```
/**************************************************************************

                         SPOOL
                         IP.LST
                         05/04/82
                         15:47:53


**************************************************************************/
```

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE IP
OBJECT MODULE PLACED IN :F1:IP.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:IP.P86 OPTIMIZE(3) XREF SET(F1) DEBUG



              $TITLE('iLNA Transport Control Layer Interface Process  04/15/82')
              $COMPACT DEBUG NOCOND
*** WARNING 10 IN 1 (LINE 2): RESPECIFIED PRIMARY CONTROL, IGNORED
              $SET(mipform)

              $IF f7
              $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



                    /*  George D Marshall  SC6-213    x7-5117  */

   1          ip: DO;

                /* The Interface Process intercepts all client requests, handles those
                it can (such as Status), and does any pre-processing possible on others
                (such as Open) before passing them along to the proper subsystem in TCL */

              $IF f7
              $ELSE
              $INCLUDE (:F1:TCLGBL.INC)
         =                                                  /*********************/
         =                                                  /** Global Literals **/
         =                                                  /*********************/
         =    /* TCL Global Literals                                       04/15/82 */
   2   1 =    DECLARE
         =        max$send$seg         LITERALLY   '07H',  /* max no of back-to-back segs that  one connection */
         =                                                 /* can send at a time */
         =        tcl$header$len       LITERALLY   '20',   /* bytes in tcl header */

         =                                         /* ETHERNET-SPECIFIC VALUES */
         =        dll$header$len       LITERALLY   '14',   /* bytes in dll header */
         =        min$pkt$len          LITERALLY   '46',   /* minimum total pkt len - bytes */
         =        max$seg$data$len$lit LITERALLY   '1480', /* (1480) max no. of client bytes in seg */
         =        tcl$protocol$code    LITERALLY   '5001H',/* DLLCONNECT user type field */
         =        tcl$protocol$code$rev LITERALLY '0150H',/* packet header user type field */

         =                                         /* Misc values */
         =        tcl$mip$port         LITERALLY   '4',    /* mip port for IP$IN$MBX */
         =        log$rb$mip$port      LITERALLY   '5',    /* debugging: mip port for logging */
         =        mip$echo$port        LITERALLY   '7',    /* mip port of on-bd tcl echo server */
```

```
          =       tcl$version$lit      LITERALLY   '101H', /* Version of this TCL for seg header */
          =       def$net$id$lit       LITERALLY   '1',    /* default Network ID: "this network" */
          =       on$bd$tcl$echo$port LITERALLY    '7',    /* TCL port of on-board tcl echo server */
          =       true                 LITERALLY   '0FFH',
          =       false                LITERALLY   '0',
          =       forever              LITERALLY   'WHILE true',

          =       Timeout$increase$state  LITERALLY '1',  /* In this state the retransmission timeout
          =                                                   is rapidly increased */
          =       Timeout$steady$state    LITERALLY '0';  /* In this state the timeout is
          =                                           slowly decreased.  This should not be
          =                                           changed, it is the initial state since
          =                                           a cdb is intialised to zero */
              $ENDIF
                                                                  /************************/
                                                                  /* TCL Global Data Base */
                                                                  /************************/

                                                      /* Externals in TSTART */
   3   1      DECLARE
                  num$cdbs          BYTE EXTERNAL,   /* no. of cdb's open now */
                                                          /* NOTE limit of 255 cdbs */
                  lcid$vector(*)    WORD EXTERNAL,   /* list of cids in use */
                  spec$type(*)      BYTE EXTERNAL,   /* remote socket spec type: full, partial, un- */
                  cur$max$cdbs      BYTE EXTERNAL,   /* max no of conns */

                  tcl$state         BYTE EXTERNAL,   /* state of tcl:??*/
                  loc$net           WORD EXTERNAL,   /* my network ID */
                  loc$host(3)       WORD EXTERNAL,   /* my host ID (from DLL) */
                  def$retran$to$dw DWORD EXTERNAL,/* of timeout */
                  def$abort$to$hi WORD EXTERNAL,
                  def$persist       WORD EXTERNAL,
                  max$seg$data$len WORD EXTERNAL, /* max no. client bytes in seg */
                  max$window$size BYTE EXTERNAL,
                  rtc$dw            DWORD EXTERNAL; /* timestamp storage for stat */


                                                      /* IP's local variables */
   4   1      DECLARE
                  cur$cdb$index     BYTE,             /* indicates current lcidvector word */
                  cur$cid           WORD,             /* cid of connection being processed */
                  first$open        BYTE    INITIAL(true),  /* "first open request" flag" */
                  last$cid          WORD PUBLIC,      /* last local cid value used, initialized
                                                      at first open. */
                  spec$type$temp    BYTE,             /* type of open specification-temp */
                  rbs_req           BYTE,
                  tot$buf$len       WORD,             /* used in enter queue routines */
                  (j,scratch)       WORD;             /* scratch variables */

              $IF dbg
              $ENDIF

              $IF log
              $ENDIF
                                                          /* Client interface routines (code-saver
                                                          approach) */
   5   1      DECLARE
```

```
              proc$location   WORD,
              procs (*)       WORD    INITIAL
                  (.open$rtn, .open$rtn, .close$rtn, .status$rtn, .def$status$rtn,
                   .send$rtn, .send$rtn, .post$rbuf, .abort$rtn);

          $IF f7
          $ELSE
          $SAVE NOLIST INCLUDE (:F1:TCLMBX.INC)

          $IF f7
          $ELSE
          $SAVE NOLIST INCLUDE (:F1:DLL.DCP)

          $IF f7
          $ELSE
          $SAVE NOLIST INCLUDE (:F1:THACF.INC)

          $IF log
          $ENDIF

 26   1       DECLARE
            cur$cdb$p          POINTER,
            c BASED cur$cdb$p
          $IF f7
          $ELSE
          $INCLUDE (:F1:TCLCDB.INC)
      =                                                         /*********************/
      =                                                         /** Conn Data Base ***/
      =                                                         /*********************/
      =     STRUCTURE ( /* Connection Data Base Template              10/16/81 */
      =                                 /* Offset */
      =     state               BYTE,   /*  0       upper bit indicates passive/act open */
      =     owner$device        BYTE,   /*  1       MIP device ID of local client */
      =     owner$process$id      WORD,/* 2       ID of local client for failure handler-no relationship to SCL ID's
              - */
      =     loc$cid             WORD,   /*  4 */
      =                                             /* Following fields (***) are aligned with
      =                                             same fields in RBO-exploited by openrtn */
      =     loc$port            WORD,   /*  6       ***     */
      =     rem$net             WORD,   /*  8       ***     */
      =     rem$host (3)        WORD,   /*  A = 10t ***     */
      =     rem$port            WORD,   /* 10 = 16T ***     */
      =     persist             WORD,   /* 12 = 18t ***     no of times to ignore connect refusal */
      =     abort$to$hi         WORD,   /* 14 = 20t ***     */
      =     rem$cid             WORD,   /* 16 = 22T */
      =     retran$to$dw        DWORD,  /* 18 = 24T re-transmit timeout value */
      =     reserved            WORD,   /* 1C = 28T */
      =     timed$seq$no        WORD,   /* 1E = 30t xmit seq no whose response is being timed */
      =     seg$trans$time$dw DWORD,/* 20 = 32t transmit timestamp for timed segment */
      =     cum$retran$dw       DWORD,  /* 24 = 36t cummulative retran time */
      =     persist$cnt         WORD,   /* 28 = 40t count of number of times my SYN refused */
      =     cbtq$hdr            POINTER,/* 2A = 42t   client buffer transmit queue hdr */
      =     pcbq$hdr            POINTER,/* 2E = 46t   posted client buffer queue hdr */
      =     def$status$p        POINTER,/* 32 = 50t   ptr to deferred status RB, if any  */

      =     my$ack$no           WORD,   /* 36 = 54T highest rcvd seg we can ack */
      =     seen                WORD,   /* 38 = 56T highest seg seq no seen by rcvr */
```

```
     =       my$credit        BYTE,   /* 3A = 58t # pkts we said remote tcl can send us*/
     =       curblk$index     BYTE,   /* 3B = 59t blk # of current rcv blk */
     =       cbdata$index     WORD,   /* 3C = 60t next avail byte in current rcv block */
     =       rcv$bytes$consumed WORD,/* 3E = 62T # bytes already saved from current numbered rcv seg */
     =       curblk$len$left WORD,    /* 40 = 64t # bytes left in current rcv blk */
     =       his$ack$no       WORD,   /* 42 = 66t highest xmit seg no acked by remote guy */
     =       next$transmit    WORD,   /* 44 = 68t seq no to be sent next (not always highest) */
     =       closed$reason    BYTE,   /* 46 = 70t Saved rbs.resp when conn aborted */
     =       his$credit       BYTE,   /* 47 = 71T # pkts remote tcl said we can send */
     =       highest$sent     WORD,   /* 48 = 72t  err/flow hidden variable for GET */

     =       cbtq$buf$cnt     BYTE,   /* 4A = 74t */
     =       pcbq$buf$cnt     BYTE,   /* 4B = 75t */

     =       pkts$rej         WORD,   /* 4C = 76t  # of rcvd pkts discarded */
     =       pkts$retran      WORD,   /* 4E = 78t  # of pkts re-transmitted */
     =       no$confid        WORD,   /* 50 = 80t  ballpark retry count since last ack*/
     =       last$no$conf$id WORD,    /* 52 = 82t  # of retries last time there was a retransmission */
     =       Retransmit$state byte,   /* 54 = 84t  State of retransmission algorithm */
     =       sendflag         BYTE,   /* 55 = 85t  */

     =       pending$rcv$data WORD,   /* 56 = 86t  no. of undelivered bytes in last pkt */
     =       rcv$buf$rej$cnt WORD,    /* 58 = 88t  no. of times pkt rejected due to no rcv buf */
     =       ayt$count        WORD,   /* 5A = 90t  no. of times ayt pkts weren't ansewered */
     =                                                /* The Alarm control Blocks */
     =       data$alarm$cb(2) WORD,   /* 5C = 92t  Data Alarm Control Block header */
     =       data$acb$irbtype BYTE,   /* 60 = 96t  Type byte positioned same as in irb */
     =       data$acb$flag    BYTE,   /* 61 = 97t  ACB Flag byte (running, expired, clrd) */
     =       data$acb$rem(10) BYTE,   /* 62 = 98t  remainder of data acb */

     =       ctl$alarm$cb(2) WORD,    /* 6C =108t  Control Alarm Control Block header */
     =       ctl$acb$irbtype BYTE,    /* 70 =112t  Type byte positioned same as in irb */
     =       ctl$acb$flag     BYTE,   /* 71 =113t  ACB Flag byte (running, expired, clrd) */
     =       ctl$acb$rem(10) BYTE)    /* 72 =114t  rest of ctl acb */
     =                                /* 7C =124t  Total Length */
         $ENDIF
             ;

         $IF f7
         $ELSE
         $SAVE NOLIST INCLUDE (:F1:TCLCSD.INC)

                                                                   /*********************/
                                                                   /***   RBO   Format **/
                                                                   /*********************/
  28   1      DECLARE
                 cur$rbs$p          POINTER,
                 cur$rbv$p          POINTER,
                 nxt$rb$p           POINTER,
                 rbo BASED cur$rbs$p
              $IF f7
              $ELSE
              $SAVE NOLIST INCLUDE (:F1:TCLRBO.INC)
                       ;

                                                                   /*********************/
                                                                   /***   RBS   Format **/
                                                                   /*********************/
  29   1       DECLARE
                   rbs BASED cur$rbs$p
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                   ,
                   rbv BASED cur$rbv$p (1)
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLRBV.INC)
                   ;
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLRBC.INC)

                                                                      /**********************/
                                                                      /**** IRB  Format  ***/
                                                                      /**********************/
               /*    TCL's Internal Request Block Template (for TCL's processes to
                     communicate with each other under CMX */

  31   1       DECLARE
                   (irb$o, irb$b)  WORD,
                   irb$p        POINTER AT(@irb$o),
                   irb BASED irb$o
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLIRB.INC)
                   ;
  32   1       DECLARE                           /* Long IRBs; used to tell TP to send RSTs */
                   lirb BASED irb$o
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TLIRB.INC)
                   ;

  33   1       DECLARE
                   def$st$p     POINTER,
                   def$st       BASED def$st$p
               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                   ;

               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLIRC.INC)

               $IF f7
               $ELSE
               $SAVE NOLIST INCLUDE (:F1:TCLRSC.INC)

               /* External Procedure declarations:  Define calls to other layers and CMX */

               $IF log
               $ENDIF
  36    1       setup$cdb: PROCEDURE(index, cdb$p$o) EXTERNAL;                     /* in TSTART */
  37    2           DECLARE   index BYTE,    cdb$p$o WORD;
  38    2           END setup$cdb;

  39    1       min: PROCEDURE(n,m) WORD EXTERNAL;                                  /* in RP */
  40    2           DECLARE      (n, m)  WORD;
  41    2           END min;

  42    1       clear$cdb$alarms: PROCEDURE(cdb$p) EXTERNAL;                        /* in RP */
  43    2           DECLARE      cdb$p    POINTER;
  44    2           END clear$cdb$alarms;

  45    1       chk$deferred$status: PROCEDURE(cdb$p) EXTERNAL;                     /* in RP */
  46    2           DECLARE cdb$p    POINTER;
  47    2           END chk$deferred$status;

  48    1       random: PROCEDURE WORD EXTERNAL;                                    /* in random */
  49    2           END random;

  50    1       search_lcid$vector: PROCEDURE(find$target) WORD EXTERNAL;           /* in TCOM */
  51    2           DECLARE      find$target WORD;
  52    2           END search_lcid$vector;

                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:KAOS.DCP)
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:MIP.DCP)



 135    1       DECLARE
                    ip$irb$list (3) BYTE    INITIAL(0FFH,0FFH,0FFH),
                    ip$irb$index    BYTE    INITIAL(0);

                                                                     /*************************/
                                                                     /***  ip$defer$irb$tp  ***/
                                                                     /*************************/

 136    1       defer$irb$tp: PROCEDURE(type,cdb$p,irb$index$o, irb$list$o) PUBLIC REENTRANT;
                                                           /* This routine accepts requests to send */
                                                           /* an irb to the transmit process (tp). */
                                                           /* The send is deferred until after ip */
                                                           /* does not have schedclock locked, to */
                                                           /* avoid a deadlock with tp. */

 137    2       DECLARE type BYTE,                         /* irb type to send */
                    cdb$p   POINTER,
                    irb$index$o WORD,
                    irb$list$o  WORD,
                    irb$index   BASED   irb$index$o BYTE,
                    irb$list    BASED   irb$list$o (3) BYTE,
                    dip$c   BASED   cdb$p
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:TCLCDB.INC)
                ;

138   2       IF (type = irb$send$check) OR (type = irb$send$flag) THEN
139   2           IF dip$c.send$flag THEN RETURN;     /* a previous sendflag subsumes either one.*/
141   2       IF type = irb$send$flag THEN dip$c.send$flag = true;    /* set it if not prev set */
143   2       irb$list(irb$index) = type; /* save the type code until later */
144   2       irb$index = irb$index + 1;
145   2       IF irb$index > 2 THEN CALL cq$halt$and$catch$fire(hacf$ip$def$irb);
147   2       END defer$irb$tp;


148   1       ip$defer$irb$tp: PROCEDURE(type);
                                                 /* code-saver routine for common call */
149   2       DECLARE type    BYTE;

150   2       CALL defer$irb$tp(type, cur$cdb$p, .ip$irb$index, .ip$irb$list);
151   2       END ip$defer$irb$tp;


152   1       send$deferred$irbs: PROCEDURE(irb$index$o, irb$list$o, cdb$index, cid) PUBLIC REENTRANT;
                                              /* common version for rp and ip for */
                                              /* companion routine to above, called */
                                              /* ONLY AFTER sched$lock has been released */
                                              /* by ip or rp, to actually send the irbs */
                                              /* requested by this invocation of ip or rp */
                                              /* while schedlock was held.  Avoids */
                                              /* the deadlock caused by tp doing a */
                                              /* schedule while holding an irb */
153   2       DECLARE j   BYTE,
                  irb$index$o     WORD,
                  irb$list$o      WORD,
                  cdb$index       BYTE,
                  cid             WORD,
                  irb$index   BASED irb$index$o    BYTE,
                  irb$list    BASED irb$list$o  (3) BYTE;

154   2       IF irb$index = 0 THEN RETURN;         /* nothing to do */
156   2       DO j = 0 TO irb$index-1;
157   3           irb$p = cq$receive(.free$irb$mbx);
158   3           irb.type = irb$list(j);
159   3           irb.cdb$index = cdb$index;
160   3           irb.cid = cid;
161   3           CALL cq$send(.tp$mbx, irb$p);
162   3       END;
163   2       irb$index = 0;
164   2       END send$deferred$irbs;


                                                                /*************************/
                                                                /**** ip_send_rbs_back ***/
                                                                /*************************/

165   1       ip_send_rbs_back: PROCEDURE(resp_code);

                                         /* shared-code routine to insert a response code
                                         in the current rbs, and send it back to client. */
```

```
 166   2        DECLARE
                    resp_code   BYTE;

 167   2        rbs.resp = resp_code;
 168   2        rbs.link = 0;                       /* always clear link field */
 169   2        CALL cq$send(.buf$mip$mbx, cur$rbs$p);
 170   2        END ip_send_rbs_back;


                                                                         /*****************/
                                                                         /*** delta$seq ***/
                                                                         /*****************/
 171   1        delta$seq: PROCEDURE(buffer$length) WORD;
                                              /* this routine is used to compute the delta to add
                                              to the first sequence number of an RB to get the last
                                              sequence number associated with that RB.   Used in
                                              enter$cbtq to assign sequence numbers, and in
                                              enter$pcbq to assign rcv buffer credit. */
 172   2        DECLARE
                    (buffer$length, delta)  WORD;
 173   2        delta = buffer$length / max$seg$data$len;
 174   2        IF (delta <> 0) AND ((buffer$length MOD max$seg$data$len) = 0 ) THEN
 175   2            delta = delta - 1;
 176   2        RETURN(delta);

 177   2        END delta$seq;
                                                                     /*********************/
                                                                     /*** last$rbs$ptr  ***/
                                                                     /*********************/
 178   1        last$rbs$ptr: PROCEDURE(lrp$p) POINTER;
                                              /* code-saver routine to find the last RBS on a TCL
                                              linked list, and return its pointer. */
 179   2        DECLARE
                    lrp$p   POINTER,
                    lrp$rbs BASED lrp$p
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                    ;

 180   2            DO WHILE lrp$rbs.link <> 0;
 181   3                lrp$p = lrp$rbs.link;       /* go next entry down the list */
 182   3            END;

 183   2        RETURN(lrp$p);
 184   2        END last$rbs$ptr;
                                                                     /*********************/
                                                                     /*** sum$of$blocks ***/
                                                                     /*********************/
 185   1        sum$of$blocks: PROCEDURE WORD;
 186   2        DECLARE sum WORD;
                                                      /* code-saver routine to sum the total bytes
                                                      in an xmit or rcv buf in the current RBS. */
 187   2        sum = 0;
 188   2        DO j = 1 TO rbs.num$blks;           /* store it in the buf len field of the RBS */
 189   3            sum = sum + rbv(j-1).blk$len;
 190   3        END;
 191    2       RETURN(sum);
 192    2       END sum$of$blocks;

                                                                      /*********************/
                                                                      /****  enter$cbtq ****/
                                                                      /*********************/

 193    1       enter$cbtq: PROCEDURE;

                               /*  Routine to enter a send request block onto the client buffer
                               transmit queue (cbtq)  The request block to be entered is the
                               current IP RBS, as specified in cur$rbs$p.  The sequence number
                               assigned to the first segment in the buffer supplied is inserted
                               in the RB seq field for use by the Transmit and Receive processes.  */

                               /* This routine supports the client buffer fragmentation mechanism.
                               Request Blocks may describe arbitrarily long (up to 65540 bytes)
                               buffers, presented as a set of arbitrarily sized non-contiguous
                               blocks.  Zero-length blocks are allowed, as are zero-length buffers,
                               although a zero length buffer will not be returned until the first
                               non-zero successor buffer has been sent and acknowledged
                               (exception: a zero length send when there is nothing else on the
                               xmit queue will be returned immediately). */

 194    2       DECLARE cbtq$rbs$p       POINTER,
                        last$seq         WORD,
                        j                BYTE,
                        cbtq$rbs BASED cbtq$rbs$p
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                        ;

 195    2       c.cbtq$buf$cnt = c.cbtq$buf$cnt + 1;     /* Bump buffer count in queue */
 196    2       IF c.cbtq$buf$cnt = 1 THEN
 197    2           DO;                                /* Queue is empty- RB becomes top entry */
 198    3               c.cbtq$hdr = cur$rbs$p;
                                                       /* record seq no of last seg to use in */
 199    3               IF (c.state AND state$mask) < estab THEN last$seq = 1;  /* generating seq's */
 201    3                   ELSE lastseq = c.next$transmit -1;
                                                   /* seq for this RB */
 202    3           END;

 203    2       ELSE                                   /* queue non-empty: put RB at end of queue */
                    DO;
 204    3               cbtq$rbs$p = last$rbs$ptr(c.cbtq$hdr);  /* Get ptr to last RBS on list */
 205    3               last$seq = cbtq$rbs.last$seq;
 206    3               cbtq$rbs.link = cur$rbs$p;  /* now link the new RB to the last RB, and ... */
 207    3           END;
                                                       /* Now compute the total number of bytes */
 208    2       tot$buf$len = sum$of$blocks;           /* in all blocks of this buffer */
 209    2       rbs.buf$len = tot$buf$len;
 210    2       rbs.link = 0;                          /* mark RB as being last one */

                                                       /* now set flag indicating whether this RB has */
                                                       /* any sequence-number-consuming contents -- i.e.,*/
                                                       /* client data or sequence-consuming tcl control */
```

```
                                             /* signals */
        rbs.contents = false;
        IF (tot$buf$len <> 0) OR (rbs_req = send$eom$req) OR (rbs_req = close$req)
            THEN rbs.contents = true;
                                             /* now assign first and last sequence numbers to this RB */
        IF rbs.contents THEN
            DO;                              /* it has sendable contents */
                rbs.first$seq = (last$seq + 1);
                rbs.last$seq = (last$seq + 1) + delta$seq(tot$buf$len);
            END;
        ELSE
            DO;                          /* zilch contents - give it null seq no */
                rbs.first$seq, rbs.last$seq = last$seq;
                IF c.cbtq$buf$cnt = 1 THEN
                    DO;                  /* RB is only one in xmit queue-send it back now */
                        CALL ip_send_rbs_back( ok$resp);
                        c.cbtq$buf$cnt = 0;
                        c.cbtq$hdr = 0;
                    END;
            END;

        END enter$cbtq;

                                                                  /*********************/
                                                                  /****  enter$pcbq ****/
                                                                  /*********************/

        enter$pcbq: PROCEDURE;

                   /*  Routine to enter a request block onto the posted client buffer
                    receive queue (pcbq).  The request block to be entered is the
                    current IP RBS, as specified in cur$rbs$p.
                       If the RB being entered goes at the top of the queue, then two
                    variables are initialized for the PUT (re-assembly) routine in
                    the Receive Process: the number of bytes available in the top
                    available block, and the currently in-use block number in the
                    top RB. */

        DECLARE pcbq$rbs$p POINTER,
                pcbq$rbs BASED pcbq$rbs$p
        $IF f7
        $ELSE
        $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                    ;

        c.pcbq$buf$cnt = c.pcbq$buf$cnt + 1;    /* incr no. of buffers in this queue */
        IF c.pcbq$buf$cnt = 1 THEN
            DO;
                c.pcbq$hdr = cur$rbs$p;         /* set queue header and base for RB */
                IF rbs.num$blks <> 0
                    THEN c.curblk$len$left = rbv(0).blk$len;    /* Set up so PUT (in RP) will work */
                    ELSE c.curblk$len$left = 0; /* no blocks */
                c.cbdata$index = 0;             /* set up for PUT */
                c.curblk$index = 0;
            END;
        ELSE
            DO;
            pcbq$rbs$p = last$rbs$ptr(c.pcbq$hdr);  /* get ptr to last rbs on queue */
            pcbq$rbs.link = cur$rbs$p;      /* end of list found - link new rbs to it  */
        END;
    rbs.buf$len = 0;                        /* Initialize "bytes-used" in RB for PUT */
    rbs.link = 0;                           /* mark end of queue */

                                            /* Now compute the total number of bytes */
    tot$buf$len = sum$of$blocks;            /* in all blocks of this buffer. */
                                            /* assign credit equal to estimated */
                                            /* number of segs that fit in buf, */
                                            /* and save it in the RB. */
    c.my$credit = c.my$credit + ( rbs.credit := 1 + delta$seq(tot$buf$len) );
    END enter$pcbq;

                                                            /*********************/
                                                            /**** clear$lists ****/
                                                            /*********************/

    clear$lists: PROCEDURE(clcdb$p, clrtncode) BYTE PUBLIC;
                            /* Routine to send the contents, if any, of the */
                            /* cbtq and pcbq of the current connection back */
                            /* to the client.  This routine is also called  */
                            /* by the Transmit Process when it handles Abort */
                            /* Returns "true" if any RBs were found so the */
                            /* caller knows whether it is ok to delete the cdb */
    DECLARE
        clrtncode   BYTE,
        rb$found    BYTE,       /* boolean indicating that >= 1 RBs were returned */
        clcdb$p     POINTER,
        clc BASED clcdb$p
    $IF f7
    $ELSE
    $SAVE NOLIST INCLUDE (:F1:TCLCDB.INC)
        ;

                            /* routine to clear a list */
        clear$one$list: PROCEDURE(list$p);
        DECLARE
            (list$p, link$ptr)  POINTER,
            cl$rbs BASED list$p
    $IF f7
    $ELSE
    $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
            ;
        DO WHILE list$p <> 0;
            link$ptr = cl$rbs.link; /* save link field so its still good after send */
                                    /* link field will be cleared by tbms */
            cl$rbs.resp = clrtncode;
            cl$rbs.link = 0;        /* always clear link field */
            CALL cq$send(.buf$mip$mbx, list$p);
            rb$found = true;        /* set flag: we found an rb to return */
            list$p = link$ptr;
        END;
        END clear$one$list;

                            /* main clear$lists code */
    rb$found = false;
```

```
264   2       CALL clear$one$list(clc.pcbq$hdr);          /* Clear the Receive Queue */
265   2       CALL clear$one$list(clc.def$status$p);      /* clear the deferred status q */
266   2       CALL clear$one$list(clc.cbtq$hdr);          /* Clear the Transmit Queue last so */
                                                          /* that the Close RB will be last */

267   2       clc.cbtq$hdr,clc.pcbq$hdr, clc.def$status$p = 0;    /* mark queues empty */
268   2       clc.pcbq$buf$cnt, clc.cbtq$buf$cnt = 0;
269   2       RETURN(rb$found);

270   2       END clear$lists;


                                                                     /*********************/
                                                                     /****  delete$cdb  ***/
                                                                     /*********************/
271   1       delete$cdb: PROCEDURE(dlcdb$index, dlcdb$p, dlrtn$code) PUBLIC;
                                        /* This routine unconditionally deletes a
                                        connection data base with all housekeeping,
                                        including killing the alarms built into it,
                                        and sending back any Request Blocks to the
                                        local client. Contrast with try$to$delete$cdb
                                        in RP, which calls this one. */
272   2       DECLARE
                  dlcdb$index BYTE,
                  dlrtn$code  BYTE,
                  dlcdb$p     POINTER;

273   2       scratch = clear$lists(dlcdb$p, dlrtn$code);
274   2       CALL clear$cdb$alarms(dlcdb$p);                        /* Kill alarms just in case */
275   2       lcid$vector(dlcdb$index) = 0;       /* delete the cdb by removing the cid */
276   2       num$cdbs = num$cdbs - 1;
277   2       spec$type(dlcdb$index) = 0FFH;      /* remove this cdb from open match list */

278   2       END delete$cdb;

                                                                     /*********************/
                                                                     /****   open$rtn  ****/
                                                                     /*********************/

279   1       open$rtn: PROCEDURE;                /* Open active and Open passive start here */

                                                  /* Check to see if it's legal */
280   2       IF rbo.loc$port = 0 THEN            /* illegal: can't have local port=0 */
281   2           DO;
282   3               CALL ip_send_rbs_back(illegal$req);
283   3               RETURN;
284   3           END;
                                                  /* determine type of match specification
                                                  and do additional legality checks */
285   2       IF (rbo.rem$host(0) = 0) AND (rbo.rem$host(1) = 0) AND (rbo.rem$host(2) = 0) THEN
286   2           DO;                             /* rem host unspecified */
287   3               IF rbo.req = opena$req THEN /* err: active open must be fully spec'd */
288   3                   DO;
289   4                       CALL ip_send_rbs_back(illegal$req);
290   4                       RETURN;
291   4                   END;
292   3               ELSE                        /* passive open: ok */
```

```
                           DO;
 293   4                      IF rbo.rem$port = 0 THEN spec$type$temp = 2;     /* unspecified */
 295   4                                               ELSE spec$type$temp = 1;     /* partially spec'd */
 296   4                   END;
 297   3             END;
 298   2          ELSE                                 /* remote host is specified */
                     DO;
 299   3                IF rbo.rem$port = 0 THEN     /* illegal whether active or passive: */
 300   3                   DO;                       /* can't have spec'd rem host and unspec'd */
 301   4                      CALL ip_send_rbs_back(illegal$req); /* remote port */
 302   4                      RETURN;
 303   4                   END;
 304   3                ELSE spec$type$temp = 0;     /* fully specified */
 305   3             END;
                                  /* Now check to see that there is no conflict with other
                                  pending opens: TCL can't cope with more than one active
                                  open with the same local port and remote socket, nor can
                                  it handle open actives and fully specified open passives
                                  with the same local port and remote socket. The reason
                                  is that two TCL modules with  multiple opens toward each
                                  other could simultaneously make conficting decisions
                                  about the binding of an incoming SYN request to a local
                                  CDB.  Note that it is ok to have any number of simultaneous
                                  connections between the same two sockets, but the client
                                  has to wait for each to become established before
                                  requesting the next one.  Partially specified passive opens
                                  which match an active open are ok, because the connection-
                                  matching logic in the Receive Process will always match
                                  a more strictly-specified CBD before a lessor-specified one
                                  on SYN processing. */

                                  /* Scan all CDB's, looking for another unsynchronized connection
                                  on the same local port, and with a matching remote socket: */
 306   2          DO j = 0 TO cur$max$cdbs-1;
 307   3             IF spec$type(j) = 0 THEN
 308   3                DO; /* found an unsynch'd CDB */
 309   4                   CALL setup$cdb(j, .cur$cdb$p);
 310   4                   IF c.loc$port = rbo.loc$port AND
                              (CMPW(@c.rem$host(0), @rbo.rem$host(0), 4) = OFFFFH) THEN
                              /*c.rem$host(0) = rbo.rem$host(0) AND*/
                              /*c.rem$host(1) = rbo.rem$host(1) AND*/
                              /*c.rem$host(2) = rbo.rem$host(2) AND*/
                              /*c.rem$port = rbo.rem$port */

 311   4                      DO; /* found a match - see if it matters ... if req is open active,
                                    then no match is ok, but if req is open passive, then
                                    other open passives are ok, but not open actives. */
 312   5                         IF rbs_req = opena$req OR
                                    (rbs_req = openp$req AND c.state = synsent) THEN
 313   5                            DO;
 314   6                               CALL ip_send_rbs_back(open$conflict);
 315   6                               RETURN;
 316   6                            END;
 317   5                      END;
 318   4                END;
 319   3          END;    /* of loop */
                                               /* we now have a legal open request - see
```

```
                                                      if there is space for it */
 320   2      IF num$cdbs >= cur$max$cdbs THEN
 321   2          DO;                                 /* Error: no resources */
 322   3              CALL ip_send_rbs_back(no$resources$resp);
 323   3              RETURN;
 324   3          END;
                                                      /* There's space: allocate a new Connection ID (local half) */
 325   2      set$cid:
              IF first$open THEN
 326   2          DO;
                                                      /* get timestamp for random value */
                                                      /* Halfway to random: this code produces
                                                      an initial CID whose lower twelve bits are
                                                      random, but whose upper 4 bits are the lower
                                                      4 bits of the local host ID.  Drop out the
                                                      first OR clause to get totally random value.
                                                      This scheme depends for its randomness on
                                                      the variability of the disc accesses for
                                                      the comm and OS boot, and (for workstations))
                                                      on the time until LOGON is typed. */

 327   3              last$cid = (SHL(loc$host(2),4) AND 0F000H) OR (random AND 00FFFH);
 328   3              first$open = false;
 329   3          END;

 330   2      newcid:
 331   2      IF (last$cid:=last$cid+1) = 0 THEN last$cid = last$cid + 1;
                                                      /* check to see that its free, repeat if
                                                      we get an "occupied" one */

 332   2      IF search_lcid$vector(last$cid) <> 0FFFFH THEN GOTO newcid;
                                                      /* there's space, and we have a new cid,
                                                      so find a hole for it in the index table */
 334   2      cur$cdb$index = search_lcid$vector(0);
                                                      /* Got the hole, now install the new cid */
 335   2      lcid$vector(cur$cdb$index), rbo.cid, cur$cid = last$cid;
 336   2      CALL setup$cdb(cur$cdb$index, .cur$cdb$p);  /* set up ptr for c structure */
 337   2      num$cdbs = num$cdbs + 1;

                                                      /* Now zero out the parts of the new cdb
                                                      area that we don't write immediately */

                                                      /* NOTE: need to check this on any changes
                                                      in the CDB format !!!!!!!! */

                                   /* 12/13/81 - R. Shah - changed from 48 to 52,as noconfid
                                   field is changed to word, and two new fields (last$noconid - word
                                   and retransmit$state - byte are added.  The following set
                                   now covers from the 22nd byte to the 121st byte. */

 338   2      if(c.data$acb$flag=0) and (c.data$acb$irbtype=84h) then
 339   2         call CQclear$alarm(@c.data$alarm$cb);     /* kludge !!!!!!!!!! */

 340   2      CALL SETW(0, @c.rem$cid, 50);
                                                      /* Now partially fill in cdb from info in
                                                      the request block  */
 341   2      spec$type(cur$cdb$index) = spec$type$temp;  /* set it for Receive process */
```

```
               /*
                       c.loc$cid = cur$cid;
                       c.loc$port = rbo.loc$port;
                       c.rem$net = rbo.rem$net;
                       c.rem$host(0) = rbo.rem$host(0);
                       c.rem$host(1) = rbo.rem$host(1);
                       c.rem$host(2) = rbo.rem$host(2);
                       c.rem$port = rbo.rem$port;
               */
 342    2      CALL MOVW(@rbo.cid, @c.loc$cid, 9); /* copy socket, cid, persist, abort$to info */
                                                           /* now check for defaults */
 343    2      IF c.persist = 0
 344    2          THEN c.persist = def$persist;
 345    2      IF c.abort$to$hi = 0
 346    2          THEN c.abort$to$hi = def$abort$to$hi;

 347    2      c.retran$to$dw = def$retran$to$dw;
 348    2      c.next$transmit = 1;
 349    2      c.owner$device = rbo.mip$owner$dev$id;
 350    2      c.owner$process$id = rbo.owner$process$id;  /* not related to SCL process ID */
 351    2      CALL cq$create$alarm(@c.data$alarm$cb);     /* Create cdb alarms so we can */
 352    2      CALL cq$create$alarm(@c.ctl$alarm$cb);      /* set and clear them later */

                                                     /* Now check for active vs passive open req */
 353    2      IF rbo.req = openp$req THEN
 354    2          DO;                                  /* PASSIVE */
 355    3              c.state = listen;
 356    3          END;
 357    2      ELSE
                   DO;                                  /* ACTIVE */
 358    3              c.state = synsent;
                                                     /* Now tell transmit process to send a SYN
                                                     control seg */
 359    3              CALL ip$defer$irb$tp(irb$send$syn);    /* note irb send is deferred */
 360    3          END;
               $IF log
               $ENDIF
 361    2      CALL ip_send_rbs_back(ok$resp);

 362    2      END open$rtn;


                                                                         /*********************/
                                                                         /****  close$rtn  ****/
                                                                         /*********************/
 363    1      close$rtn: PROCEDURE;

 364    2      close$enqueue: PROCEDURE(new$state);                 /* subroutine for use */
                                                                     /* by the close routine */
 365    3      DECLARE
                   new$state   BYTE;

 366    3          c.state = new$state;             /* Change conn state to indicated value */
               $IF log
               $ENDIF
 367    3          CALL enter$cbtq;                 /* now, put the RB on the xmit queue */
 368   3            CALL ip$defer$irb$tp(irb$send$check);   /* and tell Transmit proc */
 369   3        END close$enqueue;

                                                 /* Note: CID has already been checked */
 370   2        DO CASE (c.state AND state$mask);

                /* case listen */
 371   3            GOTO synsent$case;               /* Listen: same as synsent processing */

                /* case synsent */
 372   3        synsent$case:
                    DO;                              /* Close request issued before connection */
                                                     /* reaches a synchronized state, so just */
                                                     /* delete it */
 373   4                CALL delete$cdb(cur$cdb$index, cur$cdb$p, ok$closed$resp);
                $IF log
                $ENDIF
 374   4                CALL ip_send_rbs_back( ok$resp);
 375   4            END;

                /* case syn received */
 376   3            CALL close$enqueue(finwait$1);  /* same as established state processing */

                /* case established */
 377   3            CALL close$enqueue(finwait$1);

                /* case finwait1 */
 378   3            CALL ip_send_rbs_back(illegal$req);     /* illegal to issue a 2nd Close */

                /* case finwait2 */
 379   3            CALL ip_send_rbs_back(illegal$req);

                /* case timewait */
 380   3            CALL ip_send_rbs_back(illegal$req);

                /* case close wait */
 381   3            CALL close$enqueue(closing);

                /* case closing */
 382   3            CALL ip_send_rbs_back(illegal$req);

                /* case closed */
 383   3            CALL ip_send_rbs_back(illegal$req);

 384   3        END;    /* of do case */

 385   2        END close$rtn;

                                                                  /*********************/
                                                                  /** get$status$info **/
                                                                  /*********************/

 386   1        get$status$info: PROCEDURE(st$cdb$p, st$rbs$p) BYTE PUBLIC;
                                                 /* this routine actually copies the status */
                                                 /* info into the client's buffer */
                                                 /* (This routine is public, and requires cdbp */
                                                 /* and rbsp as parameters because it is also */
```

```
                                                        /* called by chkdeferred stat routine in RP */
 387   2      DECLARE
                 st$cdb$p          POINTER,             /* CDB ptr for status reference */
                 st$rbs$p          POINTER,             /* ptr to rbs of the status request */
                 st$rbv$p          POINTER,             /* variable part of above */
                 st$rbs BASED st$rbs$p                  /* the request block */
              $IF f7
              $ELSE
              $SAVE NOLIST INCLUDE (:F1:TCLRBS.INC)
                 ,
                 st$rbv BASED st$rbv$p (1)
              $IF f7
              $ELSE
              $SAVE NOLIST INCLUDE (:F1:TCLRBV.INC)
                 ,
                 stb$p             POINTER,             /* Status Buffer Pointer */
                 stb               BASED stb$p          /* Status Buffer */
              $IF f7
              $ELSE
              $INCLUDE (:F1:TCLSTA.INC)
          =   /* Definition of the fields returned in a Status request buffer    11/03/81 */

          =      STRUCTURE (      /* fields returned on either type of request */
          =      tcl$state    BYTE,   /* state of tcl */
          =      def$abort    WORD,   /* default abort timeout for connections */
          =      def$retran$dw   DWORD,  /* default retransmit timeout */
          =      def$persist WORD,    /* defualt prsistence value */
          =      cur$max$cdbs BYTE,   /* max number of cdbs for which there is space avail */
          =      numcdbs      BYTE,   /* number of connection data bases now allocated*/
          =      loc$net      WORD,   /* ID of our own network */
          =      loc$host(3) WORD,    /* ID of this comm board (the local host ID) */
          =      tot$pkts$rej WORD,   /* total no. of rcv packets rejected by this TCL */
          =      tot$retran$events WORD, /* total number of times retran timer expired */
          =      tot$rcv$buf$rej WORD,    /* ttl no times there was insufficient buf space */
          =      rtc$dw       DWORD,  /* real-time-clock: (read-clock units) */

          =                              /* fields returned only for cid <> 0 */
          =                           /* NOTE: locport thru remport must be in this order to agree */
          =                           /* with tclcdb.inc declaration - doing MOVW in status$req */
          =      loc$port     WORD,   /* the local port number of this connection */
          =      rem$net      WORD,   /* ID of remote network */
          =      rem$host(3) WORD,    /* ID of the remote host */
          =      rem$port     WORD,   /* the remote port number for this connection */
          =      loc$cid      WORD,   /* local half of the connection id */
          =      rem$cid      WORD,   /* remote half of the connection ID */
          =      conn$abort   WORD,   /* abort timeout value for this connection */
          =      conn$retran$to$dw DWORD,     /* conn retransmit timeout value */
          =      conn$persist    WORD,    /* persistence value for this connection */
          =      connstate    BYTE,   /* state of the connection */
          =      pending$rcv$data WORD,   /* no. of bytes of rcv data w/no rcv buf space */
          =      rcv$buf$rej$cnt WORD,/* number of times there was insuff rcv buf space */
          =      cbtq$buf$cnt BYTE,   /* Number of RBs in transmit queue */
          =      pcbq$buf$cnt BYTE,   /* Number of RBs in Receive buffer queue */
          =      loc$credit   BYTE,   /* my$credit: #pkts we said remote guy could send*/
          =      rem$credit   BYTE,   /* his$credit:#pkts remote guy said we can send */
          =      highest$sent WORD,   /* highest seg seq number sent (mod 65k)*/
          =      my$ack$no    WORD,   /* highest seg seq number we have acked */
          =       no$confid   WORD,    /* approximate number of retries since last ack */
          =                          , /* was received; value is inversely proportional*/
          =                            /* to confidence that remote tcl and/or client */
          =                            /* is still alive and healthy  */
          =       last$entry  BYTE)    /* symbolic ref to end of list: no info here */
                $ENDIF
                    ,
                    st$c BASED st$cdb$p                /* connection data base for status req */
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:TCLCDB.INC)
                    ;

 388   2        status$buf$too$short: PROCEDURE(buf$size) BYTE;
                                                       /* routine to test that buffer size supplied
                                                       by client is sufficient for the data requested */
 389   3            DECLARE
                        buf$size    WORD;

 390   3            RETURN ( (st$rbv(0).blk$len < buf$size) OR (st$rbs.num$blks = 0) );
 391   3        END status$buf$too$short;


 392   2        st$rbv$p = @st$rbs.vb;                 /* set up addressing to buffer */
                $IF mipform
 393   2        stb$p = cq$mip$get$address( st$rbv(0).blk$ptr);     /* set up ptr for status structure */
                $ELSE
                $ENDIF
 394   2        IF status$buf$too$short (.stb.locport - .stb) THEN RETURN(buf$too$short);

                                                       /* fill in fields for short req */

 396   2        CALL cq$read$clock(.rtc$dw);           /* get timestamp to send to client */
                                                       /* move everything in one chunk since
                                                       all connection-independent items have
                                                       been aligned as declared in TSTART */
 397   2        CALL MOVB(@tcl$state, @stb.tcl$state, .stb.locport - .stb);

                                                       /* Now check to see if it's a connection- */
                                                       /* specific request, and fill in more */
                                                       /* fields if there is space. */

 398   2        IF st$rbs.cid = 0 THEN RETURN(ok$resp);
 400   2        IF status$buf$too$short(.stb.last$entry - .stb) THEN RETURN(buf$too$short);

                                                       /* NOTE: fields in cdb and status buf do not */
                                                       /* currently match up properly, so change them to */
                                                       /* force a match, since mucho bytes can be saved */
                                                       /* by doing a movw like the one below! */

                                                       /* copy loc port, remnet, rem host,remport */
 402   2        CALL MOVW(@st$c.loc$port, @stb.loc$port, 6);
 403   2        stb.loc$cid = st$c.loc$cid;
 404   2        stb.rem$cid = st$c.rem$cid;
 405   2        stb.conn$abort = st$c.abort$to$hi;
 406   2        stb.conn$retran$to$dw = st$c.retran$to$dw;
 407   2        stb.conn$persist = st$c.persist;
 408   2       stb.connstate = st$c.state;
 409   2       stb.pending$rcv$data = st$c.pending$rcv$data;
 410   2       stb.rcv$buf$rej$cnt = st$c.rcv$buf$rej$cnt;
 411   2       stb.cbtq$buf$cnt = st$c.cbtq$buf$cnt;
 412   2       stb.pcbq$buf$cnt = st$c.pcbq$buf$cnt;
 413   2       stb.loc$credit = st$c.my$credit;
 414   2       stb.rem$credit = st$c.his$credit;
 415   2       stb.highest$sent = st$c.highest$sent;
 416   2       stb.my$ack$no = st$c.my$ack$no;
 417   2       stb.noconfid = st$c.noconfid;
 418   2       RETURN(ok$resp);                    /* now send it back */
 419   2       END get$status$info;

                                                                                   /*********************/
                                                                                   /**** status$rtn  ****/
                                                                                   /*********************/
 420   1       status$rtn: PROCEDURE;
                            /* the Status function returns information about TCL and/or
                            the individual connection requested.  The client must supply
                            sufficient buffer space for the type of information requested.
                            Status always returns info on TCL if there is space available for
                            it.  If connection info was requested, and there is sufficient
                            space for tcl data but not conn data, then only tcl data is
                            returned. */

                                    /* main line status code */

 421   2       CALL ip_send_rbs_back( get$status$info(cur$cdb$p, cur$rbs$p) );

 422   2       END status$rtn;


                                                                                   /*********************/
                                                                                   /**  def$status$rtn **/
                                                                                   /*********************/
 423   1       def$status$rtn: PROCEDURE;
                                                   /* Handles deferred status requests.  If
                                                   state is in Established or beyond, send
                                                   the RB back immediately marked OK.  If
                                                   state is listen, sysnsent, or synreceived,
                                                   then hold the RB until state becomes
                                                   established, then send it back. If multiple
                                                   def stat's are received, they are stored
                                                   as a stack. */

                                                   /* Always put the RB on the def stat list... */
 424   2       rbs.link = c.def$status$p;          /* link top def st rb (if any) to new RB */
 425   2       c.def$status$p = cur$rbs$p;         /* insert new rb as top one on list */

                                                   /* Now use the receive Process' routine to
                                                   take it off if state is estab or beyond */
 426   2       IF (c.state AND state$mask) >= estab THEN CALL chk$deferred$status(cur$cdb$p);
 428   2       END def$status$rtn;


                                                                                   /*********************/
                                                                                   /****   send$rtn  ****/
```

```
                                                                        /*********************/

 429   1       send$rtn: PROCEDURE;

 430   2       DO CASE (c.state AND state$mask);
               /*  listen */
 431   3           CALL enter$cbtq;

               /*  synsent */
 432   3           CALL enter$cbtq;

               /*  synrcvd */
 433   3           CALL enter$cbtq;

               /*  estab */
 434   3           GOTO case$clswait;

               /*  finwait1 */
 435   3           GOTO case$closed;

               /*  finwait2 */
 436   3           GOTO case$closed;

               /*  timewait */
 437   3           GOTO case$closed;
               /*  close wait */
 438   3           case$clswait:
                       DO;
 439   4                   CALL enter$cbtq;              /* Put RBS on the queue */
 440   4                   CALL ip$defer$irb$tp(irb$sendcheck);
 441   4               END;
               /*  closing */
 442   3           GOTO case$closed;
               /*  closed (when cdb still exists)*/
 443   3           case$closed:
                       DO;
 444   4                   CALL ip_send_rbs_back( illegal$req);        /* can't do send after close or abort */
 445   4               END;
 446   3       END; /* case */

 447   2       END send$rtn;


                                                                        /*********************/
                                                                        /****   post$rbuf ****/
                                                                        /*********************/
 448   1       post$rbuf: PROCEDURE;
                                                   /* Routine to enter a client's Receive buffer
                                                   on the posted client buffer queue... put it
                                                   on unless a FIN has previously arrived, in
                                                   which case nothing else can be received on
                                                   this connection, so give it back immediately
                                                   with an ok$fin response. */
 449   2       IF (c.state AND state$mask) >= timewait THEN    /* state is timewait, closewait,
                                                   closing, or closed, so a FIN arrived earlier,
                                                   possibly while no rcv bufs were posted, so
```

```
                                                 tell him. Note that we will tell him again */
                                                 /* everytime he issues a post rcv buf. */
 450   2           CALL ip_send_rbs_back(ok$fin$resp);
 451   2        ELSE CALL enter$pcbq;

 452   2        END post$rbuf;

                                                                  /*********************/
                                                                  /****  abort$conn  ***/
                                                                  /*********************/
 453   1        abort$conn: PROCEDURE(ac$cdb$index, ac$cdb$p) PUBLIC;
                                        /* Sends an RST segment to remote
                                        client, if the conn is in a synchronized state,
                                        and isn't already sufficiently advanced into a
                                        closing sequence.  Always deletes the CDB before
                                        returning client's RB, so a new open can re-use
                                        the cdb.  This is public because it is also called
                                        by the mip delete process routine. */

 454   2        DECLARE
                    ac$cdb$index     BYTE,
                    ac$cdb$p         POINTER,
                    ac$c     BASED ac$cdb$p
                $IF f7
                $ELSE
                $SAVE NOLIST INCLUDE (:F1:TCLCDB.INC)
                    ;

 455   2           build$send$lirb: PROCEDURE;

                                           /* Routine used by several states to send the RST: */
                                           /* build and send an lirb to TP asking it to
                                           send an RST seg (this is the only use of lirb's).
                                           The fields are set up to indicate the contents
                                           of the actual header field.  The order of the
                                           dest port thru seg ack no fields is the same
                                           as the actual seg header, so that a MOVW call
                                           can be used by TP. */

 456   3              CALL cq$signal(.sched$lock);        /* relinquish lock while we get buf */
 457   3              irb$p = cq$receive(.free$lirb$mbx);
 458   3              CALL cq$waitsem(.sched$lock);       /* now re-acquire the lock */
 459   3              lirb.reason = rst$client$abort;
 460   3              lirb.type = irb$sendrst;
 461   3              lirb.dl$dest(0) = ac$c.rem$host(0);
 462   3              lirb.dl$dest(1) = ac$c.rem$host(1);
 463   3              lirb.dl$dest(2) = ac$c.rem$host(2);
 464   3              lirb.dest$port = ac$c.rem$port;
 465   3              lirb.source$port = ac$c.loc$port;
 466   3              lirb.dest$cid = ac$c.rem$cid;
 467   3              lirb.source$cid = ac$c.loc$cid;
 468   3              lirb.seg$seq$no = ac$c.highest$sent + 1;
 469   3              lirb.seg$ack$no = ac$c.my$ack$no;
 470   3              CALL cq$send(.tp$mbx, irb$p);
 471   3           END build$send$lirb;

                                        /* main abort conn code */
 472   2        DO CASE(ac$c.state AND state$mask);

                /* listen */
 473   3            ;                            /* no rst seg sent in listen */
                /* syn sent */
 474   3            ;
                /* syn rcvd */
 475   3            CALL build$send$lirb;
                /* estab */
 476   3            CALL build$send$lirb;
                /* fin wait 1 */
 477   3            CALL build$send$lirb;
                /* fin wait 2 */
 478   3            CALL build$send$lirb;
                /* time wait */
 479   3            ;
                /* close wait */
 480   3            CALL build$send$lirb;
                /* closing */
 481   3            CALL build$send$lirb;
                /* closed */
 482   3            ;

 483   3        END;    /* of do case */

                                                    /* now that we have scheduled an RST seg
                                                    for those states that need it, delete
                                                    the cdb */

 484   2        CALL delete$cdb(ac$cdb$index,ac$cdb$p,loc$abort);   /* rtn calls clearlists, too */

 485   2        END abort$conn;


                                                                    /*********************/
                                                                    /****  abort$rtn  ****/
                                                                    /*********************/

 486   1        abort$rtn: PROCEDURE;
                                        /* Routine to honor client's request to abort
                                        a connection.  */
 487   2        CALL abort$conn(cur$cdb$index, cur$cdb$p);  /* do RST send and cdb deletion */

                $IF log
                $ENDIF
 488   2        CALL ip_send_rbs_back( ok$resp);
 489   2        END abort$rtn;


                $IF dbg
                $ENDIF
```

```
/*          The main Session Control Interface PROCESS Definition          */

                                                  /*********************/
                                                  /****   ip$proc   ****/
                                                  /*********************/

ip$proc: PROCEDURE PUBLIC;
                             /* This is the client request Interface
                             Process, one of four processes in TCL.
                             It is declared in TCLGEN.A86 (or EPGEN.A86,
                             in the Echo Server version) */

DO forever;                  /* Processes loop forever */
                             /* wait for a request block */
ip$top:
    cur$rbs$p = cq$receive(.ip$in$mbx);
processrb:
    CALL cq$waitsem(.sched$lock);   /* get control of cdb before any serious changes */

$IF dbg
$ENDIF
                             /* before doing any processing, save the link
                             field from this RB so that it can be used
                             for request block chaining regardless of the */
    nxt$rb$p = rbs.link;     /* disposition of the current RB */

    cur$rbv$p = @rbs.vb;     /* set up variable part of RB, in case needed */
$IF log
$ENDIF

                             /* check for valid request code */
    rbs_req = rbs.req;

    IF rbs_req > req$max THEN       /* possible bad req type: check it */
$IF dbg
$ENDIF
             DO;                    /* its a bad rb req code */
                rbs.resp = invalid$req; /* set RB response field */
                CALL cq$send(.buf$mip$mbx, cur$rbs$p);  /* send it back to client. */
                             /* NOTE: this is the only place in TCL
                             where we don't zero the RB link field
                             first; the reason is that there is
                             a somewhat higher chance that this type of
                             error was caused by a chunk of random
                             memory being sent to TCL, and we don't
                             want to propagate the effect by
                             interpreting the <possible> junk in the
                             link field as another RB pointer.
                             We leave the link field untouched to help
                             the client by not losing the linked RBs
                             (if any), in case this really isn't
                             random memory */
                CALL cq$signal(.sched$lock);
```

```
502   4                     GOTO ip$top;
503   4                   END;
                                /* Validate cid: cid shouldn't be checked on open or */
                                /* a status req with cid=0, but otherwise cid */
                                /* should be non-zero, and be in the localcid list */

504   3           IF ( rbs_req > openp$req )  AND
                     ( (rbs_req <> status$req) OR (rbs.cid <> 0) ) THEN
505   3               DO;                           /* CID must be validated */
506   4                   IF ( (rbs.cid = 0) OR
                             (cur$cdb$index:=search_lcid$vector(rbs.cid)) = 0FFFFH) THEN
507   4                       DO;                   /* bad cid, tell client */
508   5                           CALL ip_send_rbs_back(unknown$cid$resp);
509   5                           GOTO ip$post$proc;
510   5                       END;
                                /* cid ok, set up access to cdb */
511   4                   cur$cid = rbs.cid;
512   4                   call setup$cdb(cur$cdb$index, .cur$cdb$p);
                                /* Check for Closed state: if true, then connection was
                                previously aborted by remote node or local timeout when
                                there was no RB queued with which to notify local
                                client. We can tell him now with this RB, using the
                                resp code saved for the purpose, thus allowing the CDB
                                to be deleted. */
513   4                   IF c.state = closed THEN        /* processing is limited to sending it back */
514   4                       DO;
                  $IF log
                  $ENDIF
515   5                           CALL ip_send_rbs_back(c.closed$reason);
516   5                           CALL delete$cdb(cur$cdb$index, cur$cdb$p, c.closed$reason);
517   5                           GOTO ip$post$proc;      /* continue with link processing */
518   5                       END;
519   4               END;

                                        /* do the actual processing of the RB */
                                        /* Now demultiplex by request type */
520   3           proc$location = procs(rbs_req); /* replaces DO CASE on rbs_req */
521   3           CALL proc$location;

522   3       ip$post$proc:
              $IF dbg
              $ENDIF
                  CALL cq$signal(.sched$lock);     /* give back the cdb lock */

                                          /* now that we have released schedlock, */
523   3           CALL send$deferred$irbs
                          (.ip$irb$index, .ip$irb$list, cur$cdb$index, cur$cid);
                                          /* send any deferred irbs to tp that */
                                          /* were requested this time thru. */

                                    /* Let in any higher priority processes (such as */
                                    /* the Transmit Process), since the */
524   3           CALL cq$schedule;     /* RB just processed might have made it ready */

                                    /* Check to see if link field of RB just finished*/
                                    /* had an entry; if so, go back up and process as*/
                                    /* though it just came in tp$mbx. */
```

```
 525   3              IF nxt$rb$p <> 0  THEN
 526   3                  DO;
                $IF mipform
 527   4                      cur$rbs$p = cq$mip$get$address( nxt$rb$p);
                $ELSE
                $ENDIF
 528   4                      GOTO processrb;
 529   4                  END;

 530   3          END; /* end of IP process do forever loop */
 531   2          END ip$proc;

 532   1          END ip;
```

CROSS-REFERENCE LISTING

```
 DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES
----- ------ -----  -------------------------------


  453  0951H    79  ABORTCONN. . . . . . . .    PROCEDURE PUBLIC STACK=0022H        487
   30               ABORTREQ . . . . . . . .    LITERALLY '8'
  486  0A1FH    24  ABORTRTN . . . . . . . .    PROCEDURE STACK=0026H          5
  454  0000H   124  ACC. . . . . . . . . . .    STRUCTURE BASED(ACCDBP) IN PROC (ABORTCONN)
       0000H     1      STATE. . . . . . . .    BYTE       472
       0001H     1      OWNERDEVICE. . . . .    BYTE
       0002H     2      OWNERPROCESSID . . .    WORD
       0004H     2      LOCCID . . . . . . .    WORD       467
       0006H     2      LOCPORT. . . . . . .    WORD       465
       0008H     2      REMNET . . . . . . .    WORD
       000AH     6      REMHOST. . . . . . .    WORD ARRAY(3)         461  462  463
       0010H     2      REMPORT. . . . . . .    WORD       464
       0012H     2      PERSIST. . . . . . .    WORD
       0014H     2      ABORTTOHI. . . . . .    WORD
       0016H     2      REMCID . . . . . . .    WORD       466
       0018H     4      RETRANTODW . . . . .    DWORD
       001CH     2      RESERVED . . . . . .    WORD
       001EH     2      TIMEDSEQNO . . . . .    WORD
       0020H     4      SEGTRANSTIMEDW . . .    DWORD
       0024H     4      CUMRETRANDW. . . . .    DWORD
       0028H     2      PERSISTCNT . . . . .    WORD
       002AH     4      CBTQHDR. . . . . . .    POINTER
       002EH     4      PCBQHDR. . . . . . .    POINTER
       0032H     4      DEFSTATUSP . . . . .    POINTER
       0036H     2      MYACKNO. . . . . . .    WORD       469
       0038H     2      SEEN . . . . . . . .    WORD
       003AH     1      MYCREDIT . . . . . .    BYTE
       003BH     1      CURBLKINDEX. . . . .    BYTE
       003CH     2      CBDATAINDEX. . . . .    WORD
       003EH     2      RCVBYTESCONSUMED . .    WORD
       0040H     2      CURBLKLENLEFT. . . .    WORD
       0042H     2      HISACKNO . . . . . .    WORD
       0044H     2      NEXTTRANSMIT . . . .    WORD
       0046H     1      CLOSEDREASON . . . .    BYTE
       0047H     1      HISCREDIT. . . . . .    BYTE
       0048H     2      HIGHESTSENT. . . . .    WORD       468
       004AH     1      CBTQBUFCNT . . . . .    BYTE
       004BH     1      PCBQBUFCNT . . . . .    BYTE
       004CH     2      PKTSREJ. . . . . . .    WORD
       004EH     2      PKTSRETRAN . . . . .    WORD
       0050H     2      NOCONFID . . . . . .    WORD
       0052H     2      LASTNOCONFID . . . .    WORD
       0054H     1      RETRANSMITSTATE. . .    BYTE
       0055H     1      SENDFLAG . . . . . .    BYTE
       0056H     2      PENDINGRCVDATA . . .    WORD
       0058H     2      RCVBUFREJCNT . . . .    WORD
       005AH     2      AYTCOUNT . . . . . .    WORD
       005CH     4      DATAALARMCB. . . . .    WORD ARRAY(2)
       0060H     1      DATAACBIRBTYPE . . .    BYTE
       0061H     1      DATAACBFLAG. . . . .    BYTE
       0062H    10      DATAACBREM . . . . .    BYTE ARRAY(10)
       006CH     4      CTLALARMCB . . . . .    WORD ARRAY(2)
```

```
      0070H     1       CTLACBIRBTYPE. . .    BYTE
      0071H     1       CTLACBFLAG . . . .    BYTE
      0072H    10       CTLACBREM. . . . .    BYTE ARRAY(10)
 454  0062H     1   ACCDBINDEX . . . . . .    BYTE IN PROC (ABORTCONN) PARAMETER        454  484
 454  005EH     4   ACCDBP . . . . . . . .    POINTER IN PROC (ABORTCONN) PARAMETER        454  461  462  463
                                               464  465  466  467  468  469  472  484
 118  0000H     4   ALARMP . . . . . . . .    POINTER IN PROC (CQCLEARALARM) PARAMETER     118
 112  0000H     4   ALARMP . . . . . . . .    POINTER IN PROC (CQSETALARM) PARAMETER       112
 109  0000H     4   ALARMP . . . . . . . .    POINTER IN PROC (CQCREATEALARM) PARAMETER    109
 115  0000H     4   ALARMP . . . . . . . .    POINTER IN PROC (CQCHECKALARM) PARAMETER     115
 172  0004H     2   BUFFERLENGTH . . . . .    WORD IN PROC (DELTASEQ) PARAMETER AUTOMATIC        172  173  174
   6  0000H     2   BUFMIPMBX. . . . . . .    WORD EXTERNAL(18)      169  258  500
  12  0000H     2   BUFO . . . . . . . . .    WORD IN PROC (CQDLLRXRETBUF) PARAMETER        12
 389  0004H     2   BUFSIZE. . . . . . . .    WORD IN PROC (STATUSBUFTOOSHORT) PARAMETER AUTOMATIC       389  390
  30                BUFTOOSHORT. . . . . .    LITERALLY '8'          395  401
 455  09A0H   127   BUILDSENDLIRB. . . . .    PROCEDURE IN PROC (ABORTCONN) STACK=000AH    475  476  477  478
                                               480  481
  26  0000H   124   C. . . . . . . . . . .    STRUCTURE BASED(CURCDBP)
      0000H     1       STATE. . . . . . .    BYTE        199  312  355*  358*  366*  370  426  430  449  513

      0001H     1       OWNERDEVICE. . . .    BYTE        349*
      0002H     2       OWNERPROCESSID . .    WORD        350*
      0004H     2       LOCCID . . . . . .    WORD        342
      0006H     2       LOCPORT. . . . . .    WORD        310
      0008H     2       REMNET . . . . . .    WORD
      000AH     6       REMHOST. . . . . .    WORD ARRAY(3)          310
      0010H     2       REMPORT. . . . . .    WORD
      0012H     2       PERSIST. . . . . .    WORD        343  344*
      0014H     2       ABORTTOHI. . . . .    WORD        345  346*
      0016H     2       REMCID . . . . . .    WORD        340
      0018H     4       RETRANTODW . . . .    DWORD       347*
      001CH     2       RESERVED . . . . .    WORD
      001EH     2       TIMEDSEQNO . . . .    WORD
      0020H     4       SEGTRANSTIMEDW . .    DWORD
      0024H     4       CUMRETRANDW. . . .    DWORD
      0028H     2       PERSISTCNT . . . .    WORD
      002AH     4       CBTQHDR. . . . . .    POINTER           198*  204  225*
      002EH     4       PCBQHDR. . . . . .    POINTER           234*  242
      0032H     4       DEFSTATUSP . . . .    POINTER           424  425*
      0036H     2       MYACKNO. . . . . .    WORD
      0038H     2       SEEN . . . . . . .    WORD
      003AH     1       MYCREDIT . . . . .    BYTE        248*  248
      003BH     1       CURBLKINDEX. . . .    BYTE        239*
      003CH     2       CBDATAINDEX. . . .    WORD        238*
      003EH     2       RCVBYTESCONSUMED .    WORD
      0040H     2       CURBLKLENLEFT. . .    WORD        236*  237*
      0042H     2       HISACKNO . . . . .    WORD
      0044H     2       NEXTTRANSMIT . . .    WORD        201  348*
      0046H     1       CLOSEDREASON . . .    BYTE        515  516
      0047H     1       HISCREDIT. . . . .    BYTE
      0048H     2       HIGHESTSENT. . . .    WORD
      004AH     1       CBTQBUFCNT . . . .    BYTE        195*  195  196  221  224*
      004BH     1       PCBQBUFCNT . . . .    BYTE        231*  231  232
      004CH     2       PKTSREJ. . . . . .    WORD
      004EH     2       PKTSRETRAN . . . .    WORD
      0050H     2       NOCONFID . . . . .    WORD
```

```
       0052H      2       LASTNOCONFID . . .    WORD
       0054H      1       RETRANSMITSTATE. .    BYTE
       0055H      1       SENDFLAG . . . . .    BYTE
       0056H      2       PENDINGRCVDATA . .    WORD
       0058H      2       RCVBUFREJCNT . . .    WORD
       005AH      2       AYTCOUNT . . . . .    WORD
       005CH      4       DATAALARMCB. . . .    WORD ARRAY(2)          339  351
       0060H      1       DATAACBIRBTYPE . .    BYTE        338
       0061H      1       DATAACBFLAG. . . .    BYTE        338
       0062H     10       DATAACBREM . . . .    BYTE ARRAY(10)
       006CH      4       CTLALARMCB . . . .    WORD ARRAY(2)          352
       0070H      1       CTLACBIRBTYPE. . .    BYTE
       0071H      1       CTLACBFLAG . . . .    BYTE
       0072H     10       CTLACBREM. . . . .    BYTE ARRAY(10)
  443  092CH          CASECLOSED . . . . . .    LABEL IN PROC (SENDRTN)      435  436  437  442
  438  0921H          CASECLSWAIT. . . . . .    LABEL IN PROC (SENDRTN)      434
  194  0000H     32   CBTQRBS. . . . . . . .    STRUCTURE BASED(CBTQRBSP) IN PROC (ENTERCBTQ)
       0000H      1       CONTENTS . . . . .    BYTE
       0001H      1       CREDIT . . . . . .    BYTE
       0002H      2       LASTSEQ. . . . . .    WORD        205
       0004H      4       MIPBUFBASE . . . .    POINTER
       0008H      2       MIPLENGTH. . . . .    WORD
       000AH      1       MIPIDSID . . . . .    BYTE
       000BH      1       MIPOWNERDEVID. . .    BYTE
       000CH      2       INTERNALPROCESSID.    WORD
       000EH      1       REQ. . . . . . . .    BYTE
       000FH      1       RESP . . . . . . .    BYTE
       0010H      2       RTNMIPSKT. . . . .    WORD
       0012H      4       LINK . . . . . . .    POINTER          206*
       0016H      2       CID. . . . . . . .    WORD
       0018H      2       FIRSTSEQ . . . . .    WORD
       001AH      2       CLIENTUSE. . . . .    WORD
       001CH      2       BUFLEN . . . . . .    WORD
       001EH      1       NUMBLKS. . . . . .    BYTE
       001FH      1       VB . . . . . . . .    BYTE
  194  003AH      4   CBTQRBSP . . . . . . .    POINTER IN PROC (ENTERCBTQ)         204*  205
  153  0006H      1   CDBINDEX . . . . . . .    BYTE IN PROC (SENDDEFERREDIRBS) PARAMETER AUTOMATIC         153  159
  137  0008H      4   CDBP . . . . . . . . .    POINTER IN PROC (DEFERIRBTP) PARAMETER AUTOMATIC     137  139
   46  0000H      4   CDBP . . . . . . . . .    POINTER IN PROC (CHKDEFERREDSTATUS) PARAMETER         46
   43  0000H      4   CDBP . . . . . . . . .    POINTER IN PROC (CLEARCDBALARMS) PARAMETER      43
   37  0000H      2   CDBPO. . . . . . . . .    WORD IN PROC (SETUPCDB) PARAMETER        37
   45  0000H          CHKDEFERREDSTATUS. . .    PROCEDURE EXTERNAL(30) STACK=0000H       427
  153  0004H      2   CID. . . . . . . . . .    WORD IN PROC (SENDDEFERREDIRBS) PARAMETER AUTOMATIC         153  160
  251  0000H    124   CLC. . . . . . . . . .    STRUCTURE BASED(CLCDBP) IN PROC (CLEARLISTS)
       0000H      1       STATE. . . . . . .    BYTE
       0001H      1       OWNERDEVICE. . . .    BYTE
       0002H      2       OWNERPROCESSID . .    WORD
       0004H      2       LOCCID . . . . . .    WORD
       0006H      2       LOCPORT. . . . . .    WORD
       0008H      2       REMNET . . . . . .    WORD
       000AH      6       REMHOST. . . . . .    WORD ARRAY(3)
       0010H      2       REMPORT. . . . . .    WORD
       0012H      2       PERSIST. . . . . .    WORD
       0014H      2       ABORTTOHI. . . . .    WORD
       0016H      2       REMCID . . . . . .    WORD
       0018H      4       RETRANTODW . . . .    DWORD
```

```
        001CH      2        RESERVED . . . . .    WORD
        001EH      2        TIMEDSEQNO . . . .    WORD
        0020H      4        SEGTRANSTIMEDW . .    DWORD
        0024H      4        CUMRETRANDW. . . .    DWORD
        0028H      2        PERSISTCNT . . . .    WORD
        002AH      4        CBTQHDR. . . . . .    POINTER          266  267*
        002EH      4        PCBQHDR. . . . . .    POINTER          264  267*
        0032H      4        DEFSTATUSP . . . .    POINTER          265  267*
        0036H      2        MYACKNO. . . . . .    WORD
        0038H      2        SEEN . . . . . . .    WORD
        003AH      1        MYCREDIT . . . . .    BYTE
        003BH      1        CURBLKINDEX. . . .    BYTE
        003CH      2        CBDATAINDEX. . . .    WORD
        003EH      2        RCVBYTESCONSUMED .    WORD
        0040H      2        CURBLKLENLEFT. . .    WORD
        0042H      2        HISACKNO . . . . .    WORD
        0044H      2        NEXTTRANSMIT . . .    WORD
        0046H      1        CLOSEDREASON . . .    BYTE
        0047H      1        HISCREDIT. . . . .    BYTE
        0048H      2        HIGHESTSENT. . . .    WORD
        004AH      1        CBTQBUFCNT . . . .    BYTE       268*
        004BH      1        PCBQBUFCNT . . . .    BYTE       268*
        004CH      2        PKTSREJ. . . . . .    WORD
        004EH      2        PKTSRETRAN . . . .    WORD
        0050H      2        NOCONFID . . . . .    WORD
        0052H      2        LASTNOCONFID . . .    WORD
        0054H      1        RETRANSMITSTATE. .    BYTE
        0055H      1        SENDFLAG . . . . .    BYTE
        0056H      2        PENDINGRCVDATA . .    WORD
        0058H      2        RCVBUFREJCNT . . .    WORD
        005AH      2        AYTCOUNT . . . . .    WORD
        005CH      4        DATAALARMCB. . . .    WORD ARRAY(2)
        0060H      1        DATAACBIRBTYPE . .    BYTE
        0061H      1        DATAACBFLAG. . . .    BYTE
        0062H     10        DATAACBREM . . . .    BYTE ARRAY(10)
        006CH      4        CTLALARMCB . . . .    WORD ARRAY(2)
        0070H      1        CTLACBIRBTYPE. . .    BYTE
        0071H      1        CTLACBFLAG . . . .    BYTE
        0072H     10        CTLACBREM. . . . .    BYTE ARRAY(10)
  251   0046H      4    CLCDBP . . . . . . . .    POINTER IN PROC (CLEARLISTS) PARAMETER        251  264  265  266
   42   0000H           CLEARCDBALARMS . . . .    PROCEDURE EXTERNAL(29) STACK=0000H       274
  250   035CH    109    CLEARLISTS . . . . . .    PROCEDURE BYTE PUBLIC STACK=0012H        273
  252   03C9H     79    CLEARONELIST . . . . .    PROCEDURE IN PROC (CLEARLISTS) STACK=000EH     264  265  266
   27                   CLOSED . . . . . . . .    LITERALLY '9'          513
  364   06E9H     26    CLOSEENQUEUE . . . . .    PROCEDURE IN PROC (CLOSERTN) STACK=001CH       376  377  381
   30                   CLOSEREQ . . . . . . .    LITERALLY '2'          212
  363   0697H     82    CLOSERTN . . . . . . .    PROCEDURE STACK=0022H          5
   27                   CLOSING. . . . . . . .    LITERALLY '8'          381
  253   0000H     32    CLRBS. . . . . . . . .    STRUCTURE BASED(LISTP) IN PROC (CLEARONELIST)
        0000H      1        CONTENTS . . . . .    BYTE
        0001H      1        CREDIT . . . . . .    BYTE
        0002H      2        LASTSEQ. . . . . .    WORD
        0004H      4        MIPBUFBASE . . . .    POINTER
        0008H      2        MIPLENGTH. . . . .    WORD
        000AH      1        MIPIDSID . . . . .    BYTE
        000BH      1        MIPOWNERDEVID. . .    BYTE
```

```
      000CH     2       INTERNALPROCESSID.    WORD
      000EH     1       REQ. . . . . . . .    BYTE
      000FH     1       RESP . . . . . . .    BYTE       256*
      0010H     2       RTNMIPSKT. . . . .    WORD
      0012H     4       LINK . . . . . . .    POINTER          255  257*
      0016H     2       CID. . . . . . . .    WORD
      0018H     2       FIRSTSEQ . . . . .    WORD
      001AH     2       CLIENTUSE. . . . .    WORD
      001CH     2       BUFLEN . . . . . .    WORD
      001EH     1       NUMBLKS. . . . . .    BYTE
      001FH     1       VB . . . . . . . .    BYTE
 251  0044H     1   CLRTNCODE. . . . . . .    BYTE IN PROC (CLEARLISTS) PARAMETER      251  256
  27                CLSWAIT. . . . . . . .    LITERALLY '7'
                    CMPW . . . . . . . . .    BUILTIN          310
 114  0000H         CQCHECKALARM . . . . .    PROCEDURE BYTE EXTERNAL(54) STACK=0000H
 117  0000H         CQCLEARALARM . . . . .    PROCEDURE EXTERNAL(55) STACK=0000H       339
 108  0000H         CQCREATEALARM. . . . .    PROCEDURE EXTERNAL(52) STACK=0000H       351  352
  55  0000H         CQCREATELIST . . . . .    PROCEDURE EXTERNAL(34) STACK=0000H
  78  0000H         CQCREATEMAILBOX. . . .    PROCEDURE EXTERNAL(42) STACK=0000H
  63  0000H         CQCREATEPROCESS. . . .    PROCEDURE EXTERNAL(37) STACK=0000H
  66  0000H         CQCREATESEMAPHORE. . .    PROCEDURE EXTERNAL(38) STACK=0000H
  87  0000H         CQCRECEIVE . . . . . .    PROCEDURE POINTER EXTERNAL(45) STACK=0000H
  75  0000H         CQCWAIT. . . . . . . .    PROCEDURE BYTE EXTERNAL(41) STACK=0000H
  22  0000H         CQDLLCONNECT . . . . .    PROCEDURE BYTE EXTERNAL(26) STACK=0000H
  16  0000H         CQDLLREAD. . . . . . .    PROCEDURE WORD EXTERNAL(24) STACK=0000H
  19  0000H         CQDLLREADC . . . . . .    PROCEDURE WORD EXTERNAL(25) STACK=0000H
  11  0000H         CQDLLRXRETBUF. . . . .    PROCEDURE EXTERNAL(22) STACK=0000H
  14  0000H         CQDLLSTART . . . . . .    PROCEDURE EXTERNAL(23) STACK=0000H
   7  0000H     2   CQDLLTXFREEMBX . . . .    WORD EXTERNAL(20)
   8  0000H         CQDLLTXSEND. . . . . .    PROCEDURE EXTERNAL(21) STACK=0000H
  60  0000H         CQHALTANDCATCHFIRE . .    PROCEDURE EXTERNAL(36) STACK=0000H       146
 102  0000H         CQICRECEIVE. . . . . .    PROCEDURE POINTER EXTERNAL(50) STACK=0000H
  96  0000H         CQICWAIT . . . . . . .    PROCEDURE BYTE EXTERNAL(48) STACK=0000H
  99  0000H         CQISEND. . . . . . . .    PROCEDURE EXTERNAL(49) STACK=0000H
  93  0000H         CQISIGNAL. . . . . . .    PROCEDURE EXTERNAL(47) STACK=0000H
 123  0000H         CQMIPCONNECT . . . . .    PROCEDURE BYTE EXTERNAL(57) STACK=0000H
 129  0000H         CQMIPGETADDRESS. . . .    PROCEDURE POINTER EXTERNAL(59) STACK=0000H     393  527
 132  0000H         CQMIPGETMIPFORM. . . .    PROCEDURE POINTER EXTERNAL(60) STACK=0000H
 126  0000H         CQMIPREGISTER. . . . .    PROCEDURE BYTE EXTERNAL(58) STACK=0000H
 120  0000H         CQMIPSEND. . . . . . .    PROCEDURE BYTE EXTERNAL(56) STACK=0000H
  90  0000H         CQMRECEIVE . . . . . .    PROCEDURE POINTER EXTERNAL(46) STACK=0000H
 105  0000H         CQREADCLOCK. . . . . .    PROCEDURE EXTERNAL(51) STACK=0000H       396
  84  0000H         CQRECEIVE. . . . . . .    PROCEDURE POINTER EXTERNAL(44) STACK=0000H     157  457  492
  58  0000H         CQSCHEDULE . . . . . .    PROCEDURE EXTERNAL(35) STACK=0000H       524
  81  0000H         CQSEND . . . . . . . .    PROCEDURE EXTERNAL(43) STACK=0000H       161  169  258  470  500
 111  0000H         CQSETALARM . . . . . .    PROCEDURE EXTERNAL(53) STACK=0000H
  69  0000H         CQSIGNAL . . . . . . .    PROCEDURE EXTERNAL(39) STACK=0000H       456  501  522
  53  0000H         CQSTART. . . . . . . .    PROCEDURE EXTERNAL(33) STACK=0000H
  72  0000H         CQWAITSEM. . . . . . .    PROCEDURE EXTERNAL(40) STACK=0000H       458  493
   4  0064H     1   CURCDBINDEX. . . . . .    BYTE       334*  336  373  487  506*  512  516  523
  26  001EH     4   CURCDBP. . . . . . . .    POINTER          150  195  196  199  201  204  221  231  232
                                               242  248  309  310  312  336  338  339  340  342  343
                                               345  351  352  370  373  421  424  426  427  430  449
                                               487  512  513  515  516
   4  0000H     2   CURCID . . . . . . . .    WORD       335*  511*  523
   3  0000H     1   CURMAXCDBS . . . . . .    BYTE EXTERNAL(3)       306  320
   28  0022H      4  CURRBSP. . . . . . . .     POINTER          169  188  198  206  214  234  235  243  280
                                                  285  287  293  299  310  342  349  350  353  421  425
                                                  492*  494  495  496  500  504  506  511  527*
   28  0026H      4  CURRBVP. . . . . . . .     POINTER          189  236  495*
  106  0000H      2  DATARTNO . . . . . . .     WORD IN PROC (CQREADCLOCK) PARAMETER     106
    3  0000H      2  DEFABORTTOHI . . . . .     WORD EXTERNAL(8)          346
  136  0000H     78  DEFERIRBTP . . . . . .     PROCEDURE PUBLIC REENTRANT STACK=0010H          150
    2                DEFNETIDLIT. . . . . .     LITERALLY '1'
    3  0000H      2  DEFPERSIST . . . . . .     WORD EXTERNAL(9)          344
    3  0000H      4  DEFRETRANTODW. . . . .     DWORD EXTERNAL(7)         347
   33  0000H     32  DEFST. . . . . . . . .     STRUCTURE BASED(DEFSTP)
       0000H      1      CONTENTS . . . . .     BYTE
       0001H      1      CREDIT . . . . . .     BYTE
       0002H      2      LASTSEQ. . . . . .     WORD
       0004H      4      MIPBUFBASE . . . .     POINTER
       0008H      2      MIPLENGTH. . . . .     WORD
       000AH      1      MIPIDSID . . . . .     BYTE
       000BH      1      MIPOWNERDEVID. . .     BYTE
       000CH      2      INTERNALPROCESSID.     WORD
       000EH      1      REQ. . . . . . . .     BYTE
       000FH      1      RESP . . . . . . .     BYTE
       0010H      2      RTNMIPSKT. . . . .     WORD
       0012H      4      LINK . . . . . . .     POINTER
       0016H      2      CID. . . . . . . .     WORD
       0018H      2      FIRSTSEQ . . . . .     WORD
       001AH      2      CLIENTUSE. . . . .     WORD
       001CH      2      BUFLEN . . . . . .     WORD
       001EH      1      NUMBLKS. . . . . .     BYTE
       001FH      1      VB . . . . . . . .     BYTE
   30                DEFSTATUSREQ . . . . .     LITERALLY '4'
  423  08B9H     57  DEFSTATUSRTN . . . . .     PROCEDURE STACK=0008H          5
   33  0032H      4  DEFSTP . . . . . . . .     POINTER
  112  0000H      2  DELAYH . . . . . . . .     WORD IN PROC (CQSETALARM) PARAMETER      112
  112  0000H      2  DELAYL . . . . . . . .     WORD IN PROC (CQSETALARM) PARAMETER      112
  271  0418H     58  DELETECDB. . . . . . .     PROCEDURE PUBLIC STACK=001EH        373  484  516
  172  0036H      2  DELTA. . . . . . . . .     WORD IN PROC (DELTASEQ)       173*  174  175*  175  176
  171  00F9H     43  DELTASEQ . . . . . . .     PROCEDURE WORD STACK=0004H          217  248
  137  0000H    124  DIPC . . . . . . . . .     STRUCTURE BASED(CDBP) IN PROC (DEFERIRBTP)
       0000H      1      STATE. . . . . . .     BYTE AUTOMATIC
       0001H      1      OWNERDEVICE. . . .     BYTE AUTOMATIC
       0002H      2      OWNERPROCESSID . .     WORD AUTOMATIC
       0004H      2      LOCCID . . . . . .     WORD AUTOMATIC
       0006H      2      LOCPORT. . . . . .     WORD AUTOMATIC
       0008H      2      REMNET . . . . . .     WORD AUTOMATIC
       000AH      6      REMHOST. . . . . .     WORD ARRAY(3) AUTOMATIC
       0010H      2      REMPORT. . . . . .     WORD AUTOMATIC
       0012H      2      PERSIST. . . . . .     WORD AUTOMATIC
       0014H      2      ABORTTOHI. . . . .     WORD AUTOMATIC
       0016H      2      REMCID . . . . . .     WORD AUTOMATIC
       0018H      4      RETRANTODW . . . .     DWORD AUTOMATIC
       001CH      2      RESERVED . . . . .     WORD AUTOMATIC
       001EH      2      TIMEDSEQNO . . . .     WORD AUTOMATIC
       0020H      4      SEGTRANSTIMEDW . .     DWORD AUTOMATIC
       0024H      4      CUMRETRANDW. . . .     DWORD AUTOMATIC
       0028H      2      PERSISTCNT . . . .     WORD AUTOMATIC
       002AH      4      CBTQHDR. . . . . .     POINTER AUTOMATIC
```

```
          002EH     4       PCBQHDR. . . . . .     POINTER AUTOMATIC
          0032H     4       DEFSTATUSP . . . .     POINTER AUTOMATIC
          0036H     2       MYACKNO. . . . . .     WORD AUTOMATIC
          0038H     2       SEEN . . . . . . .     WORD AUTOMATIC
          003AH     1       MYCREDIT . . . . .     BYTE AUTOMATIC
          003BH     1       CURBLKINDEX. . . .     BYTE AUTOMATIC
          003CH     2       CBDATAINDEX. . . .     WORD AUTOMATIC
          003EH     2       RCVBYTESCONSUMED .     WORD AUTOMATIC
          0040H     2       CURBLKLENLEFT. . .     WORD AUTOMATIC
          0042H     2       HISACKNO . . . . .     WORD AUTOMATIC
          0044H     2       NEXTTRANSMIT . . .     WORD AUTOMATIC
          0046H     1       CLOSEDREASON . . .     BYTE AUTOMATIC
          0047H     1       HISCREDIT. . . . .     BYTE AUTOMATIC
          0048H     2       HIGHESTSENT. . . .     WORD AUTOMATIC
          004AH     1       CBTQBUFCNT . . . .     BYTE AUTOMATIC
          004BH     1       PCBQBUFCNT . . . .     BYTE AUTOMATIC
          004CH     2       PKTSREJ. . . . . .     WORD AUTOMATIC
          004EH     2       PKTSRETRAN . . . .     WORD AUTOMATIC
          0050H     2       NOCONFID . . . . .     WORD AUTOMATIC
          0052H     2       LASTNOCONFID . . .     WORD AUTOMATIC
          0054H     1       RETRANSMITSTATE. .     BYTE AUTOMATIC
          0055H     1       SENDFLAG . . . . .     BYTE AUTOMATIC          139  142*
          0056H     2       PENDINGRCVDATA . .     WORD AUTOMATIC
          0058H     2       RCVBUFREJCNT . . .     WORD AUTOMATIC
          005AH     2       AYTCOUNT . . . . .     WORD AUTOMATIC
          005CH     4       DATAALARMCB. . . .     WORD ARRAY(2) AUTOMATIC
          0060H     1       DATAACBIRBTYPE . .     BYTE AUTOMATIC
          0061H     1       DATAACBFLAG. . . .     BYTE AUTOMATIC
          0062H    10       DATAACBREM . . . .     BYTE ARRAY(10) AUTOMATIC
          006CH     4       CTLALARMCB . . . .     WORD ARRAY(2) AUTOMATIC
          0070H     1       CTLACBIRBTYPE. . .     BYTE AUTOMATIC
          0071H     1       CTLACBFLAG . . . .     BYTE AUTOMATIC
          0072H    10       CTLACBREM. . . . .     BYTE ARRAY(10) AUTOMATIC
   272    000AH     1     DLCDBINDEX . . . . . .   BYTE IN PROC (DELETECDB) PARAMETER AUTOMATIC         272
   272    0006H     4     DLCDBP . . . . . . . .   POINTER IN PROC (DELETECDB) PARAMETER AUTOMATIC      272  273  274
     2                    DLLHEADERLEN . . . . .   LITERALLY '14'
   272    0004H     1     DLRTNCODE. . . . . . .   BYTE IN PROC (DELETECDB) PARAMETER AUTOMATIC         272  273
   193    0185H   271     ENTERCBTQ. . . . . . .   PROCEDURE STACK=0010H        367  431  432  433  439
   229    0294H   200     ENTERPCBQ. . . . . . .   PROCEDURE STACK=000AH        451
    64    0000H     2     ENTRYO . . . . . . . .   WORD IN PROC (CQCREATEPROCESS) PARAMETER        64
    61    0000H     2     ERRORCODE. . . . . . .   WORD IN PROC (CQHALTANDCATCHFIRE) PARAMETER         61
    27                    ESTAB. . . . . . . . .   LITERALLY '3'           199  426
     2                    FALSE. . . . . . . . .   LITERALLY '0'           211  263  328
    51    0000H     2     FINDTARGET . . . . . .   WORD IN PROC (SEARCH_LCIDVECTOR) PARAMETER      51
    27                    FINWAIT1 . . . . . . .   LITERALLY '4'           376  377
    27                    FINWAIT2 . . . . . . .   LITERALLY '5'
     4    0065H     1     FIRSTOPEN. . . . . . .   BYTE INITIAL      325  328*
     2                    FOREVER. . . . . . . .   LITERALLY 'WHILE true'       491
     6    0000H     2     FREEIRBMBX . . . . . .   WORD EXTERNAL(16)       157
     6    0000H     2     FREELIRBMBX. . . . . .   WORD EXTERNAL(17)       457
    27                    FROMLISTEN . . . . . .   LITERALLY '80H'
   386    0703H   376     GETSTATUSINFO. . . . .   PROCEDURE BYTE PUBLIC STACK=000AH        421
    25                    HACFCHKACB . . . . . .   LITERALLY '438'
    25                    HACFDLLCONN. . . . . .   LITERALLY '401'
    25                    HACFDLLREADHOST. . . .   LITERALLY '402'
    25                    HACFIPDEFIRB . . . . .   LITERALLY '432'         146
   25                   HACFMIPCONNECT . . . .     LITERALLY '404'
   25                   HACFMIPREGISTER. . . .     LITERALLY '434'
   25                   HACFMIPSEND. . . . . .     LITERALLY '416'
   25                   HACFPUTCNT . . . . . .     LITERALLY '405'
   25                   HACFRPCOMPLETE . . . .     LITERALLY '413'
   25                   HACFRPDEFIRB . . . . .     LITERALLY '433'
   25                   HACFRPPUTDATAP . . . .     LITERALLY '412'
   25                   HACFRPPUTNEWBLK. . . .     LITERALLY '409'
   25                   HACFRPPUTPTR . . . . .     LITERALLY '411'
   25                   HACFRPPUTSENDRBSBACK .     LITERALLY '410'
   25                   HACFSENDBLKLEN . . . .     LITERALLY '403'
   25                   HACFTPGETBLKLEN. . . .     LITERALLY '408'
   25                   HACFTPGETCHK . . . . .     LITERALLY '407'
   25                   HACFTPIRBACB . . . . .     LITERALLY '415'
   25                   HACFTPIRBP . . . . . .     LITERALLY '414'
   25                   HACFTPIRBTYPE. . . . .     LITERALLY '406'
   25                   HACFTPTIMEOUT. . . . .     LITERALLY '435'
   25                   HACFTQSEQ. . . . . . .     LITERALLY '417'
   25                   HACFXQ . . . . . . . .     LITERALLY '436'
   25                   HACFZQ . . . . . . . .     LITERALLY '437'
   30                   ILLEGALREQ . . . . . .     LITERALLY '10'        282  289  301  378  379  380  382  383
                                                    444
   37  0000H       1    INDEX. . . . . . . . .     BYTE IN PROC (SETUPCDB) PARAMETER        37
   67  0000H       2    INIT . . . . . . . . .     WORD IN PROC (CQCREATESEMAPHORE) PARAMETER      67
   30                   INVALIDPOINTER . . . .     LITERALLY '20'
   30                   INVALIDREQ . . . . . .     LITERALLY '2'         499
       0000H            IP . . . . . . . . . .     PROCEDURE STACK=0000H
  148  004EH      27    IPDEFERIRBTP . . . . .     PROCEDURE STACK=0016H       359  368  440
    6  0000H       2    IPINMBX. . . . . . . .     WORD EXTERNAL(15)      492
  135  006BH       1    IPIRBINDEX . . . . . .     BYTE INITIAL      150  523
  135  0068H       3    IPIRBLIST. . . . . . .     BYTE ARRAY(3) INITIAL       150  523
  522  0B29H            IPPOSTPROC . . . . . .     LABEL IN PROC (IPPROC)      509  517
  490  0A37H     292    IPPROC . . . . . . . .     PROCEDURE PUBLIC STACK=0022H
  492  0A3AH            IPTOP. . . . . . . . .     LABEL IN PROC (IPPROC)      502
  165  0001H      40    IP_SEND_RBS_BACK . . .     PROCEDURE STACK=000CH       223  282  289  301  314  322  361
                                                    374  378  379  380  382  383  421  444  450  488  508
                                                    515
   31  0000H       8    IRB. . . . . . . . . .     STRUCTURE BASED(IRBO)
       0000H       4        CMXPTR . . . . . .     POINTER
       0004H       1        TYPE . . . . . . .     BYTE        158*
       0005H       1        CDBINDEX . . . . .     BYTE        159*
       0006H       2        CID. . . . . . . .     WORD        160*
   34                   IRBAYTTIMER. . . . . .     LITERALLY '7'
   31  0030H       2    IRBB . . . . . . . . .     WORD
   34                   IRBCTLACBMASK. . . . .     LITERALLY '40H'
   34                   IRBCTLTIMEOUTMASK. . .     LITERALLY '0C0H'
  137  0000H       1    IRBINDEX . . . . . . .     BYTE BASED(IRBINDEXO) IN PROC (DEFERIRBTP)      144*  144  145
  153  0000H       1    IRBINDEX . . . . . . .     BYTE BASED(IRBINDEXO) IN PROC (SENDDEFERREDIRBS)      154  156  163*
  153  000AH       2    IRBINDEXO. . . . . . .     WORD IN PROC (SENDDEFERREDIRBS) PARAMETER AUTOMATIC         153  154
                                                    156
  137  0006H       2    IRBINDEXO. . . . . . .     WORD IN PROC (DEFERIRBTP) PARAMETER AUTOMATIC         137  144  145
   34                   IRBINVALIDMASK . . . .     LITERALLY '38H'
  153  0000H       3    IRBLIST. . . . . . . .     BYTE BASED(IRBLISTO) ARRAY(3) IN PROC (SENDDEFERREDIRBS)         158
  137  0000H       3    IRBLIST. . . . . . . .     BYTE BASED(IRBLISTO) ARRAY(3) IN PROC (DEFERIRBTP)         143*
  153  0008H       2    IRBLISTO . . . . . . .     WORD IN PROC (SENDDEFERREDIRBS) PARAMETER AUTOMATIC         153  158
  137  0004H       2    IRBLISTO . . . . . . .     WORD IN PROC (DEFERIRBTP) PARAMETER AUTOMATIC         137
```

```
   34                 IRBMAXCODE . . . . . .   LITERALLY '7'
   34                 IRBNULL. . . . . . . .   LITERALLY '0FFH'
   31  002EH       2  IRBO . . . . . . . . .   WORD          31
   31  002EH       4  IRBP . . . . . . . . .   POINTER AT         157*  161  457*  470
   34                 IRBSENDCHECK . . . . .   LITERALLY '4'            138  368  440
   34                 IRBSENDFIN . . . . . .   LITERALLY '2'
   34                 IRBSENDFLAG. . . . . .   LITERALLY '5'            138  141
   34                 IRBSENDRST . . . . . .   LITERALLY '3'            460
   34                 IRBSENDSYN . . . . . .   LITERALLY '0'            359
   34                 IRBSENDSYNACK. . . . .   LITERALLY '1'
   34                 IRBTIMEOUTMASK . . . .   LITERALLY '80H'
   34                 IRBTIMEWAITTO. . . . .   LITERALLY '6'
   34                 IRBTYPEMASK. . . . . .   LITERALLY '0FH'
    4  0006H       2  J. . . . . . . . . . .   WORD        188*  188  189  190  306*  306  307  309  319
  194  006CH       1  J. . . . . . . . . . .   BYTE IN PROC (ENTERCBTQ)
  153  FFFFH       1  J. . . . . . . . . . .   BYTE IN PROC (SENDDEFERREDIRBS) AUTOMATIC      156*  156  158  162

    4  0002H       2  LASTCID. . . . . . . .   WORD PUBLIC        327*  330*  330  331*  331  332  335
  178  0124H      31  LASTRBSPTR . . . . . .   PROCEDURE POINTER STACK=0006H       204  242
  194  003EH       2  LASTSEQ. . . . . . . .   WORD IN PROC (ENTERCBTQ)      200*  201*  205*  216  217  220
    3  0000H          LCIDVECTOR . . . . . .   WORD ARRAY(0) EXTERNAL(1)           275*  335*
  253  004AH       4  LINKPTR. . . . . . . .   POINTER IN PROC (CLEARONELIST)      255*  260
   32  0000H      24  LIRB . . . . . . . . .   STRUCTURE BASED(IRBO)
       0000H       4      CMXPTR . . . . . .   POINTER
       0004H       1      TYPE . . . . . . .   BYTE        460*
       0005H       1      REASON . . . . . .   BYTE        459*
       0006H       6      DLDEST . . . . . .   WORD ARRAY(3)            461*  462*  463*
       000CH       2      DESTPORT . . . . .   WORD        464*
       000EH       2      SOURCEPORT . . . .   WORD        465*
       0010H       2      DESTCID. . . . . .   WORD        466*
       0012H       2      SOURCECID. . . . .   WORD        467*
       0014H       2      SEGSEQNO . . . . .   WORD        468*
       0016H       2      SEGACKNO . . . . .   WORD        469*
   27                 LISTEN . . . . . . . .   LITERALLY '0'            355
   56  0000H       4  LISTP. . . . . . . . .   POINTER IN PROC (CQCREATELIST) PARAMETER          56
  253  0004H       4  LISTP. . . . . . . . .   POINTER IN PROC (CLEARONELIST) PARAMETER AUTOMATIC          253  254
                                                255  258  260*
   30                 LOCABORT . . . . . . .   LITERALLY '12'           484
    3  0000H       6  LOCHOST. . . . . . . .   WORD ARRAY(3) EXTERNAL(6)           327
    3  0000H       2  LOCNET . . . . . . . .   WORD EXTERNAL(5)
   30                 LOCTIMEOUT . . . . . .   LITERALLY '16'
    2                 LOGRBMIPPORT . . . . .   LITERALLY '5'
  179  0004H       4  LRPP . . . . . . . . .   POINTER IN PROC (LASTRBSPTR) PARAMETER AUTOMATIC     179  180  181*
                                                181  183
  179  0000H      32  LRPRBS . . . . . . . .   STRUCTURE BASED(LRPP) IN PROC (LASTRBSPTR)
       0000H       1      CONTENTS . . . . .   BYTE
       0001H       1      CREDIT . . . . . .   BYTE
       0002H       2      LASTSEQ. . . . . .   WORD
       0004H       4      MIPBUFBASE . . . .   POINTER
       0008H       2      MIPLENGTH. . . . .   WORD
       000AH       1      MIPIDSID . . . . .   BYTE
       000BH       1      MIPOWNERDEVID. . .   BYTE
       000CH       2      INTERNALPROCESSID.   WORD
       000EH       1      REQ. . . . . . . .   BYTE
       000FH       1      RESP . . . . . . .   BYTE
       0010H       2      RTNMIPSKT. . . . .   WORD
```

```
        0012H     4        LINK . . . . . . . .      POINTER           180  181
        0016H     2        CID. . . . . . . . .      WORD
        0018H     2        FIRSTSEQ . . . . . .      WORD
        001AH     2        CLIENTUSE. . . . . .      WORD
        001CH     2        BUFLEN . . . . . . .      WORD
        001EH     1        NUMBLKS. . . . . . .      BYTE
        001FH     1        VB . . . . . . . . .      BYTE
   40   0000H     2    M. . . . . . . . . . . .      WORD IN PROC (MIN) PARAMETER          40
  112   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQSETALARM) PARAMETER       112
  103   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQICRECEIVE) PARAMETER      103
  100   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQISEND) PARAMETER          100
   88   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQCRECEIVE) PARAMETER        88
   85   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQRECEIVE) PARAMETER         85
   82   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQSEND) PARAMETER            82
   79   0000H     2    MAILBOXO . . . . . . . .      WORD IN PROC (CQCREATEMAILBOX) PARAMETER         79
    3   0000H     2    MAXSEGDATALEN. . . . . .      WORD EXTERNAL(10)      173  174
    2                  MAXSEGDATALENLIT . . . .      LITERALLY '1480'
    2                  MAXSENDSEG . . . . . . .      LITERALLY '07H'
    3   0000H     1    MAXWINDOWSIZE. . . . . .      BYTE EXTERNAL(11)
  124   0000H     2    MBXO . . . . . . . . . .      WORD IN PROC (CQMIPCONNECT) PARAMETER            124
   23   0000H     2    MBXO . . . . . . . . . .      WORD IN PROC (CQDLLCONNECT) PARAMETER             23
  100   0000H     4    MESSAGEP . . . . . . . .      POINTER IN PROC (CQISEND) PARAMETER       100
   82   0000H     4    MESSAGEP . . . . . . . .      POINTER IN PROC (CQSEND) PARAMETER         82
   39   0000H          MIN. . . . . . . . . . .      PROCEDURE WORD EXTERNAL(28) STACK=0000H
    2                  MINPKTLEN. . . . . . . .      LITERALLY '46'
    2                  MIPECHOPORT. . . . . . .      LITERALLY '7'
  130   0000H     4    MIP_FORM . . . . . . . .      POINTER IN PROC (CQMIPGETADDRESS) PARAMETER            130
   20   0000H     2    MODIFIER . . . . . . . .      WORD IN PROC (CQDLLREADC) PARAMETER        20
   17   0000H     2    MODIFIER . . . . . . . .      WORD IN PROC (CQDLLREAD) PARAMETER         17
                       MOVB . . . . . . . . . .      BUILTIN           397
                       MOVW . . . . . . . . . .      BUILTIN           342  402
  121   0000H     4    MSGP . . . . . . . . . .      POINTER IN PROC (CQMIPSEND) PARAMETER            121
   40   0000H     2    N. . . . . . . . . . . .      WORD IN PROC (MIN) PARAMETER          40
  330   0560H          NEWCID . . . . . . . . .      LABEL IN PROC (OPENRTN)        333
  365   0004H     1    NEWSTATE . . . . . . . .      BYTE IN PROC (CLOSEENQUEUE) PARAMETER AUTOMATIC       365  366
   30                  NORESOURCESRESP. . . . .      LITERALLY '4'          322
    3   0000H     1    NUMCDBS. . . . . . . . .      BYTE EXTERNAL(0)       276*  276  320  337*  337
   28   002AH     4    NXTRBP . . . . . . . . .      POINTER           494*  525  527
   17   0000H     2    OBJECT . . . . . . . . .      WORD IN PROC (CQDLLREAD) PARAMETER         17
   20   0000H     2    OBJECT . . . . . . . . .      WORD IN PROC (CQDLLREADC) PARAMETER        20
   30                  OKCLOSEDRESP . . . . . .      LITERALLY '9'          373
   30                  OKEOMRESP. . . . . . . .      LITERALLY '3'
   30                  OKFINRESP. . . . . . . .      LITERALLY '5'          450
   30                  OKRESP . . . . . . . . .      LITERALLY '1'          223  361  374  399  418  488
    2                  ONBDTCLECHOPORT. . . . .      LITERALLY '7'
   30                  OPENAREQ . . . . . . . .      LITERALLY '0'          287  312
   30                  OPENCONFLICT . . . . . .      LITERALLY '18'         314
   30                  OPENPREQ . . . . . . . .      LITERALLY '1'          312  353  504
  279   0452H   581    OPENRTN. . . . . . . . .      PROCEDURE STACK=001AH        5
   64   0000H     2    PCBO . . . . . . . . . .      WORD IN PROC (CQCREATEPROCESS) PARAMETER         64
  230   0000H    32    PCBQRBS. . . . . . . . .      STRUCTURE BASED(PCBQRBSP) IN PROC (ENTERPCBQ)
        0000H     1        CONTENTS . . . . . .      BYTE
        0001H     1        CREDIT . . . . . . .      BYTE
        0002H     2        LASTSEQ. . . . . . .      WORD
        0004H     4        MIPBUFBASE . . . . .      POINTER
        0008H     2        MIPLENGTH. . . . . .      WORD
```

```
      000AH    1       MIPIDSID . . . . .    BYTE
      000BH    1       MIPOWNERDEVID. . .    BYTE
      000CH    2       INTERNALPROCESSID.    WORD
      000EH    1       REQ. . . . . . . .    BYTE
      000FH    1       RESP . . . . . . .    BYTE
      0010H    2       RTNMIPSKT. . . . .    WORD
      0012H    4       LINK . . . . . . .    POINTER          243*
      0016H    2       CID. . . . . . . .    WORD
      0018H    2       FIRSTSEQ . . . . .    WORD
      001AH    2       CLIENTUSE. . . . .    WORD
      001CH    2       BUFLEN . . . . . .    WORD
      001EH    1       NUMBLKS. . . . . .    BYTE
      001FH    1       VB . . . . . . . .    BYTE
 230  0040H    4   PCBQRBSP . . . . . . .    POINTER IN PROC (ENTERPCBQ)         242*
   9  0000H    2   PKTO . . . . . . . . .    WORD IN PROC (CQDLLTXSEND) PARAMETER        9
 124  0000H    1   PORTID . . . . . . . .    BYTE IN PROC (CQMIPCONNECT) PARAMETER          124
 448  0934H   29   POSTRBUF . . . . . . .    PROCEDURE STACK=0010H          5
  30               POSTRBUFREQ. . . . . .    LITERALLY '7'
  64  0000H    2   PRI. . . . . . . . . .    WORD IN PROC (CQCREATEPROCESS) PARAMETER        64
 127  0000H    2   PROCEDUREO . . . . . .    WORD IN PROC (CQMIPREGISTER) PARAMETER         127
 493  0A49H        PROCESSRB. . . . . . .    LABEL IN PROC (IPPROC)         528
   5  000AH    2   PROCLOCATION . . . . .    WORD        520*  521
   5  000CH   18   PROCS. . . . . . . . .    WORD ARRAY(9) INITIAL          520
 133  0000H    4   PTR. . . . . . . . . .    POINTER IN PROC (CQMIPGETMIPFORM) PARAMETER          133
  48  0000H        RANDOM . . . . . . . .    PROCEDURE WORD EXTERNAL(31) STACK=0000H        327
 251  006DH    1   RBFOUND. . . . . . . .    BYTE IN PROC (CLEARLISTS)           259*  263*  269
  28  0000H   42   RBO. . . . . . . . . .    STRUCTURE BASED(CURRBSP)
      0000H    4       CMXPTR . . . . . .    POINTER
      0004H    4       MIPBUFBASE . . . .    POINTER
      0008H    2       MIPLENGTH. . . . .    WORD
      000AH    1       MIPIDSID . . . . .    BYTE
      000BH    1       MIPOWNERDEVID. . .    BYTE        349
      000CH    2       OWNERPROCESSID . .    WORD        350
      000EH    1       REQ. . . . . . . .    BYTE        287   353
      000FH    1       RESP . . . . . . .    BYTE
      0010H    2       RTNMIPSKT. . . . .    WORD
      0012H    4       LINK . . . . . . .    POINTER
      0016H    2       CID. . . . . . . .    WORD        335*  342
      0018H    2       LOCPORT. . . . . .    WORD        280   310
      001AH    2       REMNET . . . . . .    WORD
      001CH    6       REMHOST. . . . . .    WORD ARRAY(3)           285   310
      0022H    2       REMPORT. . . . . .    WORD        293   299
      0024H    2       PERSIST. . . . . .    WORD
      0026H    2       ABORTTIMEOUT . . .    WORD
      0028H    2       SEQ. . . . . . . .    WORD
  29  0000H   32   RBS. . . . . . . . . .    STRUCTURE BASED(CURRBSP)
      0000H    1       CONTENTS . . . . .    BYTE        211*  213*  214
      0001H    1       CREDIT . . . . . .    BYTE        248*
      0002H    2       LASTSEQ. . . . . .    WORD        217*  220*
      0004H    4       MIPBUFBASE . . . .    POINTER
      0008H    2       MIPLENGTH. . . . .    WORD
      000AH    1       MIPIDSID . . . . .    BYTE
      000BH    1       MIPOWNERDEVID. . .    BYTE
      000CH    2       INTERNALPROCESSID.    WORD
      000EH    1       REQ. . . . . . . .    BYTE        496
      000FH    1       RESP . . . . . . .    BYTE        167*  499*
```

```
        0010H    2        RTNMIPSKT. . . . .    WORD
        0012H    4        LINK . . . . . . .    POINTER          168*  210*  246*  424*  494
        0016H    2        CID. . . . . . . .    WORD        504  506  511
        0018H    2        FIRSTSEQ . . . . .    WORD        216*  220*
        001AH    2        CLIENTUSE. . . . .    WORD
        001CH    2        BUFLEN . . . . . .    WORD        209*  245*
        001EH    1        NUMBLKS. . . . . .    BYTE        188  235
        001FH    1        VB . . . . . . . .    BYTE        495
   4    0067H    1    RBS_REQ. . . . . . . .    BYTE        212  312  496*  497  504  520
  29    0000H    6    RBV. . . . . . . . . .    STRUCTURE BASED(CURRBVP) ARRAY(1)
        0000H    4        BLKPTR . . . . . .    POINTER
        0004H    2        BLKLEN . . . . . .    WORD        189  236
  30                  REMABORT . . . . . . .    LITERALLY '14'
  30                  REQMAX . . . . . . . .    LITERALLY '8'          497
 166    0004H    1    RESP_CODE. . . . . . .    BYTE IN PROC (IP_SEND_RBS_BACK) PARAMETER AUTOMATIC          166  167
  20    0000H    4    RETURNBUFP . . . . . .    POINTER IN PROC (CQDLLREADC) PARAMETER           20
  17    0000H    4    RETURNBUFP . . . . . .    POINTER IN PROC (CQDLLREAD) PARAMETER            17
   6    0000H    2    RPMBX. . . . . . . . .    WORD EXTERNAL(14)
  35                  RSTCLIENTABORT . . . .    LITERALLY '6'          459
  35                  RSTCONNCLOSED. . . . .    LITERALLY '2'
  35                  RSTILLEGALACK. . . . .    LITERALLY '7'
  35                  RSTNOMATCH . . . . . .    LITERALLY '3'
  35                  RSTOLDDUPL . . . . . .    LITERALLY '1'
  35                  RSTSYNREFUSED. . . . .    LITERALLY '5'
  35                  RSTVERSIONMISMATCH . .    LITERALLY '8'
  35                  RSTZERODESTCID . . . .    LITERALLY '4'
   3    0000H    4    RTCDW. . . . . . . . .    DWORD EXTERNAL(12)     396
   6    0000H    2    SCHEDLOCK. . . . . . .    WORD EXTERNAL(19)      456  458  493  501  522
   4    0008H    2    SCRATCH. . . . . . . .    WORD        273*
  50    0000H         SEARCH_LCIDVECTOR. . .    PROCEDURE WORD EXTERNAL(32) STACK=0000H        332  334  506
  97    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQICWAIT) PARAMETER         97
  94    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQISIGNAL) PARAMETER        94
  76    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQCWAIT) PARAMETER          76
  73    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQWAITSEM) PARAMETER        73
  70    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQSIGNAL) PARAMETER         70
  67    0000H    2    SEMAPHOREO . . . . . .    WORD IN PROC (CQCREATESEMAPHORE) PARAMETER     67
 152    0069H  104    SENDDEFERREDIRBS . . .    PROCEDURE PUBLIC REENTRANT STACK=0014H         523
  30                  SENDEOMREQ . . . . . .    LITERALLY '6'          212
  30                  SENDREQ. . . . . . . .    LITERALLY '5'
 429    08F2H   66    SENDRTN. . . . . . . .    PROCEDURE STACK=001AH          5
 325    053DH         SETCID . . . . . . . .    LABEL IN PROC (OPENRTN)
  36    0000H         SETUPCDB . . . . . . .    PROCEDURE EXTERNAL(27) STACK=0000H      309  336  512
                      SETW . . . . . . . . .    BUILTIN          340
                      SHL. . . . . . . . . .    BUILTIN          327
 121    0000H    2    SOCKET . . . . . . . .    WORD IN PROC (CQMIPSEND) PARAMETER       121
   3    0000H         SPECTYPE . . . . . . .    BYTE ARRAY(0) EXTERNAL(2)          277*  307  341*
   4    0066H    1    SPECTYPETEMP . . . . .    BYTE        294*  295*  304*  341
  64    0000H    2    STACKO . . . . . . . .    WORD IN PROC (CQCREATEPROCESS) PARAMETER        64
  27                  STATEMASK. . . . . . .    LITERALLY '0FH'        199  370  426  430  449  472
 388    087BH   38    STATUSBUFTOOSHORT. . .    PROCEDURE BYTE IN PROC (GETSTATUSINFO) STACK=0006H          394  400

  30                  STATUSREQ. . . . . . .    LITERALLY '3'          504
 420    08A1H   24    STATUSRTN. . . . . . .    PROCEDURE STACK=0010H          5
 387    0000H   69    STB. . . . . . . . . .    STRUCTURE BASED(STBP) IN PROC (GETSTATUSINFO)        394  397  400

        0000H    1        TCLSTATE . . . . .    BYTE        397
```

```
      0001H    2      DEFABORT . . . . .    WORD
      0003H    4      DEFRETRANDW. . . .    DWORD
      0007H    2      DEFPERSIST . . . .    WORD
      0009H    1      CURMAXCDBS . . . .    BYTE
      000AH    1      NUMCDBS. . . . . .    BYTE
      000BH    2      LOCNET . . . . . .    WORD
      000DH    6      LOCHOST. . . . . .    WORD ARRAY(3)
      0013H    2      TOTPKTSREJ . . . .    WORD
      0015H    2      TOTRETRANEVENTS. .    WORD
      0017H    2      TOTRCVBUFREJ . . .    WORD
      0019H    4      RTCDW. . . . . . .    DWORD
      001DH    2      LOCPORT. . . . . .    WORD        394  397  402
      001FH    2      REMNET . . . . . .    WORD
      0021H    6      REMHOST. . . . . .    WORD ARRAY(3)
      0027H    2      REMPORT. . . . . .    WORD
      0029H    2      LOCCID . . . . . .    WORD        403*
      002BH    2      REMCID . . . . . .    WORD        404*
      002DH    2      CONNABORT. . . . .    WORD        405*
      002FH    4      CONNRETRANTODW . .    DWORD       406*
      0033H    2      CONNPERSIST. . . .    WORD        407*
      0035H    1      CONNSTATE. . . . .    BYTE        408*
      0036H    2      PENDINGRCVDATA . .    WORD        409*
      0038H    2      RCVBUFREJCNT . . .    WORD        410*
      003AH    1      CBTQBUFCNT . . . .    BYTE        411*
      003BH    1      PCBQBUFCNT . . . .    BYTE        412*
      003CH    1      LOCCREDIT. . . . .    BYTE        413*
      003DH    1      REMCREDIT. . . . .    BYTE        414*
      003EH    2      HIGHESTSENT. . . .    WORD        415*
      0040H    2      MYACKNO. . . . . .    WORD        416*
      0042H    2      NOCONFID . . . . .    WORD        417*
      0044H    1      LASTENTRY. . . . .    BYTE        400
  387 005AH    4  STBP . . . . . . . . .    POINTER IN PROC (GETSTATUSINFO)          393*  394  397  400  402
  387 0000H  124  STC. . . . . . . . . .    STRUCTURE BASED(STCDBP) IN PROC (GETSTATUSINFO)
      0000H    1      STATE. . . . . . .    BYTE        408
      0001H    1      OWNERDEVICE. . . .    BYTE
      0002H    2      OWNERPROCESSID . .    WORD
      0004H    2      LOCCID . . . . . .    WORD        403
      0006H    2      LOCPORT. . . . . .    WORD        402
      0008H    2      REMNET . . . . . .    WORD
      000AH    6      REMHOST. . . . . .    WORD ARRAY(3)
      0010H    2      REMPORT. . . . . .    WORD
      0012H    2      PERSIST. . . . . .    WORD        407
      0014H    2      ABORTTOHI. . . . .    WORD        405
      0016H    2      REMCID . . . . . .    WORD        404
      0018H    4      RETRANTODW . . . .    DWORD       406
      001CH    2      RESERVED . . . . .    WORD
      001EH    2      TIMEDSEQNO . . . .    WORD
      0020H    4      SEGTRANSTIMEDW . .    DWORD
      0024H    4      CUMRETRANDW. . . .    DWORD
      0028H    2      PERSISTCNT . . . .    WORD
      002AH    4      CBTQHDR. . . . . .    POINTER
      002EH    4      PCBQHDR. . . . . .    POINTER
      0032H    4      DEFSTATUSP . . . .    POINTER
      0036H    2      MYACKNO. . . . . .    WORD        416
      0038H    2      SEEN . . . . . . .    WORD
      003AH    1      MYCREDIT . . . . .    BYTE        413
```

```
          0033H     1       CURBLKINDEX. . . .    BYTE
          003CH     2       CBDATAINDEX. . . .    WORD
          003EH     2       RCVBYTESCONSUMED .    WORD
          0040H     2       CURBLKLENLEFT. . .    WORD
          0042H     2       HISACKNO . . . . .    WORD
          0044H     2       NEXTTRANSMIT . . .    WORD
          0046H     1       CLOSEDREASON . . .    BYTE
          0047H     1       HISCREDIT. . . . .    BYTE        414
          0048H     2       HIGHESTSENT. . . .    WORD        415
          004AH     1       CBTQBUFCNT . . . .    BYTE        411
          004BH     1       PCBQBUFCNT . . . .    BYTE        412
          004CH     2       PKTSREJ. . . . . .    WORD
          004EH     2       PKTSRETRAN . . . .    WORD
          0050H     2       NOCONFID . . . . .    WORD        417
          0052H     2       LASTNOCONFID . . .    WORD
          0054H     1       RETRANSMITSTATE. .    BYTE
          0055H     1       SENDFLAG . . . . .    BYTE
          0056H     2       PENDINGRCVDATA . .    WORD        409
          0058H     2       RCVBUFREJCNT . . .    WORD        410
          005AH     2       AYTCOUNT . . . . .    WORD
          005CH     4       DATAALARMCB. . . .    WORD ARRAY(2)
          0060H     1       DATAACBIRBTYPE . .    BYTE
          0061H     1       DATAACBFLAG. . . .    BYTE
          0062H    10       DATAACBREM . . . .    BYTE ARRAY(10)
          006CH     4       CTLALARMCB . . . .    WORD ARRAY(2)
          0070H     1       CTLACBIRBTYPE. . .    BYTE
          0071H     1       CTLACBFLAG . . . .    BYTE
          0072H    10       CTLACBREM. . . . .    BYTE ARRAY(10)
     387  0052H     4   STCDBP . . . . . . . .    POINTER IN PROC (GETSTATUSINFO) PARAMETER      387  402  403  404
                                                   405  406  407  408  409  410  411  412  413  414  415
                                                   416  417
     387  0000H    32   STRBS. . . . . . . . .    STRUCTURE BASED(STRBSP) IN PROC (GETSTATUSINFO)
          0000H     1       CONTENTS . . . . .    BYTE
          0001H     1       CREDIT . . . . . .    BYTE
          0002H     2       LASTSEQ. . . . . .    WORD
          0004H     4       MIPBUFBASE . . . .    POINTER
          0008H     2       MIPLENGTH. . . . .    WORD
          000AH     1       MIPIDSID . . . . .    BYTE
          000BH     1       MIPOWNERDEVID. . .    BYTE
          000CH     2       INTERNALPROCESSID.    WORD
          000EH     1       REQ. . . . . . . .    BYTE
          000FH     1       RESP . . . . . . .    BYTE
          0010H     2       RTNMIPSKT. . . . .    WORD
          0012H     4       LINK . . . . . . .    POINTER
          0016H     2       CID. . . . . . . .    WORD        398
          0018H     2       FIRSTSEQ . . . . .    WORD
          001AH     2       CLIENTUSE. . . . .    WORD
          001CH     2       BUFLEN . . . . . .    WORD
          001EH     1       NUMBLKS. . . . . .    BYTE        390
          001FH     1       VB . . . . . . . .    BYTE        392
     387  004EH     4   STRBSP . . . . . . . .    POINTER IN PROC (GETSTATUSINFO) PARAMETER      387  390  392  398
     387  0000H     6   STRBV. . . . . . . . .    STRUCTURE BASED(STRBVP) ARRAY(1) IN PROC (GETSTATUSINFO)
          0000H     4       BLKPTR . . . . . .    POINTER           393
          0004H     2       BLKLEN . . . . . .    WORD        390
     387  0056H     4   STRBVP . . . . . . . .    POINTER IN PROC (GETSTATUSINFO)          390  392*  393
     186  0038H     2   SUM. . . . . . . . . .    WORD IN PROC (SUMOFBLOCKS)         187*  189*  189  191
```

```
   185  0143H    66  SUMOFBLOCKS. . . . . .     PROCEDURE WORD STACK=0002H          208  247
    27               SYNRCVD. . . . . . . .     LITERALLY '2'
    27               SYNSENT. . . . . . . .     LITERALLY '1'          312  358
   372  06C1H        SYNSENTCASE. . . . . .     LABEL IN PROC (CLOSERTN)      371
     2               TCLHEADERLEN . . . . .     LITERALLY '20'
     2               TCLMIPPORT . . . . . .     LITERALLY '4'
     2               TCLPROTOCOLCODE. . . .     LITERALLY '5001H'
     2               TCLPROTOCOLCODEREV . .     LITERALLY '0150H'
     3  0000H     1  TCLSTATE . . . . . . .     BYTE EXTERNAL(4)       397
     2               TCLVERSIONLIT. . . . .     LITERALLY '101H'
     2               TIMEOUTINCREASESTATE .     LITERALLY '1'
     2               TIMEOUTSTEADYSTATE . .     LITERALLY '0'
    27               TIMEWAIT . . . . . . .     LITERALLY '6'          449
     4  0004H     2  TOTBUFLEN. . . . . . .     WORD       208*  209  212  217  247*  248
     6  0000H     2  TPMBX. . . . . . . . .     WORD EXTERNAL(13)      161  470
     2               TRUE . . . . . . . . .     LITERALLY 'OFFH'         4  142  213  259  491
   149  0004H     1  TYPE . . . . . . . . .     BYTE IN PROC (IPDEFERIRBTP) PARAMETER AUTOMATIC       149  150
   137  000CH     1  TYPE . . . . . . . . .     BYTE IN PROC (DEFERIRBTP) PARAMETER AUTOMATIC         137  138  141
                                                 143
    23  0000H     2  TYPE . . . . . . . . .     WORD IN PROC (CQDLLCONNECT) PARAMETER            23
    30               UNKNOWNCIDRESP . . . .     LITERALLY '6'          508
    91  0000H     2  WCBO . . . . . . . . .     WORD IN PROC (CQMRECEIVE) PARAMETER       91


MODULE INFORMATION:

     CODE AREA SIZE     = 0B53H   2907D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 006EH    110D
     MAXIMUM STACK SIZE = 0026H     38D
     2314 LINES READ
     1 PROGRAM WARNING
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```