTE,        **  F = 15T reponse code: ok or error type  **
=           rtn$mip$skt      WORD,        ** 10 = 16T return address: kaos mbx or MIP socket **
=           link             POINTER,     ** 12 = 18t to rb tcl keeps for abort signaling  **
=           CID              WORD,        ** 16 = 22t cid returned by open processing  **
=           loc$port         WORD,        ** 18 = 24t local port for connection  **
=           rem$net          WORD,        ** 1A = 26t remote net id  :           **
=           rem$host(3)      WORD,        ** 1C = 28t remote host id     : socket     **
=           rem$port         WORD,        ** 22 = 34t remote port id :           **
=           persist          WORD,        ** 24 = 36t no times to ignore remote connect refusal  **
=           abort$timeout    WORD,        ** 26 = 38t optional abort timeout value  **
=           seq              WORD,        ** 28 = 40t reserved for TCL  **
=                                         ** 2A = 42t size of open request block
=    */
    $ENDIF

128  1      DCL
            NMLmbx1 byte external,  /* All commands from MIP come to this port */
            NMLmbx2 byte external,  /* Buffers supplied for boot are queued here */
            NMLmbx3 byte external,  /* Buffers supplied by NCP and others are queued herea */
            NMLmbx4 byte external,

            NML$SOCKET          LITERALLY '0002H',
            NML$SCR$SOCKET      LITERALLY '0003H',
            TCL$SOCKET          LITERALLY '0004H',

            NML$PORT            LITERALLY '0003H',  /* TRANSPORT port */


            READ$TYPE           LITERALLY '0',
            READC$TYPE          LITERALLY '1',
```

```
                SET$TYPE            LITERALLY '2',
                CAUSE_EVENT$TYPE    literally '3',
                READ$MEMORY$TYPE    Literally '4',
                SET$MEMORY$TYPE     literally '5';

129   1         dcl  i byte,dummy word;

130   1         VECTOR: procedure (A,B,C, V) byte external;
131   2           DECLARE (A,B, V) word, C pointer;
132   2         END VECTOR;



133   1         LINE$WATCHER: procedure (PACKET$O) PUBLIC;
                                /*************************************************************
                                LINE$WATCHER:

                                The address of this routine is passed to BOOT via CQBOOTREGISTER
                                call.  Whenever boot receives a net management type packet, it
                                calls LINE$WATCHER process if it does not know what to do with
                                it.   Two types of buffers may be supplied to NET management
                                Layer.  One by the bootstrap server, and other by NCP and other
                                utilities.  The Bootstrap server expects two types of packets:
                                        BOOT$REQUEST (cmd = 04) and
                                        BOOT$DATA$REQUEST (cmd = 06h).
                                If the received packet is one of the above, it copies it to a
                                bootstrap server supplied buffer, otherwise it copies it to the
                                NCP supplied buffer.   If no buffer is supplied the incoming
                                packet is ignored.  In any case the buffer is returned to
                                DATA LINK receive buffer pool.
                                *************************************************************/

134   2           DCL packet$o word,
                      packet based packet$o structure (
                              KAOS          pointer,
                              LEN           word,
                              DLDEST(3)     word,
                              DLSRC(3)      word,
                              DLTYPE        word,
                              BOOTCMD       byte,           /** Boot COmmand Field **/
                              STUFF(1499)   byte),          /** First byte used for Bootcmd **/

                      BOOT$REQ literally '4',               /** Defined in BOOT EPS **/
                      BOOT$DATA$REQ literally '6';          /** Defined in BOOT EPS **/

135   2           COPY: procedure (MBX);

                      /****  Copies contents of received packet to buffer lying on
                      specified mailbox.  Ignore if buffer not available.    ****/

136   3               DCL MBX word, (BUF$O, BUF$F) word, BUF$P pointer AT (@BUF$O),
                          BUF BASED BUF$P STRUCTURE(
                              MIP(12)     byte,
```

```
                        PROCID      word,
                        CMD         byte,
                        RESULT      byte,
                        RESP$SOC    word,
                        LEN         word,
                        STUFF(1514) byte);

137   3             BUF$P = CQ$CRECEIVE (MBX);
138   3             IF BUF$O <> OFFFFH then do;
140   4               CALL MOVB (@PACKET.LEN, @BUF.LEN, PACKET.LEN+2);
141   4               CALL CQMIP$SEND (BUF.RESP$SOC, BUF$P);
142   4             ENDIF;

143   3         END COPY;


144   2         If packet.bootcmd = BOOT$REQ or packet.bootcmd = BOOT$DATA$REQ
145   2                 then call COPY (.NMLmbx2);  /** copy into bootserver buffer **/
146   2         else call copy (.NMLmbx3);              /** copy into NCP buffer **/

147   2         CALL CQDLLRXRETBUF (PACKET$O);      /** return packet to RBP **/

148   2       END LINE$WATCHER;



149   1       EXECUTE: procedure (REQ$PTR, RESP$PTR, RESP$MAX, RESP$LEN$O);
                                  /*************************************************************
                                  EXECUTE:
                                  This routine process the REQUEST at req$ptr and puts the
                                  result in the area pointed by RESP$PTR.  The request can
                                  be READ, READC or SET.  The objects, modifier, len, value
                                  fields are as described in the NML EPS.  req$ptr should
                                  point to the type filed of the NML request block.

                                  EXECUTE is called by NML - the command processor for local
                                  commands and by the remote executor when a remote command
                                  is received.
                                  *************************************************************/

150   2         dcl (REQ$PTR, RESP$PTR) pointer, (RESP$MAX, RESP$LEN$O) word,

                  RESP$LEN BASED RESP$LEN$O word,

                  (REQ$O, REQ$F) word AT (@REQ$PTR),      /** Will increment req$o **/

                  Type$n$num based req$ptr structure (type byte, num byte),

                  FUNCT  byte, modifier word, len byte,
                  rmng$buf$len word,
                  object word, (object$l, object$h) byte at (@object),

                  REQ    BASED REQ$PTR STRUCTURE (
                             OBJECT   word,
                             MODIFIER word,
```

```
                        LEN     byte,
                        VALUE(1) byte),

                    RESP BASED RESP$PTR STRUCTURE (
                        OBJECT   word,
                        MODIFIER word,
                        LEN     byte,
                        VALUE(1) byte),

                    (RESP$O,  RESP$F)  word AT (@RESP$PTR), /** Will increment resp$o **/

                    svd$resp$PTR pointer,       /**  Will point to beginning of RESP area **/
                    (svd$resp$O, svd$resp$F) word AT (@svd$resp$PTR),

                    RESP$num BASED svd$resp$PTR byte;

151   2         DECLARE NUM$OBJECTS byte,

                RM$req based req$ptr structure (
                    type byte,
                    pointr pointer,
                    len byte),

                Set$req based req$ptr structure (
                    type byte,
                    pointr pointer,
                    len byte,
                    value (1) byte),

                  Event$req based req$ptr structure (
                      type byte,
                      event word,
                      params (1) byte),

                  Event$resp based resp$ptr byte;

152   2       dcl LAYER(2) STRUCTURE (FUNCT(3) word) DATA (
                              .CQDLLREAD,   .CQDLLREADC,  .CQDLLREAD,
                              .TCLREAD,     .TCLREADC,    .TCLSET);



153   2         resp$len = 0;
154   2         i = RM$req.len;

155   2         if (funct := type$n$num.type) = READ$MEMORY$TYPE then do;
157   3           if i >= resp$max then i = resp$max - 1;
159   3           call movb (RM$req.pointr, resp$ptr, i);
160   3           resp$len = i;
161   3         end;

162   2         else if funct = SET$MEMORY$TYPE then
163   2           call movb (@SET$req.value, SET$req.pointr, SET$req.len);

164   2         else if funct = CAUSE_EVENT$TYPE and Event$req.event = 0101h then do;
166   3           Event$resp = Add$CDB$Memory (@Event$req.params);
167   3           resp$len = 1;
```

```
168   3         end;

169   2         else if FUNCT <= SET$TYPE then do;

171   3           svd$resp$PTR = RESP$PTR;            /* Save resp$ptr */
172   3           resp$num = 0;
173   3           RESP$O = RESP$O + 1;                /* begin object responses here */

174   3           NUM$OBJECTS = type$n$num.num;
175   3           REQ$O = REQ$O + 2;

176   3           do while num$objects > 0;
                          /*****************************************************
                          Processes  Layer  Objects.  Makes an Appropriate
                          CALL to the Layer specified in the Object.  The High
                          byte of the Object is 0 for DLL Objects and 1 for
                          TCL Objects.
                          *****************************************************/


177   4               object = req.object;
178   4               modifier = req.modifier;
179   4               len = req.len;

180   4               rmng$buf$len = resp$max - (resp$o - svd$resp$o + 5);
181   4               If object$h > 2             /* Illegal object */
                       or rmng$buf$len < 6        /* Less than 6 bytes left in resp buffer */
                       or len > 6                 /* Invalid length in request */
                       or (object = 100h and rmng$buf$len < maxcdbs)
                          /*
                          special case for CIDVECTOR - the buffer len should not be less
                          than the size of the CIDVECTOR
                          left in the response buffer.
                      */
182   4               then return;

183   4               if len <> 0 then call movb (@req.value,  @resp.value, len);

185   4               REQ$O = len + REQ$O + 5;

186   4               len = VECTOR (OBJECT$l, MODIFIER,  @resp.value,
                          Layer (object$h). funct (funct));
187   4               if len <> 0 then do;
189   5                   Resp.len = len;
190   5                   RESP.OBJECT = OBJECT;
191   5                   resp.modifier = modifier;
192   5                   RESP$O = len + 5 + RESP$O;
193   5                   resp$num = resp$num + 1;
194   5               ENDIF;

195   4               NUM$OBJECTS = NUM$OBJECTS - 1;
196   4           end;    /* do */

197   3           RESP$LEN = RESP$O - svd$resp$O;

198   3         end;
199   2       END EXECUTE;
```

```
200   1     NML: procedure PUBLIC;
                                  /************************************************
                                  NML -  COMMAND PROCESSOR
                                  ************************************************/

201   2       DECLARE REQ$PTR pointer;
202   2       DECLARE REQ BASED REQ$PTR STRUCTURE (
                  MIP(12)  byte,
                  PROCID   address,
                  CMD   byte,
                  RESULT   byte,
                  RESP$SOC address,
                  RESP$PTR pointer,
                  RESP$MAX address,
                  TYPE     byte,
                  NUM   byte,
                  OBJECT   address,
                  MODIFIER address,
                  LEN   byte,
                  VALUE(3) address);

203   2     dcl resp$ptr pointer;

204   2         TAKEBACK: procedure (MBX);
205   3             DECLARE MBX word, (BUF$O, BUF$F) word, BUF$P pointer AT (@BUF$O);

206   4             LOOP;
208   4               BUF$P = CQ$CRECEIVE (MBX);
209   4               IF BUF$O = OFFFFH THEN EXITLOOP;
211   4               CALL CQMIP$SEND (REQ.RESP$SOC, BUF$P);
212   4             ENDLOOP;
214   3             CALL CQMIP$SEND (REQ.RESP$SOC, @REQ);

215   3         END TAKEBACK;


216   2       CALL CQMIP$CONNECT (LOW(NML$SOCKET), .NMLMBX1);
217   2       CALL CQ$BOOTREGISTER (.LINE$WATCHER);
218   2       DO FOREVER;
219   3         REQ$PTR = CQRECEIVE (.NMLMBX1);
220   3         REQ.RESULT = 0;    /* DEFAULT TO SUCCESS */
221   3         i = req.cmd;

222   3         if i = 1 then
                              /*** READ, READC or  SET  request  ***/
223   3             do;
224   4               if not get$chk$address (req.resp$ptr, .resp$ptr) then
                                  /* BAD response pointer - return req block */
225   4                     req.result = 1;
226   4               else CALL EXECUTE (@REQ.TYPE, RESP$PTR, REQ.RESP$MAX, .DUMMY);
227   4               CALL CQMIP$SEND (REQ.RESP$SOC, REQ$PTR);
228   4             end;

229   3         else if i = 2 then
                              /*** SUPPLYBUF for Boot strap server ***/
```

```
230   3             CALL CQSEND (.NMLMBX2, @REQ);

231   3         else if i = 3 then
                                /*** SupplyBuf for NCP ***/
232   3             CALL CQSEND (.NMLMBX3, @REQ);

233   3         else if i = 4 then
                                /*** TAKEBACK bootstrap server buffers ***/
234   3             CALL TAKEBACK (.NMLMBX2);

235   3         else if i = 5 then
                                /*** TAKEBACK NCP buffers ***/
236   3             CALL TAKEBACK (.NMLMBX3);

237   3         else do;
                                /*** Invalid Command - return 1 ***/
238   4             req.result = 1;
239   4             call CQ$MIPSEND (req.resp$soc, req$ptr);
240   4         end;

241   3       ENDDO;
242   2     END NML;



243   1       DECLARE TCL$OPENP     LITERALLY '1';
244   1       DECLARE TCL$CLOSE     LITERALLY '2';
245   1       DECLARE TCL$SEND      LITERALLY '5';
246   1       DECLARE TCL$SEND$EOM  LITERALLY '6';
247   1       DECLARE TCL$POST$RBUF LITERALLY '7';
248   1       DECLARE TCL$ABORT     LITERALLY '8';

249   1       Declare Open$RB open$rb$type initial (
                  0, 0, 0, 0, 0, 0, 0,          /* mip part and process ID */
                  0, 0, NML$SCR$SOCKET, 0, 0,   /* Req, resp, rtn$socket, link, cid */
                  0, 0, 0, 0, 0, 0,             /* loc$port, rem$net, rem$host, rem$port */
                  0, 0, 0);                     /* persist, abort, seq */

250   1       Declare TCL$rb st$rb$type at (@Open$rb);

251   1     REMOTE$EXECUTOR: procedure PUBLIC;
                                /*************************************************
                                REMOTE EXECUTOR:
                                Processes READ, READC and SET commands received over TCL from
                                a remote node.  It does an open passive to TCL.  When anything
                                is received it executes it and sends the results back.
                                It then aborts the connection and reopens another
                                *************************************************/



252   2         declare resp$ptr pointer;

253   2         DECLARE RCV$BUFL  LITERALLY '100';
254   2         DECLARE XMIT$BUFL LITERALLY '200';
255   2         DECLARE RCV$BUF   (RCV$BUFL) byte;
```

```
256   2       DECLARE XMIT$BUF (XMIT$BUFL) byte;

257   2       CALL CQMIP$CONNECT (LOW(NML$SCR$SOCKET), .NMLMBX4);

258   2       DO FOREVER;

259   3         call setw (0,@Open$RB.cid,9);          /* Zero cid field to abort$timeout field */
260   3         Open$RB.Req = TCL$OPENP;
261   3         Open$RB.loc$port = NML$PORT;

262   3         CALL CQMIP$SEND (TCL$SOCKET, @Open$RB);     /* Passive Open Request for any */
263   3         resp$ptr = CQ$RECEIVE (.NMLMBX4);           /* remote node and port */

264   3         TCL$rb.req = TCL$POST$RBUF;            /* Post Receive Buffer command  */
265   3         TCL$rb.num$blks = 1;
266   3         TCL$rb.BLK1$PTR = CQ$MIPgetmipform (@RCV$BUF); /* Translate address to mip form */
267   3         TCL$rb.BLK1$LEN = RCV$BUFL;            /* Buffer Length */
268   3         CALL CQMIP$SEND (TCL$SOCKET, @TCL$rb);
269   3         resp$ptr = CQ$RECEIVE (.NMLMBX4);           /* Wait till a remote command is rcvd */


270   3         If not TCL$rb.resp or (TCL$rb.buf$len = 0) then TCL$rb.num$blks = 0;
272   3         else    /* If good rcv then execute command */
                    CALL EXECUTE (@RCVBUF, @XMITBUF, XMIT$BUFL, .TCL$rb.BLK1$len);

273   3         TCL$rb.req = TCL$Close;             /* SEND the result and close connection*/
274   3         TCL$rb.BLK1$PTR = CQ$MIPgetmipform (@XMITBUF);
275   3         CALL CQMIP$SEND (TCL$SOCKET, @TCL$rb);
276   3         resp$ptr = CQ$RECEIVE (.NMLMBX4);       /* Wait till SEND-close complete */
                    /* GO DO another Passive Open */
277   3       ENDDO;

278   2     END REMOTE$EXECUTOR;

279   1     END NML;
```

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES					
111	0000H	2	A.	WORD IN PROC (CQDLLSET) PARAMETER	111			
114	0000H	2	A.	WORD IN PROC (TCLREAD) PARAMETER	114			
108	0000H	2	A.	WORD IN PROC (CQDLLREADC) PARAMETER	108			
117	0000H	2	A.	WORD IN PROC (TCLREADC) PARAMETER	117			
105	0000H	2	A.	WORD IN PROC (CQDLLREAD) PARAMETER	105			
120	0000H	2	A.	WORD IN PROC (TCLSET) PARAMETER	120			
131	0000H	2	A.	WORD IN PROC (VECTOR) PARAMETER	131			
123	0000H		ADDCDBMEMORY . . .	PROCEDURE BYTE EXTERNAL(37) STACK=0000H				166
69	0000H	4	ALARMP	POINTER IN PROC (CQCREATEALARM) PARAMETER				69
72	0000H	4	ALARMP	POINTER IN PROC (CQSETALARM) PARAMETER				72
78	0000H	4	ALARMP	POINTER IN PROC (CQCLEARALARM) PARAMETER				78
75	0000H	4	ALARMP	POINTER IN PROC (CQCHECKALARM) PARAMETER				75
131	0000H	2	B.	WORD IN PROC (VECTOR) PARAMETER	131			
117	0000H	2	B.	WORD IN PROC (TCLREADC) PARAMETER	117			
114	0000H	2	B.	WORD IN PROC (TCLREAD) PARAMETER	114			
108	0000H	2	B.	WORD IN PROC (CQDLLREADC) PARAMETER	108			
111	0000H	2	B.	WORD IN PROC (CQDLLSET) PARAMETER	111			
120	0000H	2	B.	WORD IN PROC (TCLSET) PARAMETER	120			
105	0000H	2	B.	WORD IN PROC (CQDLLREAD) PARAMETER	105			
134			BOOTDATAREQ. . . .	LITERALLY '6' IN PROC (LINEWATCHER)	144			
134			BOOTREQ.	LITERALLY '4' IN PROC (LINEWATCHER)	144			
136	0000H	1534	BUF.	STRUCTURE BASED(BUFP) IN PROC (COPY)				
	0000H	12	MIP.	BYTE ARRAY(12)				
	000CH	2	PROCID	WORD				
	000EH	1	CMD.	BYTE				
	000FH	1	RESULT	BYTE				
	0010H	2	RESPSOC. . . .	WORD 141				
	0012H	2	LEN.	WORD 140				
	0014H	1514	STUFF.	BYTE ARRAY(1514)				
205	001CH	2	BUFF	WORD IN PROC (TAKEBACK)				
136	0006H	2	BUFF	WORD IN PROC (COPY)				
136	0004H	2	BUFO	WORD IN PROC (COPY)	136	138		
205	001AH	2	BUFO	WORD IN PROC (TAKEBACK)	205	209		
205	001AH	4	BUFP	POINTER IN PROC (TAKEBACK) AT	208*	211		
136	0004H	4	BUFP	POINTER IN PROC (COPY) AT	137*	140	141	
114	0000H	4	C.	POINTER IN PROC (TCLREAD) PARAMETER	114			
131	0000H	4	C.	POINTER IN PROC (VECTOR) PARAMETER	131			
117	0000H	4	C.	POINTER IN PROC (TCLREADC) PARAMETER	117			
108	0000H	4	C.	POINTER IN PROC (CQDLLREADC) PARAMETER				108
120	0000H	4	C.	POINTER IN PROC (TCLSET) PARAMETER	120			
105	0000H	4	C.	POINTER IN PROC (CQDLLREAD) PARAMETER				105
111	0000H	4	C.	POINTER IN PROC (CQDLLSET) PARAMETER	111			
128			CAUSE_EVENTTYPE. .	LITERALLY '3' 164				
135	002DH	68	COPY	PROCEDURE IN PROC (LINEWATCHER) STACK=000CH			145	146
91	0000H		CQBOOTREGISTER . .	PROCEDURE EXTERNAL(26) STACK=0000H	217			
74	0000H		CQCHECKALARM . . .	PROCEDURE BYTE EXTERNAL(21) STACK=0000H				
77	0000H		CQCLEARALARM . . .	PROCEDURE EXTERNAL(22) STACK=0000H				
68	0000H		CQCREATEALARM. . .	PROCEDURE EXTERNAL(19) STACK=0000H				
15	0000H		CQCREATELIST . . .	PROCEDURE EXTERNAL(1) STACK=0000H				
38	0000H		CQCREATEMAILBOX. .	PROCEDURE EXTERNAL(9) STACK=0000H				
23	0000H		CQCREATEPROCESS. .	PROCEDURE EXTERNAL(4) STACK=0000H				
26	0000H		CQCREATESEMAPHORE.	PROCEDURE EXTERNAL(5) STACK=0000H				

```
 47  0000H        CQCRECEIVE . . . .   PROCEDURE POINTER EXTERNAL(12) STACK=0000H     137  208
 35  0000H        CQCWAIT. . . . . .   PROCEDURE BYTE EXTERNAL(8) STACK=0000H
104  0000H        CQDLLREAD. . . . .   PROCEDURE EXTERNAL(30) STACK=0000H       152
107  0000H        CQDLLREADC . . . .   PROCEDURE EXTERNAL(31) STACK=0000H       152
 80  0000H        CQDLLRXRETBUF. . .   PROCEDURE EXTERNAL(23) STACK=0000H       147
110  0000H        CQDLLSET . . . . .   PROCEDURE EXTERNAL(32) STACK=0000H
 20  0000H        CQHALTANDCATCHFIRE   PROCEDURE EXTERNAL(3) STACK=0000H
 62  0000H        CQICRECEIVE. . . .   PROCEDURE POINTER EXTERNAL(17) STACK=0000H
 56  0000H        CQICWAIT . . . . .   PROCEDURE BYTE EXTERNAL(15) STACK=0000H
 59  0000H        CQISEND. . . . . .   PROCEDURE EXTERNAL(16) STACK=0000H
 53  0000H        CQISIGNAL. . . . .   PROCEDURE EXTERNAL(14) STACK=0000H
 83  0000H        CQMIPCONNECT . . .   PROCEDURE EXTERNAL(24) STACK=0000H       216  257
100  0000H        CQMIPGETADDRESS. .   PROCEDURE POINTER EXTERNAL(29) STACK=0000H
 94  0000H        CQMIPGETMIPFORM. .   PROCEDURE POINTER EXTERNAL(27) STACK=0000H     266  274
 87  0000H        CQMIPSEND. . . . .   PROCEDURE EXTERNAL(25) STACK=0000H       141  211  214  227  239
                                        262  268  275
 50  0000H        CQMRECEIVE . . . .   PROCEDURE POINTER EXTERNAL(13) STACK=0000H
 65  0000H        CQREADCLOCK. . . .   PROCEDURE EXTERNAL(18) STACK=0000H
 44  0000H        CQRECEIVE. . . . .   PROCEDURE POINTER EXTERNAL(11) STACK=0000H     219  263  269  276

 18  0000H        CQSCHEDULE . . . .   PROCEDURE EXTERNAL(2) STACK=0000H
 41  0000H        CQSEND . . . . . .   PROCEDURE EXTERNAL(10) STACK=0000H       230  232
 71  0000H        CQSETALARM . . . .   PROCEDURE EXTERNAL(20) STACK=0000H
 29  0000H        CQSIGNAL . . . . .   PROCEDURE EXTERNAL(6) STACK=0000H
 13  0000H        CQSTART. . . . . .   PROCEDURE EXTERNAL(0) STACK=0000H
 32  0000H        CQWAITSEM. . . . .   PROCEDURE EXTERNAL(7) STACK=0000H
 66  0000H     2  DATARTNO . . . . .   WORD IN PROC (CQREADCLOCK) PARAMETER      66
  2                DCL. . . . . . .   LITERALLY 'declare'           94   97  100  127  128  133  135
                                        149  151  202
 72  0000H     2  DELAYH . . . . . .   WORD IN PROC (CQSETALARM) PARAMETER       72
 72  0000H     2  DELAYL . . . . . .   WORD IN PROC (CQSETALARM) PARAMETER       72
129  0000H     2  DUMMY. . . . . .   WORD       226
  4                ENDCASE. . . . . .   LITERALLY 'END'
 10                ENDDO. . . . . .   LITERALLY 'END'        240  276
  3                ENDIF. . . . . .   LITERALLY 'END'        141  193
  9                ENDLOOP. . . . . .   LITERALLY 'GOTO LOOPLAB; EXITLAB: END'         211
 24  0000H     2  ENTRYO . . . . . .   WORD IN PROC (CQCREATEPROCESS) PARAMETER       24
 21  0000H     2  ERRORCODE. . . . .   WORD IN PROC (CQHALTANDCATCHFIRE) PARAMETER            21
151  0000H     4  EVENTREQ . . . . .   STRUCTURE BASED(REQPTR) IN PROC (EXECUTE)
      0000H     1      TYPE . . . . .   BYTE
      0001H     2      EVENT. . . . .   WORD       164
      0003H     1      PARAMS . . . .   BYTE ARRAY(1)          166
151  0000H     1  EVENTRESP. . . . .   BYTE BASED(RESPPTR) IN PROC (EXECUTE)          166*
149  0071H   451  EXECUTE. . . . . .   PROCEDURE STACK=001AH        226  272
213  0327H        EXITLAB. . . . . .   LABEL IN PROC (TAKEBACK)     210
  6                EXITLOOP . . . . .   LITERALLY 'GOTO EXITLAB'           209
 12                FOREVER. . . . . .   LITERALLY 'WHILE TRUE'       218  258
150  004DH     1  FUNCT. . . . . .   BYTE IN PROC (EXECUTE)       155*  162  164  169  186
 97  0000H        GETCHKADDRESS. . .   PROCEDURE BYTE EXTERNAL(28) STACK=0000H       224
129  004CH     1  I. . . . . . . . .   BYTE       154*  157  158*  159  160  221*  222  229  231  233
                                        235
101  0000H     4  IDSIDANDPTR. . . .   POINTER IN PROC (CQMIPGETADDRESS) PARAMETER          101
 27  0000H     2  INIT . . . . . . .   WORD IN PROC (CQCREATESEMAPHORE) PARAMETER      27
152  0000H    12  LAYER. . . . . . .   STRUCTURE ARRAY(2) IN PROC (EXECUTE) DATA
      0000H     6      FUNCT. . . . .   WORD ARRAY(3)          186
103                LAYERIF. . . . . .   LITERALLY 'procedure (A,B,C) external;DECLARE (A,B) word, C pointer; END'
```

```
                                        104  107  110  113  116  119
 150  004EH     1  LEN. . . . . .      BYTE IN PROC (EXECUTE)       179*  181  183  184  185  186*  187
                                         189  192
 133  0000H    45  LINEWATCHER. . . .  PROCEDURE PUBLIC STACK=0010H       217
  16  0000H     4  LISTP. . . . . .    POINTER IN PROC (CQCREATELIST) PARAMETER        16
   5             LOOP . . . . . .     LITERALLY 'DO; LOOPLAB:'           205
 207  02FFH        LOOPLAB. . . . . .  LABEL IN PROC (TAKEBACK)     212
                   LOW. . . . . .      BUILTIN          216  257
  72  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQSETALARM) PARAMETER       72
  63  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQICRECEIVE) PARAMETER      63
  48  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQCRECEIVE) PARAMETER       48
  45  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQRECEIVE) PARAMETER       45
  42  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQSEND) PARAMETER          42
  60  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQISEND) PARAMETER         60
  39  0000H     2  MAILBOX0 . . . . .  WORD IN PROC (CQCREATEMAILBOX) PARAMETER       39
 122  0000H     1  MAXCOBS. . . . . .  BYTE EXTERNAL(36)      181
 136  0004H     2  MBX. . . . . .      WORD IN PROC (COPY) PARAMETER AUTOMATIC       136  137
 205  0004H     2  MBX. . . . . .      WORD IN PROC (TAKEBACK) PARAMETER AUTOMATIC          205  208
  84  0000H     2  MBX0 . . . . . .    WORD IN PROC (CQMIPCONNECT) PARAMETER          85
  60  0000H     4  MESSAGEP . . . . .  POINTER IN PROC (CQISEND) PARAMETER       60
  42  0000H     4  MESSAGEP . . . . .  POINTER IN PROC (CQSEND) PARAMETER       42
 150  0008H     2  MODIFIER . . . . .  WORD IN PROC (EXECUTE)       178*  186  191
                   MOVB . . . . . .    BUILTIN          140  159  163  184
  88  0000H     4  MSGPTR . . . . . .  POINTER IN PROC (CQMIPSEND) PARAMETER          89
 200  0234H   200  NML. . . . . .      PROCEDURE PUBLIC STACK=001EH
      0000H        NML. . . . . .      PROCEDURE STACK=0000H
 128  0000H     1  NMLMBX1. . . . . .  BYTE EXTERNAL(38)      216  219
 128  0000H     1  NMLMBX2. . . . . .  BYTE EXTERNAL(39)      145  230  234
 128  0000H     1  NMLMBX3. . . . . .  BYTE EXTERNAL(40)      146  232  236
 128  0000H     1  NMLMBX4. . . . . .  BYTE EXTERNAL(41)      257  263  269  276
 128             NMLPORT. . . . . .   LITERALLY '0003H'      261
 128             NMLSCRSOCKET . . .   LITERALLY '0003H'      249  257
 128             NMLSOCKET. . . . .   LITERALLY '0002H'      216
 151  004FH     1  NUMOBJECTS . . . .  BYTE IN PROC (EXECUTE)       174*  176  195*  195
  98  0000H     2  O. . . . . .        WORD IN PROC (GETCHKADDRESS) PARAMETER          98
 150  000CH     2  OBJECT . . . . . .  WORD IN PROC (EXECUTE)       150  177*  181  190
 150  000DH     1  OBJECTH. . . . . .  BYTE IN PROC (EXECUTE) AT          181  186
 150  000CH     1  OBJECTL. . . . . .  BYTE IN PROC (EXECUTE) AT          186
 249  001EH    42  OPENRB . . . . . .  STRUCTURE INITIAL      250  262
      0000H    14      MIPPART. . . .  WORD ARRAY(7)
      000EH     1      REQ. . . . . .  BYTE       260*
      000FH     1      RESP . . . . .  BYTE
      0010H     2      RTNMIPSKT. . .  WORD
      0012H     4      LINK . . . . .  POINTER
      0016H     2      CID. . . . . .  WORD       259
      0018H     2      LOCPORT. . . .  WORD       261*
      001AH     2      REMNET . . . .  WORD
      001CH     6      REMHOST. . . .  WORD ARRAY(3)
      0022H     2      REMPORT. . . .  WORD
      0024H     2      PERSIST. . . .  WORD
      0026H     2      ABORTTIMEOUT .  WORD
      0028H     2      SEQ. . . . . .  WORD
 127             OPENRBTYPE . . . .   LITERALLY 'structure(mippart(7)word,tclpart)'          249
 124  0000H     4  P. . . . . .        POINTER IN PROC (ADDCDBMEMORY) PARAMETER       124
  98  0000H     4  P. . . . . .        POINTER IN PROC (GETCHKADDRESS) PARAMETER       98
 134  0000H  1520  PACKET . . . . . .  STRUCTURE BASED(PACKETO) IN PROC (LINEWATCHER)
```

CROSS-REFERENCE LISTING

```
       0000H     4       KAOS . . . . .    POINTER
       0004H     2       LEN. . . . . .    WORD        140
       0006H     6       DLDEST . . . .    WORD ARRAY(3)
       000CH     6       DLSRC. . . . .    WORD ARRAY(3)
       0012H     2       DLTYPE . . . .    WORD
       0014H     1       BOOTCMD. . . .    BYTE        144
       0015H  1499       STUFF. . . . .    BYTE ARRAY(1499)
  134  0002H     2   PACKETO. . . . . .    WORD IN PROC (LINEWATCHER) PARAMETER     134  140  144  147
   24  0000H     2   PCBO . . . . . . .    WORD IN PROC (CQCREATEPROCESS) PARAMETER          24
   81  0000H     2   PO . . . . . . . .    WORD IN PROC (CQDLLRXRETBUF) PARAMETER           81
   84  0000H     1   PORTID . . . . . .    BYTE IN PROC (CQMIPCONNECT) PARAMETER            84
   24  0000H     2   PRI. . . . . . . .    WORD IN PROC (CQCREATEPROCESS) PARAMETER          24
   95  0000H     4   PTR. . . . . . . .    POINTER IN PROC (CQMIPGETMIPFORM) PARAMETER             95
  255  0050H   100   RCVBUF . . . . . .    BYTE ARRAY(100) IN PROC (REMOTEEXECUTOR)        266  272
  253              RCVBUFL. . . . . .    LITERALLY '100'  IN PROC (REMOTEEXECUTOR)       255  267
  128              READCTYPE. . . . .    LITERALLY '1'
  128              READMEMORYTYPE . .    LITERALLY '4'           155
  128              READTYPE . . . . .    LITERALLY '0'
  251  0338H   217   REMOTEEXECUTOR . .    PROCEDURE PUBLIC STACK=001EH
  202  0000H    37   REQ. . . . . . . .    STRUCTURE BASED(REQPTR) IN PROC (NML)           214  230  232
       0000H    12       MIP. . . . . .    BYTE ARRAY(12)
       000CH     2       PROCID . . . .    WORD
       000EH     1       CMD. . . . . .    BYTE        221
       000FH     1       RESULT . . . .    BYTE        220*  225*  238*
       0010H     2       RESPSOC. . . .    WORD        211  214  227  239
       0012H     4       RESPPTR. . . .    POINTER          224
       0016H     2       RESPMAX. . . .    WORD        226
       0018H     1       TYPE . . . . .    BYTE        226
       0019H     1       NUM. . . . . .    BYTE
       001AH     2       OBJECT . . . .    WORD
       001CH     2       MODIFIER . . .    WORD
       001EH     1       LEN. . . . . .    BYTE
       001FH     6       VALUE. . . . .    WORD ARRAY(3)
  150  0000H     6   REQ. . . . . . . .    STRUCTURE BASED(REQPTR) IN PROC (EXECUTE)
       0000H     2       OBJECT . . . .    WORD        177
       0002H     2       MODIFIER . . .    WORD        178
       0004H     1       LEN. . . . . .    BYTE        179
       0005H     1       VALUE. . . . .    BYTE ARRAY(1)           184
  150  000EH     2   REQF . . . . . . .    WORD IN PROC (EXECUTE) AT AUTOMATIC
  150  000CH     2   REQO . . . . . . .    WORD IN PROC (EXECUTE) AT AUTOMATIC      175*  175  185*  185
  201  0012H     4   REQPTR . . . . . .    POINTER IN PROC (NML)        211  214  219*  221  224  226  227
                                            230  232  239
  150  000CH     4   REQPTR . . . . . .    POINTER IN PROC (EXECUTE) PARAMETER AUTOMATIC        150  154  155
                                            159  163  164  166  174  177  178  179  184
  150  0000H     6   RESP . . . . . . .    STRUCTURE BASED(RESPPTR) IN PROC (EXECUTE)
       0000H     2       OBJECT . . . .    WORD        190*
       0002H     2       MODIFIER . . .    WORD        191*
       0004H     1       LEN. . . . . .    BYTE        189*
       0005H     1       VALUE. . . . .    BYTE ARRAY(1)           184  186
  150  000AH     2   RESPF. . . . . . .    WORD IN PROC (EXECUTE) AT AUTOMATIC
  150  0000H     2   RESPLEN. . . . . .    WORD BASED(RESPLENO) IN PROC (EXECUTE)          153*  160*  167*  197*
  150  0004H     2   RESPLENO . . . . .    WORD IN PROC (EXECUTE) PARAMETER AUTOMATIC      150
  150  0006H     2   RESPMAX. . . . . .    WORD IN PROC (EXECUTE) PARAMETER AUTOMATIC      150  157  158  180
  150  0000H     1   RESPNUM. . . . . .    BYTE BASED(SVDRESPPTR) IN PROC (EXECUTE)        172*  193*  193
  150  0008H     2   RESPO. . . . . . .    WORD IN PROC (EXECUTE) AT AUTOMATIC      173*  173  180  192*  192
                                            197
```

```
 203  0016H     4  RESPPTR. . . . . .  POINTER IN PROC (NML)          224  226
 150  0008H     4  RESPPTR. . . . . .  POINTER IN PROC (EXECUTE) PARAMETER AUTOMATIC
                                        134  186                                    150  159  171
 252  0048H     4  RESPPTR. . . . . .  POINTER IN PROC (REMOTEEXECUTOR)          263*  269*  276*
 150  000AH     2  RMNGBUFLEN . . . .  WORD IN PROC (EXECUTE)          180*  181
 151  0000H     6  RMREQ. . . . . .    STRUCTURE BASED(REQPTR) IN PROC (EXECUTE)
      0000H     1      TYPE . . . . .  BYTE
      0001H     4      POINTR . . . .  POINTER          159
      0005H     1      LEN. . . . . .  BYTE       154
  92  0000H     2  RO . . . . . .      WORD IN PROC (CQBOOTREGISTER) PARAMETER          92
  57  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQICWAIT) PARAMETER          57
  54  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQISIGNAL) PARAMETER          54
  36  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQCWAIT) PARAMETER          36
  33  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQWAITSEM) PARAMETER          33
  30  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQSIGNAL) PARAMETER          30
  27  0000H     2  SEMAPHOREO . . . .  WORD IN PROC (CQCREATESEMAPHORE) PARAMETER       27
 128             SETMEMORYTYPE. . .   LITERALLY '5'          162
 151  0000H     7  SETREQ . . . . . .  STRUCTURE BASED(REQPTR) IN PROC (EXECUTE)
      0000H     1      TYPE . . . . .  BYTE
      0001H     4      POINTR . . . .  POINTER          163
      0005H     1      LEN. . . . . .  BYTE       163
      0006H     1      VALUE. . . . .  BYTE ARRAY(1)          163
 128             SETTYPE. . . . . .   LITERALLY '2'          169
                 SETW . . . . . . .   BUILTIN          259
  88  0000H     2  SOCKET . . . . . .  WORD IN PROC (CQMIPSEND) PARAMETER          88
  24  0000H     2  STACKO . . . . . .  WORD IN PROC (CQCREATEPROCESS) PARAMETER          24
 126             STRBTYPE . . . . .   LITERALLY 'structure(mippart(7) word,req BYTE,resp BYTE,rtn$mip$skt WORD,li
                                        -nk POINTER,CID WORD,seq WORD,client$use WORD,buf$len WORD,num$blks BYTE,B
                                        -LK1$ptr pointer,blk1$len word)'          250
 150  0010H     2  SVDRESPF . . . . .  WORD IN PROC (EXECUTE) AT
 150  000EH     2  SVDRESPO . . . . .  WORD IN PROC (EXECUTE) AT          180  197
 150  000EH     4  SVDRESPPTR . . . .  POINTER IN PROC (EXECUTE)          150  171*  193
 204  02FCH    60  TAKEBACK . . . . .  PROCEDURE IN PROC (NML) STACK=000CH       234  236
 248             TCLABORT . . . . .   LITERALLY '8'
 244             TCLCLOSE . . . . .   LITERALLY '2'          273
 243             TCLOPENP . . . . .   LITERALLY '1'          260
 127             TCLPART. . . . . .   LITERALLY 'req BYTE,resp BYTE,rtn$mip$skt WORD,link POINTER,CID WORD,loc$po
                                        -rt WORD,rem$net WORD,rem$host(3)WORD,rem$port WORD,persist WORD,abort$tim
                                        -eout WORD,seq WORD'          249
 247             TCLPOSTRBUF. . . .   LITERALLY '7'          264
 250  001EH    37  TCLRB. . . . . .    STRUCTURE AT     268  275
      0000H    14      MIPPART. . . .  WORD ARRAY(7)
      000EH     1      REQ. . . . . .  BYTE       264*  273*
      000FH     1      RESP . . . . .  BYTE       270
      0010H     2      RTNMIPSKT. . .  WORD
      0012H     4      LINK . . . . .  POINTER
      0016H     2      CID. . . . . .  WORD
      0018H     2      SEQ. . . . . .  WORD
      001AH     2      CLIENTUSE. . .  WORD
      001CH     2      BUFLEN . . . .  WORD       270
      001EH     1      NUMBLKS. . . .  BYTE       265*  271*
      001FH     4      BLK1PTR. . . .  POINTER          266*  274*
      0023H     2      BLK1LEN. . . .  WORD       267*  272
 113  0000H         TCLREAD. . . . . .  PROCEDURE EXTERNAL(33) STACK=0000H          152
 116  0000H         TCLREADC . . . . .  PROCEDURE EXTERNAL(34) STACK=0000H          152
 245             TCLSEND. . . . . .   LITERALLY '5'
```

```
246                 TCLSENDEOM . . . .    LITERALLY '6'
119   0000H         TCLSET . . . . . .    PROCEDURE EXTERNAL(35) STACK=0000H      152
128                 TCLSOCKET. . . . .    LITERALLY '0004H'      262  268  275
 11                 TRUE . . . . . . .    LITERALLY 'OFFH'       218  258
150   0000H     2   TYPENNUM . . . . .    STRUCTURE BASED(REQPTR) IN PROC (EXECUTE)
      0000H     1       TYPE . . . . .    BYTE       155
      0001H     1       NUM. . . . . .    BYTE       174
  7                 UNTIL. . . . . . .    LITERALLY 'IF NOT('
131   0000H     2   V. . . . . . . . .    WORD IN PROC (VECTOR) PARAMETER         131
  8                 VALID. . . . . . .    LITERALLY ') THEN GOTO LOOPLAB; EXITLAB: END'
130   0000H         VECTOR . . . . . .    PROCEDURE BYTE EXTERNAL(42) STACK=0000H      186
 51   0000H     2   WCBO . . . . . . .    WORD IN PROC (CQMRECEIVE) PARAMETER       51
256   00B4H   200   XMITBUF. . . . . .    BYTE ARRAY(200) IN PROC (REMOTEEXECUTOR)      272  274
254                 XMITBUFL . . . . .    LITERALLY '200'  IN PROC (REMOTEEXECUTOR)     256  272
```

MODULE INFORMATION:

```
CODE AREA SIZE     = 0411H   1041D
CONSTANT AREA SIZE = 000CH     12D
VARIABLE AREA SIZE = 017CH    380D
MAXIMUM STACK SIZE = 001EH     30D
646 LINES READ
1 PROGRAM WARNING
0 PROGRAM ERRORS
```

END OF PL/M-86 COMPILATION