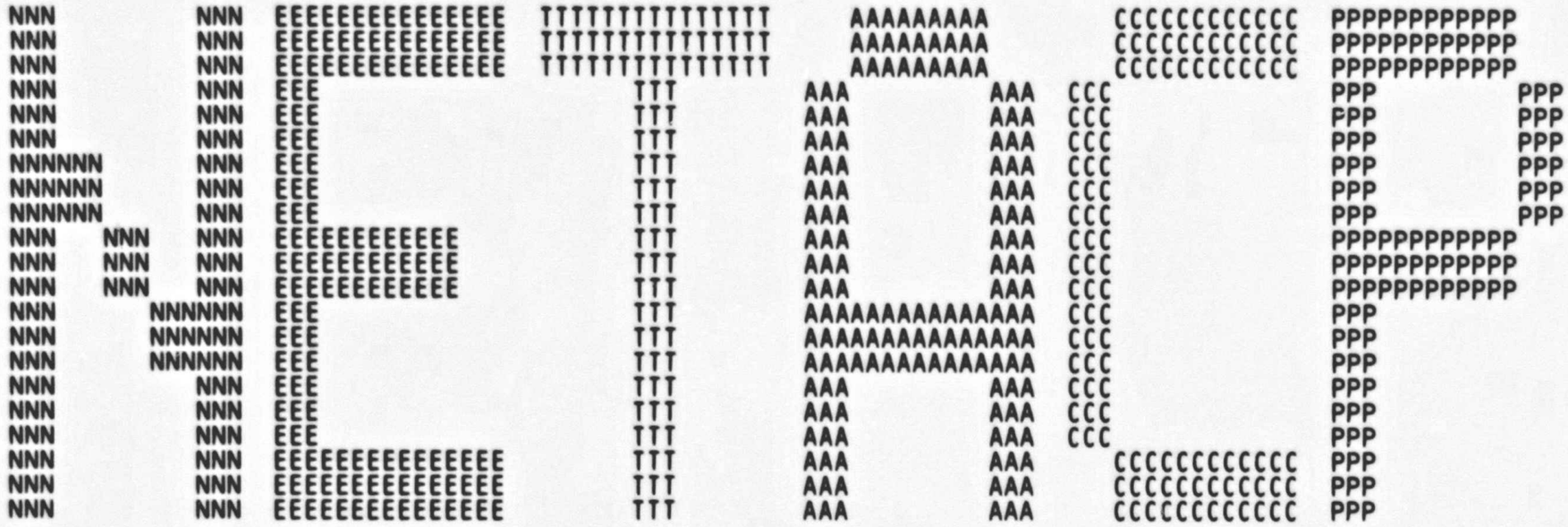
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT    AAAAAA      CCCCCCCC  PPPPPPPP    TTTTTTTTTT  RRRRRRRR    NN      NN
NN      NN  EEEEEEEEEE  TTTTTTTTTT    AAAAAA      CCCCCCCC  PPPPPPPP    TTTTTTTTTT  RRRRRRRR    NN      NN
NN      NN  EE              TT      AA      AA  CC          PP      PP      TT      RR      RR  NN      NN
NN      NN  EE              TT      AA      AA  CC          PP      PP      TT      RR      RR  NN      NN
NNNN    NN  EE              TT      AA      AA  CC          PP      PP      TT      RR      RR  NNNN    NN
NNNN    NN  EE              TT      AA      AA  CC          PP      PP      TT      RR      RR  NNNN    NN
NN  NN  NN  EEEEEEEE        TT      AA      AA  CC          PPPPPPPP        TT      RRRRRRRR    NN  NN  NN
NN  NN  NN  EEEEEEEE        TT      AA      AA  CC          PPPPPPPP        TT      RRRRRRRR    NN  NN  NN
NN    NNNN  EE              TT      AAAAAAAAAA  CC          PP              TT      RR  RR      NN    NNNN
NN    NNNN  EE              TT      AAAAAAAAAA  CC          PP              TT      RR  RR      NN    NNNN
NN      NN  EE              TT      AA      AA  CC          PP              TT      RR    RR    NN      NN
NN      NN  EE              TT      AA      AA  CC          PP              TT      RR    RR    NN      NN
NN      NN  EEEEEEEEEE      TT      AA      AA    CCCCCCCC  PP              TT      RR      RR  NN      NN
NN      NN  EEEEEEEEEE      TT      AA      AA    CCCCCCCC  PP              TT      RR      RR  NN      NN

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  NETACPTRN  - Control network local node state transitions
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,WORD
0000     4
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ; FACILITY:     NETWORK ACP
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;
0000    32 ;       This module performs ACP state transitions including initialization,
0000    33 ;       termination, and  allocating/deallocating I/O database control blocks
0000    34 ;       in system non-paged pool.
0000    35 ;
0000    36 ; ENVIRONMENT:
0000    37 ;
0000    38 ;       Kernel mode
0000    39 ;
0000    40 ;       .SBTTL  HISTORY
0000    41 ;
0000    42 ; AUTHOR:       SCOTT G. DAVIS, CREATION DATE: 03-AUG-77
0000    43 ;
0000    44 ; MODIFIED BY:
0000    45 ;
0000    46 ;       V03-022 PRB0343         Paul Beck       24-Jul-1984  20:48
0000    47 ;               Synchronize with NETDRIVER before decrementing transaction count.
0000    48 ;
0000    49 ;       V021    RNG0021         Rod Gamache     27-Mar-1984
0000    50 ;               Fix problem in timer code in previous modification, the
0000    51 ;               WQE was never being deallocated.
0000    52 ;
0000    53 ;       V020    RNG0020         Rod Gamache     13-Feb-1984
0000    54 ;               Change macro call from $NDCDEF to $LLIDEF. Add new timer
0000    55 ;               element that allows the XPORT layer to stop sending
0000    56 ;               'hello' messages in the event the cluster stalls and the
0000    57 ;               NETACP process cannot process events.
```

```
0000     58 ;
0000     59 ;       V019    TMH0019         Tim Halvorsen   11-Jul-1983
0000     60 ;               Store new LNI alias (cluster address) parameter in RCB.
0000     61 ;
0000     62 ;       V018    TMH0018         Tim Halvorsen   06-Apr-1983
0000     63 ;               Reduce initial size of virtual pool, since automatic
0000     64 ;               pool expansion will increase it when necessary.
0000     65 ;
0000     66 ;       V017    TMH0017         Tim Halvorsen   05-Mar-1983
0000     67 ;               Add initialization of DLE driver during NETACP
0000     68 ;               initialization.
0000     69 ;               Support SEGMENT BUFFER SIZE.
0000     70 ;
0000     71 ;       V016    TMH0016         Tim Halvorsen   14-Feb-1983
0000     72 ;               Make buffer size calculation assume a point-to-point
0000     73 ;               datalink, thus backing out TMH0015.  Instead, the
0000     74 ;               line BUS action routine will add the additional Ethernet
0000     75 ;               route header overhead if it is an NI datalink.
0000     76 ;
0000     77 ;       V015    TMH0015         Tim Halvorsen   14-Jan-1983
0000     78 ;               Fix problem with sending messages (close to or equal
0000     79 ;               to the segment buffer size) between two older ECLs
0000     80 ;               thru an intermediate node having an Ethernet datalink.
0000     81 ;
0000     82 ;       V014    TMH0014         Tim Halvorsen   07-Dec-1982
0000     83 ;               Fix bugs in deallocation of control blocks during
0000     84 ;               shutdown, which was causing a non-fatal bugcheck.
0000     85 ;
0000     86 ;       V013    TMH0013         Tim Halvorsen   13-Oct-1982
0000     87 ;               Add area routing support.
0000     88 ;               Allow STATE to be changed to SHUT while the executor
0000     89 ;               address is still not set, so that the network can be
0000     90 ;               shutdown with STATE SHUT at this point.
0000     91 ;               Double size of virtual pool - it was getting close.
0000     92 ;
0000     93 ;       V012    TMH0012         Tim Halvorsen   15-Sep-1982
0000     94 ;               Do not use PSI mutex channel for PSI incoming
0000     95 ;               call declaration, since deassigning the channel
0000     96 ;               (when the parameter is cleared) causes the mutex
0000     97 ;               to be lost, and PSI to clean out it's databases.
0000     98 ;               Modify PSI declare code so that it doesn't report
0000     99 ;               an error if PSI is not yet initialized.  Make it
0000    100 ;               callable so that when the declaration is actually
0000    101 ;               needed, it can be tried again when PSI is loaded.
0000    102 ;               Remove check that MAX BUFFERS must be greater than
0000    103 ;               MAX CIRCUITS, because it was never really needed.
0000    104 ;               Allow MAX CIRCUITS to be reduced as long as no active
0000    105 ;               LPD slot is removed in the process.
0000    106 ;
0000    107 ;       V011    TMH0011         Tim Halvorsen   07-Jul-1982
0000    108 ;               Change LPD vector into a vector of longword pointers
0000    109 ;               to the actual LPD blocks.
0000    110 ;               Remove allocation of local IRP.
0000    111 ;               Allocate the local LPD when the LPD vector is being
0000    112 ;               created, and use the LPD index LPD$C_LOC_INX to represent
0000    113 ;               it, rather than a negative index.
0000    114 ;               Move code to initialize routing database into DLLTRN,
```

```
0000   115 ;               and add a call here to a routine to initialize routing.
0000   116 ;               Do not allow EXECUTOR TYPE to be changed unless there
0000   117 ;               are no active LPDs.
0000   118 ;
0000   119 ;       V010    TMH0010         Tim Halvorsen   22-Jun-1982
0000   120 ;               Fix allocation of private ACP pool.  The last page
0000   121 ;               was not getting allocated properly.
0000   122 ;               Initialize CXB_FREE listhead in RCB.
0000   123 ;               Initialize local LPD to look just like a normal LPD.
0000   124 ;               Add $DYNDEF definition.
0000   125 ;
0000   126 ;       V009    TMH0009         Tim Halvorsen   04-Apr-1982
0000   127 ;               Remove unused NOP instructions.
0000   128 ;               Add check for error trying to lock down pages.
0000   129 ;               Change all references to SCH$GL_CURPCB to CTL$GL_PCB.
0000   130 ;               Use G^ for absolute references to executive symbols.
0000   131 ;               Change check for MNT already set into same segment
0000   132 ;               which checks for DMT already set.
0000   133 ;               Allocate AQB along with VCB, LPD, local IRP and TQE.
0000   134 ;               Remove storing of MY_PID, since it is never used.
0000   135 ;               Increase pool size to nice round number.
0000   136 ;               Use .ADDRESS to generate longword addresses.
0000   137 ;               Fix psect naming conventions
0000   138 ;               Change OPCOM calling interface.
0000   139 ;               Add code to declare NETACP as a PSI process (which
0000   140 ;               accepts incoming X.25 calls) if the EXECUTOR SUBADDRESS
0000   141 ;               parameter is specified.
0000   142 ;               Log event 2.0 if the executor state is changed.
0000   143 ;
0000   144 ;       V008    TMH0008         Tim Halvorsen   05-Mar-1982
0000   145 ;               Mark ACP in "dismount" state when the mount count goes
0000   146 ;               to zero, to avoid a race between the final DLE XWB coming
0000   147 ;               back from NETDRIVER and a new ACCESS function coming in
0000   148 ;               from a user.  The "dismount" state will signal the EXEC
0000   149 ;               to reject the QIO request.
0000   150 ;               Fix ACP state transition table, to disallow changing from
0000   151 ;               the OFF state to the SHUT state.  This closes off a window
0000   152 ;               which could cause the mount count to stay positive forever.
0000   153 ;               Remove code to insert default setting of default proxy
0000   154 ;               access executor parameter into the LNI database, leaving
0000   155 ;               the parameter to be defaulted when it is referenced.
0000   156 ;
0000   157 ;       V02-07  ADE0028         A.Eldridge      16-Feb-82
0000   158 ;               Added support for PIPELINE QUOTA
0000   159 ;
0000   160 ;       V02-06  ADE0027         A.Eldridge      10-Feb-82
0000   161 ;               Exit with error status from call to NET$CREATE_MBX.
0000   162 ;
0000   163 ;       V02-05  ADE0026         A.Eldridge      05-FEB-82
0000   164 ;               For the IRP used to deliver packets from the Transport to
0000   165 ;               the NSP layer, init the IRP$L_ASTPRM field to point to the
0000   166 ;               RCB.
0000   167 ;
0000   168 ;       V02-04  ADE0025         A.Eldridge      30-Nov-81
0000   169 ;               Added setting of default values for both the proxy login
0000   170 ;               and default access LNI fields.
0000   171 ;
```

```
0000   172 ;       V02-03  ADE0024         A.Eldridge      23-NOV-81
0000   173 ;               Cosmetic cleanup to comments etc.  No code changes.
0000   174 ;
0000   175 ;       V02-02                  A.Eldridge      23-SEP-81
0000   176 ;               Allocate the Cost/Hops matrix according to the number of
0000   177 ;               circuits specified in the database.
0000   178 ;
0000   179 ;       V02-01                  A.Eldridge      20-JUL-81
0000   180 ;               Remove all references to the DLI database
0000   181 ;
0000   182 ;       V02-00                  A.Eldridge      20-NOV-80
0000   183 ;               Added code to setup running UIC, process name, privileges,
0000   184 ;               directory.
0000   185 ;
0000   186 ;       V01-05                  A.Eldridge      20-MAR-80
0000   187 ;               Enhanced to include local node state transitions.  Renamed
0000   188 ;               module to NETACPTRN (formerly NETINI).  Re-implemented the
0000   189 ;               ACP "run-down".
0000   190 ;
0000   191 ;       V01-04  SGDX01           S.G.D.         17-Mar-1980
0000   192 ;               Get address of XMB update routine for connect delivery.
0000   193 ;               Reenable OPCOM call.
0000   194 ;
0000   195 ;       V01-03                  A.Eldridge      11-JUL-79
0000   196 ;               Extensive rewrite.  Added code so that the ACP mounts itself
0000   197 ;               by building its own I/O database control structures.  Added
0000   198 ;               support for the new configuration database.
0000   199 ;
0000   200 ;       V01-02                  S.G.D.          11-JUN-79
0000   201 ;               Modify for routing version.
0000   202 ;
0000   203 ;-
```

```
                        0000    205          .SBTTL  DECLARATIONS
                        0000    206 ;
                        0000    207 ; INCLUDE FILES:
                        0000    208 ;
                        0000    209          ;
                        0000    210          ;  System data structure definitions
                        0000    211          ;
                        0000    212          $AQBDEF                 ; ACP Queue Block
                        0000    213          $CCBDEF                 ; Channel Control Block
                        0000    214          $CRBDEF                 ; Controller Request Block
                        0000    215          $CXBDEF                 ; Complex Chained Buffer
                        0000    216          $DDBDEF                 ; Device Data Block
                        0000    217          $DEVDEF                 ; Device characteristics bits
                        0000    218          $DYNDEF                 ; Dynamic structure types
                        0000    219          $IPLDEF                 ; Interrupt Priority Levels
                        0000    220          $IRPDEF                 ; I/O Request Packet
                        0000    221          $JIBDEF                 ; Job Information Block
                        0000    222          $LOGDEF                 ; Logical name table definitions
                        0000    223          $MSGDEF                 ; Mailbox message definitions
                        0000    224          $NDBDEF                 ; Node Descriptor Block
                        0000    225          $NMADEF                 ; Network management definitions
                        0000    226          $PCBDEF                 ; Process Control Block
                        0000    227          $PHDDEF                 ; Process Header
                        0000    228          $TQEDEF                 ; Timer Queue Element
                        0000    229          $UCBDEF                 ; Unit Control Block
                        0000    230          $VECDEF                 ; Interrupt dispatch Vector
                        0000    231
                        0000    232          ;
                        0000    233          ;  Network Specific Definitions
                        0000    234          ;
                        0000    235          $CNRDEF                 ; Configuration Root block
                        0000    236          $CNFDEF                 ; Configuration data block
                        0000    237          $NFBDEF                 ; Network Function Block
                        0000    238          $LLIDEF                 ; Logical link information block
                        0000    239          $LNIDEF                 ; Local Node Information block
                        0000    240          $LPDDEF                 ; Logical Path Descriptor block
                        0000    241          $ADJDEF                 ; Adjacency block
                        0000    242          $LTBDEF                 ; Logical-link Table
                        0000    243          $NETSYMDEF              ; Some general network symbol definitions
                        0000    244          $NETUPDDEF              ; Describes ACP function calls to NETDRIVER
                        0000    245          $RCBDEF                 ; Routing Control Block (DECnet Volume Control
                        0000    246                                  ; block)
                        0000    247          $WQEDEF                 ; Work Queue Element
                        0000    248          $EVCDEF                 ; Event logging definitions
                        0000    249          $NSPMSGDEF              ; DNA message definitions
                        0000    250          ;
                        0000    251          ;   PSI specific definitions
                        0000    252          ;
                        0000    253          $PSIDEF                 ; PSI user definitions
                        0000    254
                        0000    255 ;
                        0000    256 ; EQUATED SYMBOLS:
                        0000    257 ;
                        0000    258                                  ; The routing tables are zero indexed
00000041                0000    259 NUM_LINES = NET$C_MAX_LINES + 1   ; line 0 is an "internal" path
00000400                0000    260 NUM_NODES = NET$C_MAX_NODES + 1   ; node 0 is the "local" node
                        0000    261
```

```
000186A0  0000    262 POOL_LENGTH = 100000                    ; no. of bytes in pool
          0000    263
```

```
0000    265 ;
0000    266 ;  MACROS
0000    267 ;
0000    268 .MACRO  $EVT  event, i,n,r,s,f,h        ; Create state table entries
0000    269                                         ; for the specified event
0000    270
0000    271         ACP$C_MAX_EVT = ACP$C_MAX_EVT + 1       ; Bump max event value
0000    272         ACP$C_'event' = ACP$C_MAX_EVT           ; Define line event symbol
0000    273
0000    274                 $ENT    i,_i                    ; Create table entry
0000    275                 $ENT    n,_n
0000    276                 $ENT    r,_r
0000    277                 $ENT    s,_s
0000    278                 $ENT    f,_f
0000    279                 $ENT    h,_h
0000    280
0000    281 .ENDM
0000    282
0000    283 .MACRO  $ENT    entry,def_sta                ; Create single state table
0000    284                                             ; entry
0000    285
0000    286                 acp$c_sta_. = acp$c_sta'def_sta'; Redefine default next state
0000    287         .BYTE   acp$c_sta_%EXTRACT(0,1,entry)   ; Setup next state
0000    288
0000    289                  $ent = %LENGTH(entry)-1
0000    290         .BYTE   %EXTRACT(1,_$ent,entry)         ; Setup action routine index
0000    291 .ENDM
```

```
0000   293 ;
0000   294 ; Overview of ACP state transitions
0000   295 ;
0000   296 ; NETACP is responsible for mounting and dismounting itself.  When it is
0000   297 ; initially started, NETACP builds its own non-paged pool I/O data structures
0000   298 ; (VCB, AQB, etc) and sets up a default configuration data base.   A  network
0000   299 ; control process issues Qios to the ACP to update the configuration data base
0000   300 ; and to change the state of the ACP.  When this process requests either the
0000   301 ; "off" or "shut" state, the ACP (eventually) deallocates its non-paged pool
0000   302 ; data structures and exits.  The dismount procedure is complicated by the
0000   303 ; fact that NETDRIVER also accesses the non-paged pool data structures.  Hence,
0000   304 ; NETDRIVER must rundown before the ACP can dismount.  RCB$W_MCOUNT and
0000   305 ; RCB$W_TRANS are used to co-ordinate NETDRIVER and NETACP rundown activities
0000   306 ; as outlined below.
0000   307 ;
0000   308 ; NETDRIVER is responsible for queuing all IRPs to the datalink drivers by
0000   309 ; using a special form of an IRP, sometimes referred to as an "internal IRP".
0000   310 ; One of the data links is the so called "local datalink" which is used when
0000   311 ; both ends of a logical link reside in the local node.  There is no actual
0000   312 ; hardware for this datalink and its "driver" is actually a small routine found
0000   313 ; at the end of NETDRIVER.
0000   314 ;
0000   315 ; When NETACP brings up a datalink, NETDRIVER is notified so that it can build
0000   316 ; a single IRP which is continually recycled to receive all messages over that
0000   317 ; particular datalink.  Each time the IRP is returned to NETDRIVER the device
0000   318 ; status bits stored in the IRP are checked.  If the device is still active
0000   319 ; then the IRP is requeued to the datalink driver, otherwise NETDRIVER signals
0000   320 ; NETACP by queuing the IRP to the AQB.   RCB$W_TRANS is incremented by
0000   321 ; NETDRIVER when it allocates the IRP; unmodifed when NETDRIVER queues the IRP
0000   322 ; to the AQB to signal that the datalink has gone inactive; and decremented by
0000   323 ; NETACP when it deallocates the IRP.
0000   324 ;
0000   325 ; NETDRIVER maintains a pool of IRPs to be used to transmit messages over the
0000   326 ; various datalinks.  NETACP uses field RCB$W_MAX_PKT to specify the number of
0000   327 ; IPRs in this pool.  NETDRIVER allocates and deallocates IRPs as needed to
0000   328 ; adjust to the specified pool size, and uses RCB$W_CUR_PKT to indicate the
0000   329 ; current size of the pool.  RCB$W_TRANS is incremented by NETDRIVER when it
0000   330 ; allocates an IRP and decremented when it deallocates one.
0000   331 ;
0000   332 ; NETACP allocates a timer queue element (TQE) upon initialization.  This TQE
0000   333 ; is used by NETDRIVER for timing out connects and for retransmitting unACKed
0000   334 ; messages.  NETDRIVER starts the TQE ticking when it allocates the receive IRP
0000   335 ; for the "local datalink". If RCB$W_MCOUNT = 0 when the timer fires, NETDRIVER
0000   336 ; senses that the ACP is dismounting.  It shuts off the TQE$V_REPEAT bit to
0000   337 ; stop the timer and signals the ACP that it is done by queuing to the AQB the
0000   338 ; "local datalink's" receive IRP.
0000   339 ;
0000   340 ;
0000   341 ;  RCB$W_MCOUNT
0000   342 ;
0000   343 ;    This field is used to keep track of the number of active logical links and
0000   344 ;    to allow NETACP to signal NETDRIVER that it wishes to dismount.  The UCB
0000   345 ;    DEV$V_DMT bit is not used since setting it would prevent all subsequent
0000   346 ;    logical link connects.  The goal is to always permit logical link connects
0000   347 ;    on behalf of users with the OPER privilege.
0000   348 ;
0000   349 ;    1.  Initialized by NETACP to 1 when the ACP mounts.
```

```
0000    350 ;   2.  Incremented by NETDRIVER for each logical link context block (XWB).
0000    351 ;   3.  Decremented by NETACP each time it deallocates an XWB.
0000    352 ;   4.  Decremented by NETACP when entering either the OFF or SHUT state.
0000    353 ;
0000    354 ;
0000    355 ;  RCB$W_TRANS
0000    356 ;
0000    357 ;   This field is used to keep track of all in-progress network activity, not
0000    358 ;   including active logical links.  When decremented to zero by NETACP it
0000    359 ;   serves a signal to dismount. (When RCB$W_TRANS becomes zero, RCB$W_MCOUNT
0000    360 ;   should also be at zero according to the current design.)
0000    361 ;
0000    362 ;   1.  Initialized by NETACP to 0 when the ACP mounts.
0000    363 ;   2.  Incremented by the EXEC whenever it queues anything to the AQB.
0000    364 ;   3.  Incremented by NETDRIVER for each message buffer it queues to the AQB.
0000    365 ;   4.  Incremented by NETDRIVER for each Transport IRP which it allocates.
0000    366 ;   5.  Decremented by NETDRIVER for each Transport IPR which it deallocates.
0000    367 ;   6.  Decremented by NETACP for each block it deques from its AQB.
0000    368 ;   7.  Unmodified when NETDRIVER queues the datalink receive IRP to the AQB
0000    369 ;       in order to signal datalink shutdown.  RCB$W_TRANS was incremented by
0000    370 ;       NETDRIVER when it allocated this IRP and will be decremented by NETACP
0000    371 ;       when it deallocates it.
0000    372 ;
0000    373 ;
0000    374 ;
0000    375
0000    376 ;
0000    377 ;  States:
0000    378 ;
0000    379 ;  The following states control the ACP transitions.
0000    380 ;
0000    381 ;$EQULST ACP$C_STA_,GLOBAL,0,1,<-
0000    382 ;
0000    383 ;       <I>     -; Initializing All connects are allowed
0000    384 ;       <N>     -; On           All connects are allowed
0000    385 ;       <R>     -; Restricted   Connect initiates only
0000    386 ;       <S>     -; Shut         Soft shutdown, no new links allowed.  Dismount
0000    387 ;                               when the last link disconnects.
0000    388 ;       <F>     -; Off          Hard shutdown, break all links, clear all data
0000    389 ;                               links, dismount.
0000    390 ;       <H>     -; Hibernate    The ACP is permanently hibernating to avoid a
0000    391 ;                               bugcheck.  A message is printed to reboot.
0000    392 ;                               (*** NOT YET IMPLEMENTED ***).
0000    393 ;>
```

```
                     0000   395 ;
                     0000   396 ; OWN STORAGE:
                     0000   397 ;
                00000000    398         .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                     0000   399
                FFFFFFFF    0000   400 ACP$C_MAX_EVT  = -1
                     0000   401 ACP$AW_STA_TAB::
                     0000   402 ;
                     0000   403 ;                    I    N    R    S    F    H
                     0000   404 ;                   ----------------------------
                     0000   405 $EVT  evt_nop        .    .    .    .    .    .
                     000C   406
                     000C   407 $EVT  opr_init       .    .8   .8   .8   .8   .7
                     0018   408 $EVT  opr_on         N    .    N    N1   N1   .7
                     0024   409 $EVT  opr_rstr       R    R    .    R1   R1   .7
                     0030   410 $EVT  opr_shut       S3   S3   S3   .5   .8   .7
                     003C   411 $EVT  opr_off        F2   F2   F2   F4   .4   .7
                     0048   412
                     0048   413 $EVT  lpd_loc        F2   F2   F2   F4   .4   .7
                     0054   414
                     0054   415 $EVT  evt_bug        H6   H6   H6   H6   H6   .7
                     0060   416
                     0060   417
                00000006    0060   418 ACP$C_STATES  = 6
                00000008    0060   419 ACP$C_EVENTS  = ACP$C_MAX_EVT+1
                     0060   420
                     0060   421 ACP$AL_ACTION::                    ; Action routine dispatch table
                     0060   422                                    ;
                00000333'    0060   423         .ADDRESS  ACT_NOP          ;  0    NOP
                0000020E'    0064   424         .ADDRESS  ACT_REVIVE       ;  1    If MOUNT neq 0  Then   inc MOUNT
                     0068   425                                    ;               Else   abort with error
                0000031A'    0068   426         .ADDRESS  ACT_NODE_OFF     ;  2    Dec MOUNT
                     006C   427                                    ;       Shut off all lines
                     006C   428                                    ;       Break all links
                00000313'    006C   429         .ADDRESS  ACT_NODE_SHUT    ;  3    Dec MOUNT
                0000031D'    0070   430         .ADDRESS  ACT_CLEANUP      ;  4    Shut off all lines
                     0074   431                                    ;       Break all links
                00000316'    0074   432         .ADDRESS  ACT_BRDCST       ;  5    Broadcast node is shutting down
                00000340'    0078   433         .ADDRESS  ACT_STALL        ;  6    Disable
                00000340'    007C   434         .ADDRESS  ACT_BUG          ;  7
                00000339'    0080   435         .ADDRESS  ACT_ERROR        ;  8    Return error
                     0084   436
                     0084   437
                     0084   438 OPR_EVT_MAP:                      ; Convert requested states to events
                     0084   439
                01 05    0084   440         .BYTE   ACP$C_OPR_OFF,  LNI$C_STA_OFF
                00 02    0086   441         .BYTE   ACP$C_OPR_ON,   LNI$C_STA_ON
                02 04    0088   442         .BYTE   ACP$C_OPR_SHUT, LNI$C_STA_SHUT
                03 03    008A   443         .BYTE   ACP$C_OPR_RSTR, LNI$C_STA_RSTR
                04 01    008C   444         .BYTE   ACP$C_OPR_INIT, LNI$C_STA_INIT
                00 07    008E   445         .BYTE   ACP$C_EVT_BUG,  0               ; Mark end of table
                     0090   446
                     0090   447
                     0090   448 EVL_STA_MAP:                      ; Convert internal ACP state to EVL state
                01 00    0090   449         .BYTE   ACP$C_STA_I, NMA$C_STATE_OFF
                00 01    0092   450         .BYTE   ACP$C_STA_N, NMA$C_STATE_ON
                03 02    0094   451         .BYTE   ACP$C_STA_R, NMA$C_STATE_RES
```

```
02 03  0096   452          .BYTE   ACP$C_STA_S, NMA$C_STATE_SHU
01 04  0098   453          .BYTE   ACP$C_STA_F, NMA$C_STATE_OFF
01 05  009A   454          .BYTE   ACP$C_STA_H, NMA$C_STATE_OFF
01 FF  009C   455          .BYTE   -1, NMA$C_STATE_OFF                 ; End of table; default state
```

```
                00000000   457          .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                    0000   458
00000000            0000   459 ACP_L_EVT:              .LONG 0          ; Value of state field
                    0004   460
00000006            0004   461 NET$GW_DLECHAN:        .BLKW 1          ; Channel number to DLE driver
0000000A            0006   462 NET$GL_DLE_UCB::       .BLKL 1          ; Address of UCB for DLE driver
0000000E            000A   463 NET$GL_DLE_UCB0::      .BLKL 1          ; Address of UCB0 for DLE driver
                    000E   464
00000002            000E   465 NET$GL_MY_POOL::       .LONG 2          ; IPL for pool synchronization
00000000            0012   466                        .LONG 0          ; Ptr to first free block
00000000            0016   467                        .LONG 0          ; Size of this block
                    001A   468
00000000            001A   469 SAVE_STA_TAB:          .LONG 0          ; Save the state table address
                    001E   470
00000026            001E   471 IOSB:                  .BLKL   2        ; I/O status block
                    0026   472
00000000            0026   473 CURRENT_SAD:           .LONG   0        ; Current PSI subaddress declaration
        0000        002A   474 PSI_DECL_CHAN:         .WORD   0        ; PSI declaration channel
                    002C   475
                    002C   476
                00000000   477          .PSECT  NET_LOCK_IMPURE,WRT,GBL,LONG
                    0000   478
                    0000   479
00000000            0000   480 NET$GL_LOC_LPD:        .LONG 0          ; Saved address of local LPD block
                    0004   481
00000054            0004   482 NEW_LNI_IMAGE:  .BLKB   LNI$C_LENGTH    ; Holds image of new local CNF (without
                    0054   483                                         ; storage for strings).  This is needed
                    0054   484                                         ; since this CNF is referrenced at high
                    0054   485                                         ; IPL and must therefore be locked in
                    0054   486                                         ; the working set.
                    0054   487
                0000009E   488          .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                    009E   489
   3A 54 45 4E 5F 000000A6'010E0000'    009E   490 NET_DESC:         .ASCID  '_NET:'          ; For assign channel to NET driver
      3A 44 4E 5F 000000B3'010E0000'    00AB   491 DLE_DESC:         .ASCID  '_ND:'           ; For assign channel to DLE driver
                     54 45 4E 00'       00B7   492 NET_NAME:         .ASCIC  'NET'            ; For DDB search
                              03        00B7
4F 4E 24 53 59 53 000000C3'010E0000'    00BB   493 NODE_DESC:        .ASCID  'SYS$NODE'       ; For deleting local node's logical
                           45 44        00C9
                                        00CB   494                                           ; name on shutdown
                                        00CB   495
                                        00CB   496
                        FFFFFFFF        00CB   497 PRIVMSK:          .LONG   -1               ; For setting up running privileges
                        FFFFFFFF        00CF   498                   .LONG   -1               ;! *** IS THIS OVERKILL ? ***
                                        00D3   499
                        00010004        00D3   500 UIC               =       ^O1@16+^O4       ; For setting NETACP's UIC
49 44 24 53 59 53 000000DB'010E0000'    00D3   501 SYSDISK:          .ASCID  'SYS$DISK'       ; Default disk logical name
                              4B 53     00E1
59 53 24 53 59 53 000000EB'010E0000'    00E3   502 SYSTEM_ROOT:     .ASCID  'SYS$SYSROOT:'   ; Default disk equivalence name
                  3A 54 4F 4F 52 53     00F1
47 4D 53 59 53 5B 000000FF'010E0000'    00F7   503 SYSMGR:          .ASCID  '[SYSMGR]'       ; Default directory
                              5D 52     0105
                  54 45 4E 43 45 44     0107   504 USERNAME:        .ASCII  'DECNET'         ; Username for NETACP process
                        00000006        010D   505 USERNAMLEN = .-USERNAME
                                        010D   506
                                        010D   507
                        00000000'       010D   508 LKWSET_ADDR:    .ADDRESS BEGLCK          ; Descriptor for locking stuff running
```

```
                                                B 16
00000000' 0111   509                .ADDRESS ENDLCK          ; at high IPL into working set
          0115   510
          0115   511
 00000000 0000   512          .PSECT  NET_LOCK_A,GBL,NOWRT,LONG ; PSECTs are linked alpabetically - all
          0000   513                                          ; locked regions should use a PSECT
          0000   514 BEGLCK:                                  ; begining with "NET_LOCK"
          0000   515
 00000000 0000   516          .PSECT  NET_LOCK_Z,GBL,WRT      ; Mark end of locked region
          0000   517                                          ; The WRT attribute is to ensure it is
          0000   518                                          ; the last NET_LOCKxxx .psect in the
          0000   519                                          ; linker's collating sequence
          0000   520 ENDLCK:
```

```
                                0000    522          .SBTTL  INITIALIZATION
                                0000    523 ;++
                                0000    524 ; NET$INITIALIZE                        ; ACP entry point
                                0000    525 ;
                                0000    526 ; This is the ACP transfer address.  All the control sturctures required
                                0000    527 ; to mount device NET are allocated, initialized, and attached to the NET
                                0000    528 ; UCB(s).  A channel to NET is created in order to allow NETDRIVER to send
                                0000    529 ; mailbox messages to the ACP.  The configuration database is initialized.
                                0000    530 ; Control is then passed to the ACP work queue processing module.
                                0000    531 ;
                                0000    532 ;--
                           00000000     533          .PSECT  NET_CODE,NOWRT,EXE
                                0000    534
                                0000    535 NET$INITIALIZE::
                           0000 0000    536          .WORD   0                      ; ACP entry point
                                0002    537
                                0002    538          $LKWSET_S LKWSET_ADDR          ; Lock I/O data base code into
                                0011    539                                         ; working set
                        2C 50   E9 0011 540          BLBC    R0,90$                 ; Exit if error detected
                                0014    541          ;
                                0014    542          ;   Set the default disk to SYS$SYSROOT: and the default directory to
                                0014    543          ;   [SYSMGR]
                                0014    544          ;
                                0014    545          $CRELOG_S LOGNAM=SYSDISK,-     ; Set default disk
                                0014    546                    EQLNAM=SYSTEM_ROOT,- ;
                                0014    547                    TBLFLG=#LOG$C_PROCESS ;
                             7E 7C  0027 548          CLRQ    -(SP)                  ; No arguments 2 and 3
                     00F7'CF    9F  0029 549          PUSHAB  SYSMGR                 ; Address of new default directory
           00000000'GF    03    FB 002D 550          CALLS   #3,G^SYS$SETDDIR       ; Set default directory
                                0034    551          ;
                                0034    552          ;   The remainder of the ACP executes in kernel mode
                                0034    553          ;
                                0034    554          $CMKRNL_S B^STARTUP            ; Go to kernel mode forever
                                  04 0040 555 90$:   RET                            ; Come here upon error during startup
                                0041    556
                                0041    557
                                0041    558 STARTUP:
                           0000 0041    559          .WORD   0                      ; entry point
                                0043    560
                                0043    561          ;
                                0043    562          ;   Set the UIC = [1,4]
                                0043    563          ;
                     55 00000000'GF  D0 0043 564          MOVL    G^CTL$GL_PCB,R5        ; Get current PCB address
                00BC C5 00010004 8F  D0 004A 565          MOVL    #UIC,PCB$L_UIC(R5)     ; Set permanent UIC
                                0053    566          ;
                                0053    567          ;   Enable all privileges
                                0053    568          ;
                                0053    569          ASSUME  PHD$Q_PRIVMSK EQ 0
                     6C B5   00CB'CF  7D 0053 570          MOVQ    PRIVMSK,@PCB$L_PHD(R5) ; Setup running privileges
                                0059    571          ;
                                0059    572          ;   Set the user name
                                0059    573          ;
                     56   0080 C5  D0 0059 574          MOVL    PCB$L_JIB(R5),R6       ; Get JIB address
OC A6 0C   20   0107'CF   06  2C 005E 575          MOVC5   #USERNAMLEN,USERNAME,#^A' ',#12,JIB$T_USERNAME(R6)
      0C   20   0107'CF   06  2C 0067 576          MOVC5   #USERNAMLEN,USERNAME,#^A' ',#12,G^CTL$T_USERNAME
                 00000000'GF      006E
                                0073    577          ;
```

```
                     0073   578          ;   Assign a channel to DLE driver, to make sure it is present
                     0073   579          ;
                     0073   580          $ASSIGN_S DEVNAM = DLE_DESC,-   ; Assign channel to DLE driver
                     0073   581                    CHAN = NET$GW_DLECHAN   ; Store channel number
           3B 50     E9  0084   582          BLBC    R0,10$                  ; Exit if error detected
50   0004'CF        3C  0087   583          MOVZWL  NET$GW_DLECHAN,R0       ; Get channel to DLE driver
 00000000'GF        16  008C   584          JSB     G^IOC$VERIFYCHAN        ; CCB -> R1, Status -> R0,
                     0092   585                                          ; R2,R3 are garbaged
                     0092   586                                          ; NO need to check return status since
                     0092   587                                          ; the info is always returned
      55   61       D0  0092   588          MOVL    CCB$L_UCB(R1),R5        ; Get DLE's UCB address
0006'CF    55       D0  0095   589          MOVL    R5,NET$GL_DLE_UCB       ; Save the UCB address
  50   28 A5       D0  009A   590          MOVL    UCB$L_DDB(R5),R0        ; Get the DDB
       04 A0       D0  009E   591          MOVL    DDB$L_UCB(R0),-         ; Save the DLE UCB0 address
     000A'CF               00A1   592                  NET$GL_DLE_UCB0
                     00A4   593          ;
                     00A4   594          ;   Set up mailbox for process termination and received connect
                     00A4   595          ;   notifications.  Assign a channel to NETDRIVER with this associated
                     00A4   596          ;   mailbox.
                     00A4   597          ;
         FF59'       30  00A4   598          BSBW    NET$CREATE_MBX          ; Create the mailbox
        18 50        E9  00A7   599          BLBC    R0,10$                  ; If LBC then error
                     00AA   600          $ASSIGN_S -
                     00AA   601                  DEVNAM = NET_DESC,-     ; "_NET:" refers to control path
                     00AA   602                  CHAN   = NET$GW_NETCHAN,-   ; Store channel #
                     00AA   603                  MBXNAM = NET$GQ_MBX_NAME    ; Specify associated mailbox
        03 50        E8  00BF   604          BLBS    R0,20$                  ; If lbc error
          0080       31  00C2   605 10$:     BRW     100$                    ; Error
50   0000'CF        3C  00C5   606 20$:     MOVZWL  NET$GW_NETCHAN,R0       ; Get channel to "_NET:"
 00000000'GF        16  00CA   607          JSB     G^IOC$VERIFYCHAN        ; CCB -> R1, Status -> R0,
                     00D0   608                                          ; R2,R3 are garbaged
                     00D0   609                                          ; NO need to check return status since
                     00D0   610                                          ; the info is always returned
      55   61       D0  00D0   611          MOVL    CCB$L_UCB(R1),R5        ; Get _NET:'s UCB address
0000'CF    55       D0  00D3   612          MOVL    R5,NET$GL_NET_UCB       ; Save the UCB address
  50   28 A5       D0  00D8   613          MOVL    UCB$L_DDB(R5),R0        ; Get the DDB
       04 A0       D0  00DC   614          MOVL    DDB$L_UCB(R0),-         ; Save the NET0 UCB address
     0000'CF               00DF   615                  NET$GL_PTR_UCB0
 00000000'GF       7D  00E2   616          MOVQ    G^SYS$GQ_VERSION,-      ; Stuff system version into
     0000'CF               00E8   617                  NET$GQ_VERSION          ; default executor ident string
                     00EB   618          ;
                     00EB   619          ;   Create a pool for internally generated messages and control blocks.
                     00EB   620          ;
56   0012'CF        9E  00EB   621          MOVAB   NET$GL_MY_POOL+4,R6     ; Point to pool head
                     00F0   622          $EXPREG_S -
                     00F0   623                  PAGCNT = #<POOL_LENGTH+511>/512,- ; no. of pages
                     00F0   624                  RETADR = (R6)+              ; Address for pool address
         3F 50       E9  0103   625          BLBC    R0,100$                 ; If LBC then error
            76       D4  0106   626          CLRL    -(R6)                   ; Set up pool head
      56    76       D0  0108   627          MOVL    -(R6),R6                ; Get address of pool
            86       D4  010B   628          CLRL    (R6)+                   ; Init 1st pool block
66   000186A0 8F       D0  010D   629          MOVL    #POOL_LENGTH,(R6)       ; It's this long
                     0114   630          ;
                     0114   631          ;   Set up control blocks
                     0114   632          ;
          FEE9'      30  0114   633          BSBW    MOUNT                   ; Setup VCB,AQB, init UCB's
        2B 50        E9  0117   634          BLBC    R0,100$                 ;
```

```
                FEE3'  30  011A   635          BSBW    NET$INIT_ROUTING        ; Initialize routing database
              1D 50    E9  011D   636          BLBC    R0,90$                  ; Br on error
00000000'GF     00     FB  0120   637          CALLS   #0,G^NET$INI_CONFIG     ; Initialize CNR/CNF database
              13 50    E9  0127   638          BLBC    R0,90$                  ; Br on error
                         012A   639          ;
                         012A   640          ;   Initialize NETDRIVER
                         012A   641          ;
               0019      30  012A   642          BSBW    INIT_NETDRIVER          ; Initialize NETDRIVER
             0D 50       E9  012D   643          BLBC    R0,90$                  ; Branch if error detected
                         0130   644          ;
                         0130   645          ;   Announce we are up
                         0130   646          ;
            50   01       90  0130   647          MOVB    #NDB$C_MSG_START,R0     ; DECnet starting
            0000'CF      16  0133   648          JSB     NET$OPCOM               ; Print it
                         0137   649          ;
                         0137   650          ;   Activate mailbox and ACP queue
                         0137   651          ;
              FEC6'      30  0137   652          BSBW    NET$MBX_QIO             ; Post the read to mailbox
              FEC3'      31  013A   653          BRW     NET$DISPATCH            ; Process all the work queues
                         013D   654                                              ; - will hibernate
                         013D   655
                         013D   656  ;
                         013D   657  ; Come here on errors
                         013D   658  ;
                         013D   659
                50       DD  013D   660  90$:    PUSHL   R0                      ; Save final status
              FFEE'      30  013F   661          BSBW    DISMOUNT                ; Dismount - remove control blocks
                50 8ED0  0142   662          POPL    R0                      ; Restore final status
                         04  0145   663  100$:   RET                             ; Come here on error during startup
```

```
                                0146   665          .SBTTL  MOUNT - Mount the NET device
                                0146   666 ;+
                                0146   667 ; MOUNT - Load/Init I/O data base control blocks
                                0146   668 ;
                                0146   669 ; This routine is called to create the I/O database control blocks
                                0146   670 ; necessary to allow communication with the ACP.
                                0146   671 ;
                                0146   672 ; Inputs:
                                0146   673 ;
                                0146   674 ;       None
                                0146   675 ;
                                0146   676 ; Outputs:
                                0146   677 ;
                                0146   678 ;       None
                                0146   679 ;-
                                0146   680          .SAVE_PSECT
                            00000000   681          .PSECT  NET_LOCK_CODE,NOWRT,GBL
                                0000   682
01B0'CF    00    FB             0000   683 MOUNT:   CALLS   #0,LOCK_IODB              ; Lock the io database
           0B    10             0005   684          BSBB    10$                      ; Mount the device
           50    DD             0007   685          PUSHL   R0                       ; Save status
01C7'CF    00    FB             0009   686          CALLS   #0,UNLOCK_IODB           ; Unlock database
           50  8ED0             000E   687          POPL    R0                       ; Recover status
               05               0011   688          RSB
                                0012   689
                                0012   690          ;
                                0012   691          ;   Check to see if ACP is already mounted somewhere else
                                0012   692          ;   in the system.  Check to see if ACP is marked for dismount.
                                0012   693          ;
53    0000'CF    D0             0012   694 10$:     MOVL    NET$GL_PTR_UCB0,R3       ; Get the NET0 UCB address
              13  E0             0017   695          BBS     #DEV$V_MNT,-             ; Exit if NETACP already mounted
          05 38 A3              0019   696                  UCB$L_DEVCHAR(R3),20$    ; elsewhere in the system
              15  E1             001C   697          BBC     S^#DEV$V_DMT,-           ; Br unless device marked for
          06 38 A3              001E   698                  UCB$L_DEVCHAR(R3),40$    ; dismount
50    0000'8F    3C             0021   699 20$:     MOVZWL  #SS$_DEVMOUNT,R0         ; Set return code
               05               0026   700          RSB
                                0027   701
                                0027   702 40$:     ;
                                0027   703          ;  Allocate the ACP queue block (AQB), Volume Control Block (VCB),
                                0027   704          ;  LPD for primary ECL (NSP), an IRP for local LPD, and a Timer
                                0027   705          ;  Queue Element (TQE) for NETDRIVER's clock (eventually, allocation
                                0027   706          ;  of the TQE should be moved to NETDRIVER).
                                0027   707          ;
                                0027   708
                    0000000F    0027   709          XMSK = ^X<F>                     ; Used for quadword alignment
                                0027   710
                    00000020    0027   711 AQB$C_XLNG = <AQB$C_LENGTH+XMSK> & ^C<XMSK>     ; Define rounted lth for alloc
                    000000B0    0027   712 RCB$C_XLNG = <RCB$C_LENGTH+XMSK> & ^C<XMSK>     ; Define rounded lth for alloc
                    00000070    0027   713 LPD$C_XLNG = <LPD$C_LENGTH+XMSK> & ^C<XMSK>     ; Define rounded lth for alloc
                    00000030    0027   714 TQE$C_XLNG = <TQE$C_LENGTH+XMSK> & ^C<XMSK>     ; Define rounded lth for alloc
                                0027   715
                                0027   716 LNG =    AQB$C_XLNG + -                   ; Compute size of all blocks
                                0027   717          RCB$C_XLNG + -
                                0027   718          LPD$C_XLNG + -
                    00000170    0027   719          TQE$C_XLNG
                                0027   720
 0000'CF    D4                  0027   721          CLRL    NET$GL_PTR_AQB           ; Flag 'no AQB'
```

```
                0000'CF    D4   002B   722          CLRL    NET$GL_PTR_VCB                  ; Flag 'no VCB'
                0000'CF    D4   002F   723          CLRL    NET$GL_LOC_LPD                  ; Flag 'no local LPD'
             51 0170 8F    3C   0033   724          MOVZWL  #LNG,R1                         ; Get length of block
                   FFC5'   30   0038   725          BSBW    NET$ALONPGD_Z                   ; Get and zero the block
                  01 50    E8   003B   726          BLBS    R0,INIT_AQB                     ; Br if successful
                           05   003E   727          RSB                                     ; Else, exit with status
                                003F   728
                                003F   729  INIT_AQB:                                      ; Initialize the AQB
          0000'CF    52     D0   003F   730          MOVL    R2,NET$GL_PTR_AQB               ; Save AQB address
                62   52     D0   0044   731          MOVL    R2,AQB$L_ACPQFL(R2)             ; Set queue forward link
             04 A2   52     D0   0047   732          MOVL    R2,AQB$L_ACPQBL(R2)             ; Set queue back link
             0B A2   01     90   004B   733          MOVB    #1,AQB$B_MNTCNT(R2)             ; Initialize mount count
       50 00000000'GF       D0   004F   734          MOVL    G^CTL$GL_PCB,R0                 ; Get current PCB
          0C A2    60 A0     D0   0056   735          MOVL    PCB$L_PID(R0),AQB$L_ACPPID(R2)  ; Store ACP's PID
             0A A2   03     90   005B   736          MOVB    #DYN$C_AQB,   AQB$B_TYPE(R2)    ; Note type of block
             08 A2   1C     9B   005F   737          MOVZBW  #AQB$C_LENGTH,AQB$W_SIZE(R2)    ; Record size of AQB
             15 A2   04     90   0063   738          MOVB    #AQB$K_NET,   AQB$B_ACPTYPE(R2) ; Flag ACP type
             14 A2   01     88   0067   739          BISB    #AQB$M_UNIQUE,AQB$B_STATUS(R2)  ; Note status of ACP
             52   20 A2     9E   006B   740          MOVAB   AQB$C_XLNG(R2),R2               ; Point to rest of storage
                            006F   741
                            006F   742  INIT_RCB:                                      ; Initialize the RCB
          0000'CF    52     D0   006F   743          MOVL    R2,NET$GL_PTR_VCB               ; Store VCB address
             54 A2   01     B0   0074   744          MOVW    #1,                RCB$W_MCOUNT(R2) ; Init mount count
             0A A2   11     90   0078   745          MOVB    #DYN$C_VCB,        RCB$B_TYPE(R2)   ; Init block type
          10 A2   0000'CF   D0   007C   746          MOVL    NET$GL_PTR_AQB, RCB$L_AQB(R2)   ; Link to AQB
          14 A2   0000'CF   D0   0082   747          MOVL    NET$GL_NET_UCB, RCB$L_ACP_UCB(R2) ; Set pointer to the ACP's
    0090 C2   00000000'GF   D0   0088   748          MOVL    G^EXE$GL_ABSTIM,RCB$L_ABS_TIM(R2) ; Reset time last zeroed
                            0091   749          ;
                            0091   750          ;   Initialize queue headers
                            0091   751          ;
                            0091   752          ASSUME  RCB$Q_IRP_FREE EQ 0
                            0091   753          ASSUME  RCB$Q_LOC_XMT  EQ RCB$Q_LOC_RCV+8
                            0091   754          ASSUME  RCB$Q_IRP_WAIT EQ RCB$Q_LOC_XMT+8
                            0091   755
             62   52     D0   0091   756          MOVL    R2,(R2)                         ; Free IRP listhead
          04 A2   52     DC   0094   757          MOVL    R2,4(R2)                        ;
          50   3C A2     9E   0098   758          MOVAB   RCB$Q_LOC_RCV(R2),R0            ;
             60   50     D0   009C   759          MOVL    R0,(R0)                         ; Local-datalink rcv listhead
             80   80     DE   009F   760          MOVAL   (R0)+,(R0)+                     ;
             60   50     D0   00A2   761          MOVL    R0,(R0)                         ; Local-datalink xmt listhead
             80   80     DE   00A5   762          MOVAL   (R0)+,(R0)+                     ;
             60   50     D0   00A8   763          MOVL    R0,(R0)                         ; Wait for IRP listhead
             80   80     DE   00AB   764          MOVAL   (R0)+,(R0)+                     ;
                            00AE   765
          50 00A0 C2     9E   00AE   766          MOVAB   RCB$Q_CXB_FREE(R2),R0
             60   50     D0   00B3   767          MOVL    R0,(R0)                         ; Free CXB listhead
             80   80     DE   00B6   768          MOVAL   (R0)+,(R0)+                     ;
          53 00B0 C2     9E   00B9   769          MOVAB   RCB$C_XLNG(R2),R3               ; Point to "local datalink"
                            00BE   770
                            00BE   771  INIT_LPD:
          0000'CF    53     D0   00BE   772          MOVL    R3,NET$GL_LOC_LPD               ; Save address of local LPD
                            00C3   773
                            00C3   774          ASSUME  LPD$Q_REQ_WAIT EQ 0
                            00C3   775
             63   53     D0   00C3   776          MOVL    R3,(R3)                         ; Init wait queue listhead
          04 A3   53     D0   00C6   777          MOVL    R3,4(R3)                        ;
          1E A3   01     90   00CA   778          MOVB    #1,LPD$B_XMT_SRL(R3)            ; Init square root limiter
```

```
                1F A3   01    90   00CE   779          MOVB    #1,LPD$B_XMT_IPL(R3)            ; Init input packet limiter
                20 A3   01    90   00D2   780          MOVB    #LPD$C_LOC_INX,LPD$B_PTH_INX(R3) ; Init path i.d.
                                   00D6   781          ;
                                   00D6   782          ;   Initialize status flags - don't set ACTIVE flag, since NETDRIVER
                                   00D6   783          ;   "owns" the flag for it's own purposes.
                                   00D6   784          ;
                               B0  00D6   785          MOVW    #LPD$M_XBF!-                    ; Reads and writes are Buffered
                                   00D7   786                    LPD$M_RBF!-
                                   00D7   787                    LPD$M_RUN,-                     ; Mark it up
             22 A3    0070 8F       00D7   788                    LPD$W_STS(R3)
                53      70 A3   9E  00DC   789          MOVAB   LPD$C_XLNG(R3),R3               ; Point to next block
                                   00E0   790
                                   00E0   791 INIT_TQE:                                        ; Init timer queue element
                30 A2   53      D0  00E0   792          MOVL    R3,RCB$L_PTR_TQE(R2)            ; Store TQE address
                0A A3   0F      90  00E4   793          MOVB    #DYN$C_TQE,TQE$B_TYPE(R3)       ; Setup structure type
                                   00E8   794                                                  ; Let TQE$W_SIZE stay at zero
                                   00E8   795                                                  ; to trap deallocation bugs
                                   00E8   796          ;
                                   00E8   797          ;   Link data structures into I/O data base
                                   00E8   798          ;
              53    0000'CF   D0  00E8   799          MOVL    NET$GL_PTR_UCB0,R3              ; Get NET UCB0 pointer
              51    000A'CF   D0  00ED   800          MOVL    NET$GL_DLE_UCB0,R1              ; Get DLE UCB0 pointer
                                   00F2   801          ASSUME  IPL$_SYNCH  LE  NET$C_IPL
                                   00F2   802          DSBINT  #NET$C_IPL                      ; Synch I/O data base
                                   00F8   803          ;
                                   00F8   804          ;   Make all DLE UCBs mounted, by storing the ACP
                                   00F8   805          ;   VCB address, and setting the MNT bit.
                                   00F8   806          ;
                34 A1   52      D0  00F8   807 40$:     MOVL    R2,UCB$L_VCB(R1)                ; Link DLE driver to VCB
                                   00FC   808          SETBIT  #DEV$V_MNT,UCB$L_DEVCHAR(R1)    ; DLE device is mounted
                51    30 A1     D0  0101   809          MOVL    UCB$L_LINK(R1),R1               ; Get next UCB
                          F1     12  0105   810          BNEQ    40$                             ; If EQL then done
                                   0107   811          ;
                                   0107   812          ;   Make all NET UCBs mounted, by storing the ACP
                                   0107   813          ;   VCB address, and setting the MNT bit.
                                   0107   814          ;
                34 A3   52      D0  0107   815 50$:     MOVL    R2,UCB$L_VCB(R3)                ; Link NET driver to VCB
                                   010B   816          SETBIT  #DEV$V_MNT,UCB$L_DEVCHAR(R3)    ; Device is mounted
                53    30 A3     D0  0110   817          MOVL    UCB$L_LINK(R3),R3               ; Get next UCB
                          F1     12  0114   818          BNEQ    50$                             ; If EQL then done
                                   0116   819          ;
                                   0116   820          ;   Add our AQB to the system-wide AQB list
                                   0116   821          ;
            52    0000'CF   D0  0116   822          MOVL    NET$GL_PTR_AQB,R2               ; Get AQB
51  00000000'GF   9E  011B   823          MOVAB   G^IOC$GL_AQBLIST,R1             ; Get ptr to list head
              10 A2   61      D0  0122   824          MOVL    (R1),AQB$L_LINK(R2)             ; Link list to AQB
                 61   52      D0  0126   825          MOVL    R2,(R1)                         ; Make this AQB the 1st
                                   0129   826          ENBINT                                  ; Restore IPL
                 50   01      D0  012C   827          MOVL    #1,R0                           ; Return success
                                   012F   828          RSB
                                   0130   829
                                00000146  830          .RESTORE_PSECT
```

```
                          0146   832          .SBTTL  INIT_NETDRIVER - Tell NETDRIVER to initialize
                          0146   833 ;++
                          0146   834 ; INIT_NETDRIVER - Tell NETDRIVER to initialize itself
                          0146   835 ;
                          0146   836 ; This routine is called when all nonpaged data structures have
                          0146   837 ; been setup.
                          0146   838 ;
                          0146   839 ; Inputs:
                          0146   840 ;
                          0146   841 ;       None
                          0146   842 ;
                          0146   843 ; Outputs:
                          0146   844 ;
                          0146   845 ;       None
                          0146   846 ;-
                          0146   847 INIT_NETDRIVER:
                          0146   848          ;
                          0146   849          ;   Tell NETDRIVER we've mounted the ACP
                          0146   850          ;
55   0000'CF    D0    0146   851          MOVL    NET$GL_NET_UCB,R5               ; Get the UCB address
  52     34 A5    D0    014B   852          MOVL    UCB$L_VCB(R5),R2                ; Get RCB address
       51    01    D0    014F   853          MOVL    #LPD$C_LOC_INX,R1               ; Get local LPD index
51     28 B241    D0    0152   854          MOVL    @RCB$L_PTR_LPD(R2)[R1],R1       ; Get local LPD address
       50    05    D0    0157   855          MOVL    S^#NETUPD$_DLL_ON,R0            ; Set function code
             01    10    015A   856          BSBB    CALL_NETDRIVER                  ; Tell NETDRIVER about LPD
                         05    015C   857          RSB
```

```
                          015D   859          .SBTTL  CALL_NETDRIVER - Call NETDRIVER entry point
                          015D   860 ;+
                          015D   861 ; CALL_NETDRIVER - Call NETDRIVER entry point
                          015D   862 ;
                          015D   863 ; This routine is called to call NETDRIVER's entry point.
                          015D   864 ;
                          015D   865 ; Inputs:
                          015D   866 ;
                          015D   867 ;       R0-R5 are setup according to function code in R0.
                          015D   868 ;
                          015D   869 ; Outputs:
                          015D   870 ;
                          015D   871 ;       R0 = status
                          015D   872 ;-
                          015D   873 CALL_NETDRIVER::
            51    DD      015D   874          PUSHL   R1                              ; Save R1
51  0000'CF     D0      015F   875          MOVL    NET$GL_PTR_UCB0,R1              ; Get UCB0 address
  51    24 A1   D0      0164   876          MOVL    UCB$L_CRB(R1),R1                ; Get the CRB address
  51    40 A1   DD      0168   877          PUSHL   CRB$L_INTD+VEC$L_START(R1) ; Push NETDRIVER entry address
  51    04 AE   D0      016B   878          MOVL    4(SP),R1                        ; Restore R1
        9E      16      016F   879          JSB     @(SP)+                          ; Call NETDRIVER entry point
     5E   04    C0      0171   880          ADDL    #4,SP                           ; Pop saved value of R1
                05      0174   881          RSB                                     ; Return to caller
```

```
                                0175   883          .SBTTL  NET$DEC_TRANS - Decrement transaction count
                                0175   884 ;++
                                0175   885 ; NET$DEC_TRANS - Decrement the transaction count
                                0175   886 ;
                                0175   887 ; Decrement the transaction count in the RCB (really a Volume Control Block)
                                0175   888 ; and dismount if it goes to zero.
                                0175   889 ;
                                0175   890 ; Inputs:
                                0175   891 ;
                                0175   892 ;       None
                                0175   893 ;
                                0175   894 ; Outputs:
                                0175   895 ;
                                0175   896 ;       None
                                0175   897 ;
                                0175   898 ;       R0 is destroyed; all other registers are preserved.
                                0175   899 ;
                                0175   900 ; RCB and related structures are deallocated if the device is dismounted.
                                0175   901 ;--
                                0175   902 NET$DEC_TRANS::                                ;
                          3E    BB  0175   903          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save R4
                50  0000'CF     D0  0177   904          MOVL    NET$GL_PTR_UCB0,R0      ; Address the NET0 UCB
                  50    34 A0   D0  017C   905          MOVL    UCB$L_VCB(R0),R0        ; Get the RCB address
                        0C A0   B7  0180   906          DECW    RCB$W_TRANS(R0)         ; Decrement the Transaction count
                        09      12  0183   907          BNEQ    20$                     ; If NEQ then don't dismount
                        54 A0   B5  0185   908          TSTW    RCB$W_MCOUNT(R0)        ; Is mount count zero ?
                        07      13  0188   909          BEQL    50$                     ; If so, time to go away
                                    018A   910          BUG_CHECK NETNOSTATE,FATAL      ; Else there's a bug
                                    018E   911
                          3E    BA  018E   912 20$:     POPR    #^M<R1,R2,R3,R4,R5>     ; Restore regs
                                05  0190   913          RSB
                                    0191   914
                                    0191   915 ;
                                    0191   916 ; The reference count has gone to zero.  Dismount the ACP and go away.
                                    0191   917 ;
                                    0191   918
                             FF9C'  30  0191   919 50$:     BSBW    DISMOUNT                ; Dismount the ACP
                0000'CF      D5  0194   920          TSTL    NET$GL_PTR_AQB          ; Is the AQB still being referenced?
                             F4  12  0198   921          BNEQ    20$                     ; If so, stick around (this is a bug)
                50    02     90  019A   922          MOVB    #NDB$C_MSG_SHUT,R0      ; Decnet shutting down
                0000'CF      16  019D   923          JSB     NET$OPCOM               ; Print it
                                    01A1   924          $DELPRC_S                       ; Go away safely
                                    01AC   925          BUG_CHECK NETSYSSRV,FATAL       ; Should never get here
```

```
                                01B0   927          .SBTTL  DISMOUNT - Dismount the NET device
                                01B0   928 ;+
                                01B0   929 ; DISMOUNT - Dismount the NET device
                                01B0   930 ;
                                01B0   931 ; This routine is called to cleanup the I/O database, before going away.
                                01B0   932 ;
                                01B0   933 ; Inputs:
                                01B0   934 ;
                                01B0   935 ;       None
                                01B0   936 ;
                                01B0   937 ; Outputs:
                                01B0   938 ;
                                01B0   939 ;       None
                                01B0   940 ;-
                                01B0   941          .SAVE_PSECT
                            00000130   942          .PSECT  NET_LOCK_CODE,NOWRT,GBL
                                0130   943
                                0130   944 DISMOUNT:
01B0'CF    00    FB             0130   945          CALLS   #0,LOCK_IODB            ; Lock the i/o data base
                                0135   946          ASSUME  IPL$_SYNCH LE NET$C_IPL
                                0135   947          DSBINT  #NET$C_IPL              ; Sync with I/O data base and NETDRIVER
                   3C   10      013B   948          BSBB    100$                    ; Dismount the NET device
                                013D   949          ENBINT                          ; Restore IPL
01C7'CF    00        FB          0140   950          CALLS   #0,UNLOCK_IODB          ; Unlock the data base
              FEB8'  30          0145   951          BSBW    NET$KILL_MBX            ; Delete the mailbox
                                0148   952          $DASSGN_S CHAN = NET$GW_NETCHAN ; Deassign the network channel
                                0154   953          $DASSGN_S CHAN = NET$GW_DLECHAN ; Deaasign channel to DLE driver
                                0160   954          $CANTIM_S ACMODE = #0,-         ; Cancel all timers (an Exec fatal bug
                                0160   955                    REQIDT = #0           ;  could occur on rundown otherwise)
                                0169   956          $DELLOG_S  LOGNAM = NODE_DESC   ; Deassign SYS$NODE
                   05           0178   957          RSB
                                0179   958
55    0000'CF    D0             0179   959 100$:   MOVL    NET$GL_PTR_UCB0,R5      ; Address the NET UCB0
51    000A'CF    D0             017E   960          MOVL    NET$GL_DLE_UCB0,R1      ; Address the DLE UCB0
  54    34 A5    D0             0183   961          MOVL    UCB$L_VCB(R5),R4        ; Get RCB address
             01    DD            0187   962          PUSHL   #1                      ; Mark end of deallocation list
                                0189   963          ;
                                0189   964          ;  Mark all DLE UCB's as dismounted, by clearing the VCB
                                0189   965          ;  pointer in each one.
                                0189   966          ;
00280000  8F    CA              0189   967 105$:   BICL    #DEV$M_MNT!DEV$M_DMT,-  ;
        38 A1                   018F   968                  UCB$L_DEVCHAR(R1)       ; Device is dismounted
        34 A1    D4             0191   969          CLRL    UCB$L_VCB(R1)           ; No more VCB
  51    30 A1    D0             0194   970          MOVL    UCB$L_LINK(R1),R1       ; Get next UCB address
             EF    12            0198   971          BNEQ    105$                    ; If NEQ more to go
                                019A   972          ;
                                019A   973          ;  Mark all NET UCB's as dismounted, by clearing the VCB
                                019A   974          ;  pointer in each one.
                                019A   975          ;
00280000  8F    CA              019A   976 110$:   BICL    #DEV$M_MNT!DEV$M_DMT,-  ;
        38 A5                   01A0   977                  UCB$L_DEVCHAR(R5)       ; Device is dismounted
        34 A5    D4             01A2   978          CLRL    UCB$L_VCB(R5)           ; No more VCB
  55    30 A5    D0             01A5   979          MOVL    UCB$L_LINK(R5),R5       ; Get next UCB address
             EF    12            01A9   980          BNEQ    110$                    ; If NEQ more to go
                                01AB   981          ;
                                01AB   982          ;  Decrement AQB reference count, and make sure its zero.
                                01AB   983          ;
```

```
                51    10 A4    D0  01AB   984          MOVL    RCB$L_AQB(R4),R1        ; Get the AQB address
                   0B A1    97  01AF   985          DECB    AQB$B_MNTCNT(R1)        ; Another device not mounted
                      04    13  01B2   986          BEQL    115$                    ; If equal, then ok
                                01B4   987          BUG_CHECK NETNOSTATE,FATAL      ; Only one device allowed for now
                                01B8   988          ;
                                01B8   989          ;   Unhook the AQB from the system AQB list
                                01B8   990          ;
             52 FFFFFFF0'GF     9E  01B8   991 115$:  MOVAB   G^IOC$GL_AQBLIST-AQB$L_LINK,R2  ; Setup for AQB scan
                      04    11  01BF   992          BRB     130$                    ; Start scan from listhead
                52    10 A2    D0  01C1   993 120$:  MOVL    AQB$L_LINK(R2),R2       ; Get next AQB
                51    10 A2    D1  01C5   994 130$:  CMPL    AQB$L_LINK(R2),R1       ; Does this point to our AQB
                      F6    12  01C9   995          BNEQ    120$                    ; If NEQ no
                   10 A1    D0  01CB   996          MOVL    AQB$L_LINK(R1),-        ; Unhook the AQB
                   10 A2        01CE   997                  AQB$L_LINK(R2)          ;
                      51    DD  01D0   998          PUSHL   R1                      ; Include AQB in deallocation list
                0000'CF     D4  01D2   999          CLRL    NET$GL_PTR_AQB          ; Say "no AQB"
                0000'CF     D4  01D6  1000          CLRL    NET$GL_PTR_VCB          ; Say "no VCB"
                                01DA  1001          ;
                                01DA  1002          ;
                                01DA  1003          ;   Deallocate the RCB and all associated data structures
                                01DA  1004          ;
                                01DA  1005          ;
                   24 A4    DD  01DA  1006          PUSHL   RCB$L_PTR_LTB(R4)       ; Build list of blocks to deallocate
                   28 A4    DD  01DD  1007          PUSHL   RCB$L_PTR_LPD(R4)       ; Get LPD block offset by 8
                      03    13  01E0  1008          BEQL    150$
             6E       08    C2  01E2  1009          SUBL    #8,(SP)                 ; Adjust actual start of block
                2C A4    DD  01E5  1010 150$:  PUSHL   RCB$L_PTR_ADJ(R4)       ; Get ADJ block offset by 8
                      03    13  01E8  1011          BEQL    152$
             6E       08    C2  01EA  1012          SUBL    #8,(SP)                 ; Adjust actual start of block
                34 A4    DD  01ED  1013 152$:  PUSHL   RCB$L_PTR_NDC(R4)       ; Get actual start of block
                1C A4    DD  01F0  1014          PUSHL   RCB$L_PTR_OA(R4)        ; Get OA block offset by 12
                      03    13  01F3  1015          BEQL    154$
             6E       0C    C2  01F5  1016          SUBL    #12,(SP)                ; Adjust actual start of block
                20 A4    DD  01F8  1017 154$:  PUSHL   RCB$L_PTR_AOA(R4)       ; Get AOA block offset by 12
                      03    13  01FB  1018          BEQL    170$
             6E       0C    C2  01FD  1019          SUBL    #12,(SP)                ; Adjust actual start of block
                                0200  1020
                   50 8ED0    0200  1021 170$:  POPL    R0                      ; Get next block
                      FB    13  0203  1022          BEQL    170$                    ; If EQL then try next entry
                05 50    E8  0205  1023          BLBS    R0,190$                 ; If LBS then at end of list
                 FDF5'       30  0208  1024          BSBW    NET$DEALLOCATE          ; Deallocate the block pointed to by R0
                      F3    11  020B  1025          BRB     170$
                      05    020D  1026 190$:  RSB
                                020E  1027
                               000001B0  1028          .RESTORE_PSECT
```

```
                                                B 1
                                01B0  1030          .SBTTL  I/O data base synchronization
                                01B0  1031 ;+
                                01B0  1032 ; LOCK_IODB      - Lock the I/O data base for write
                                01B0  1033 ; UNLOCK_IODB    - Unlock the I/O data base
                                01B0  1034 ;
                                01B0  1035 ; CALLING SEQUENCE:
                                01B0  1036 ;
                                01B0  1037 ;       CALL LOCK_IODB ()
                                01B0  1038 ;        ALL UNLOCK_IODB ()
                                01B0  1039 ;
                                01B0  1040 ; ROUTINE VALUE:
                                01B0  1041 ;       NONE
                                01B0  1042 ;
                                01B0  1043 ; SIDE EFFECTS:
                                01B0  1044 ;
                                01B0  1045 ;       I/O Data Base MUTEX and IPL are affected as shown
                                01B0  1046 ;
                                01B0  1047 ;--
                                01B0  1048 LOCK_IODB::
                         003F   01B0  1049          .WORD   ^M<R0,R1,R2,R3,R4,R5>   ; Save registers
     50  00000000'GF         DE  01B2  1050          MOVAL   G^IOC$GL_MUTEX,R0       ; Specify I/O data base mutex
     54  00000000'GF         D0  01B9  1051          MOVL    G^CTL$GL_PCB,R4         ; Get own PCB address
         00000000'GF         16  01C0  1052          JSB     G^SCH$LOCKW             ; Get mutex, raise to IPL$_ASTDEL
                             04  01C6  1053          RET
                                 01C7  1054
                                 01C7  1055 UNLOCK_IODB::
                         003F   01C7  1056          .WORD   ^M<R0,R1,R2,R3,R4,R5>   ; Save registers
     50  00000000'GF         DE  01C9  1057          MOVAL   G^IOC$GL_MUTEX,R0       ; Get I/O data base mutex
     54  00000000'GF         D0  01D0  1058          MOVL    G^CTL$GL_PCB,R4         ; And own PCB address
         00000000'GF         16  01D7  1059          JSB     G^SCH$UNLOCK            ; Return mutex
                                 01DD  1060          SETIPL  #0                      ; Also lower ipl
                             04  01E0  1061          RET
```

```
                                01E1  1063          .SBTTL  START_TIMER - Startup ACP activity timer
                                01E1  1064 ;+
                                01E1  1065 ; START_TIMER   - Startup ACP activity timer
                                01E1  1066 ;
                                01E1  1067 ; FUNCTIONAL DESCRIPTION:
                                01E1  1068 ;
                                01E1  1069 ; This routine is called to start the ACP activity timer. This timer is
                                01E1  1070 ; mearly used to reset the timeout cell in the RCB. The XPORT layer is
                                01E1  1071 ; continuously decrementing the activity timer to see if the ACP has been
                                01E1  1072 ; stalled - as in the event that the cluster in re-configuring. This will
                                01E1  1073 ; make the XPORT stop transmitting 'hello' messages and helps to prevent
                                01E1  1074 ; the stalled system from becoming a 'black hole'. The 'black hole' effect
                                01E1  1075 ; is caused because the ACP cannot process events and so cann send routing
                                01E1  1076 ; messages to notify other systems when a node becomes 'unreachable'.
                                01E1  1077 ;
                                01E1  1078 ; Inputs:
                                01E1  1079 ;                   R5 = WQE address or zero
                                01E1  1080 ;
                                01E1  1081 ; Outputs:
                                01E1  1082 ;                   None.
                                01E1  1083 ;
                                01E1  1084 ;                   R0-R2 are destroyed.
                                01E1  1085 ;
                                01E1  1086 ;-
                                01E1  1087 NET$START_TIMER:
                7E    53   7D   01E1  1088          MOVQ    R3,-(SP)                        ; Save R3,R4
                50    55   D0   01E4  1089          MOVL    R5,R0                           ; Copy WQE address
                      03   13   01E7  1090          BEQL    10$                             ; Br if none
                         FE14' 30   01E9  1091          BSBW    WQE$DEALLOCATE                  ; Deallocate the WQE
                52   0000'CF   D0   01EC  1092 10$:   MOVL    NET$GL_PTR_VCB,R2               ; Get RCB address
                      31   13   01F1  1093          BEQL    90$                             ; Br if not available
                51   0403 8F   3C   01F3  1094          MOVZWL  #<<WQE$C_QUAL_ACT>a8>!-         ; Set activity timer i.d.
                                01F8  1095                    NET$C_TID_ACT,R1
                         FE05' 30   01F8  1096          BSBW    WQE$CANCEL_TIM                  ; Cancel all previous timers
                                01FB  1097          $DISPATCH  RCB$B_ETY(R2),TYPE=B,-
                                01FB  1098          <-              ; type          ; action
                                01FB  1099                  <ADJ$C_PTY_AREA 50$>,-          ; Areas, have timers
                                01FB  1100                  <ADJ$C_PTY_PH4  50$>,-          ; So do other routers,
                                01FB  1101                  <ADJ$C_PTY_PH3  50$>,-          ;  but endnodes don't
                                01FB  1102          >
                                020B  1103
                      17   11   020B  1104          BRB     90$                             ; All others, just exit
                                020D  1105
                      1E   90   020D  1106 50$:    MOVB    #NET$C_ACT_TIMER,-              ; Reset activity timer
                   008F C2      020F  1107                    RCB$B_ACT_TIMER(R2)
                52    CC AF   9E   0212  1108          MOVAB   B^NET$START_TIMER,R2            ; Set up action routine
53   00000000 08F0D180 8F   7D   0216  1109          MOVQ    #5*1000*1000*NET$C_ACT_TIMER,R3 ; Set half the delta interval
                         FDDC' 30   0221  1110          BSBW    WQE$RESET_TIM                   ; Reactivate the activity timer
                53    8E   7D   0224  1111 90$:    MOVQ    (SP)+,R3                        ; Restore R3,R4
                           05   0227  1112          RSB                                     ; Return to caller
```

```
                                0228  1114          .SBTTL  PROC_EVT - Process ACP event
                                0228  1115 ;+
                                0228  1116 ; PROC_EVT      - Process ACP event
                                0228  1117 ;
                                0228  1118 ; FUNCTIONAL DESCRIPTION:
                                0228  1119 ;
                                0228  1120 ; This routine processes the ACP events and is state table driven.  Action
                                0228  1121 ; routines are called until the null event is detected.  Each action routine
                                0228  1122 ; generates a new event, which it returns in R1, and returns with the low bit
                                0228  1123 ; set in R0 only if the indicated state change is to be performed.
                                0228  1124 ;
                                0228  1125 ; INPUTS:        R1      Event code
                                0228  1126 ;
                                0228  1127 ; OUTPUTS:       R0      Status
                                0228  1128 ;                 All registers are clobbered
                                0228  1129 ;
                                0228  1130 ;-
                                0228  1131 NET$LOCLPD_DOWN::                                      ; Local LPD shut down
                  OFFC 8F   BB  0228  1132          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>    ; Save regs
                  51   06   D0  022C  1133          MOVL    S^#ACP$C_LPD_LOC,R1                     ; Set event
                       05   10  022F  1134          BSBB    PROC_EVT                                ; Process it
                  OFFC 8F   BA  0231  1135          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>    ; Restore regs
                            05  0235  1136          RSB
                                0236  1137
                                0236  1138
                                0236  1139 PROC_EVT:                                             ; Process all ACP events
                  50   01   D0  0236  1140          MOVL    #1,R0                                   ; Assume success
              52  0000'CF   D0  0239  1141          MOVL    NET$GL_PTR_VCB,R2                       ; Get RCB pointer
                                023E  1142          ;
                                023E  1143          ;
                                023E  1144          ;   Find appropriate state table entry
                                023E  1145          ;
                                023E  1146          ;
                                023E  1147 5$:      ASSUME  ACP$C_EVT_NOP  EQ  0
                    54   51   D0  023E  1148          MOVL    R1,R4                               ; Save event; is this the NOP event ?
                         67   13  0241  1149          BEQL    25$                                 ; If so, we're done
                    51   07   D1  0243  1150          CMPL    #ACP$C_MAX_EVT,R1                   ; Is event within range ?
                         63   1F  0246  1151          BLSSU   30$                                 ; If LSSU then bug exists
                    51   06   C4  0248  1152          MULL    #ACP$C_STATES,R1                    ; Bias for current event
                53     61 A2   9A  024B  1153          MOVZBL  RCB$B_STI(R2),R3                    ; Get internal ACP state
                    51   53   C0  024F  1154          ADDL    R3,R1                               ; Add current state offset
              51  0000'CF41   3E  0252  1155          MOVAW   ACP$AW_STA_TAB[R1],R1               ; Address state table entry
                                0258  1156          ;
                                0258  1157          ;
                                0258  1158          ;   Dispatch to the action routine with the following:
                                0258  1159          ;
                                0258  1160          ;   INPUTS:     R2      RCB ptr
                                0258  1161          ;               R1,R0   Scratch
                                0258  1162          ;
                                0258  1163          ;   ON RETURN:  R1      Next event to be processed
                                0258  1164          ;               R0      Low bit set if state change is permitted,
                                0258  1165          ;                       Low bit clear to avoid state change
                                0258  1166          ;
                                0258  1167          ;               All other regs may be clobbered
                                0258  1168          ;
                                0258  1169          ;
                001A'CF    51   D0  0258  1170          MOVL    R1,SAVE_STA_TAB                     ; Save state table address
```

```
                FDA0'  30  025D  1171          BSBW    NET$JNX_CO                      ; Log this event into the journal
                       0260  1172                                                      ; Clobbers R0
              22 50    E9  0260  1173          BLBC    R0,7$                           ; If LBC journalling is inactive
           81   AE 8F   9B  0263  1174          MOVZBW  #^X<AE>,(R1)+                   ; Enter journal record type
     00000000'8F   C3  0267  1175          SUBL3   #ACP$AW_STA_TAB,-
                                                   SAVE_STA_TAB,(R1)+                  ; Enter state table offset
           81   0C A2   B0  0271  1177          MOVW    RCB$W_TRANS(R2),(R1)+           ; Enter transaction count
           81   54 A2   B0  0275  1178          MOVW    RCB$W_MCOUNT(R2),(R1)+          ; Enter mount count
        81   0082 C2    B0  0279  1179          MOVW    RCB$W_MAX_PKT(R2),(R1)+         ; Enter max xmt IRPs allowed
        81   0080 C2    B0  027E  1180          MOVW    RCB$W_CUR_PKT(R2),(R1)+         ; Enter current xmt IRPs active
                 9E    16  0283  1181          JSB     @(SP)+                          ; Journal it
                 54    DD  0285  1182  7$:     PUSHL   R4                              ; Save original event code
              7E 81    9A  0287  1183          MOVZBL  (R1)+,-(SP)                     ; Save next state value
                 52    DD  028A  1184          PUSHL   R2                              ; Save RCB
              51 61    9A  028C  1185          MOVZBL  (R1),R1                         ; Get action routine index
     51   0060'CF41    D0  028F  1186          MOVL    ACP$AL_ACTION[R1],R1            ; Address action routine
                 61    16  0295  1187          JSB     (R1)                            ; Dispatch
                 1C    BA  0297  1188  10$:    POPR    #^M<R2,R3,R4>                   ; Get RCB ptr, next state and orig. event
                 A2 50 E9  0299  1189          BLBC    R0,5$                           ; Avoid state change if LBC
           61 A2    53    91  029C  1190          CMPB    R3,RCB$B_STI(R2)                ; Any change in state?
                 9C    13  02A0  1191          BEQL    5$                              ; Branch if not
                 0B    10  02A2  1192          BSBB    100$                            ; Log executor state change event
           61 A2    53    90  02A4  1193          MOVB    R3,RCB$B_STI(R2)                ; Change state
                 94    11  02A8  1194          BRB     5$                              ; Process next event
                 05    02AA  1195  25$:    RSB
                       02AB  1196
                       02AB  1197  30$:    BUG_CHECK NETNOSTATE,FATAL                  ; Bugcheck
                       02AF  1198
                       02AF  1199  ;
                       02AF  1200  ; Log an event indicating that the executor state has changed
                       02AF  1201  ;
                       02AF  1202  ; R3 = New state
                       02AF  1203  ; R2 = RCB address
                       02AF  1204  ; R4 = Event which triggered state change
                       02AF  1205  ;
                       02AF  1206
                    06 BB  02AF  1207  100$:   PUSHR   #^M<R1,R2>                      ; Save registers
                    51 D4  02B1  1208          CLRL    R1                              ; Indicate no extra WQE space needed
              50    01 D0  02B3  1209          MOVL    #WQE$C_SUB_ACP,R0               ; Set WQE subtype
                 FD47' 30  02B6  1210          BSBW    WQE$ALLOCATE                    ; Allocate a WQE to log an event
              55    52 D0  02B9  1211          MOVL    R2,R5                           ; Transfer WQE address
                    06 BA  02BC  1212          POPR    #^M<R1,R2>                      ; Restore registers
1C A5    0080 8F    B0  02BE  1213          MOVW    #EVC$C_SCL_LNS,WQE$W_EVL_CODE(R5) ; Set event code
     18 A5    01 90  02C4  1214          MOVB    #EVC$C_SCL_PRSN_NOR,WQE$L_EVL_PKT(R5) ; Assume "normal operation"
              01    54 91  02C8  1215          CMPB    R4,#ACP$C_OPR_INIT              ; Is it one of the "operator events"?
                    09 1F  02CB  1216          BLSSU   110$                            ; Branch if not
              05    54 91  02CD  1217          CMPB    R4,#ACP$C_OPR_OFF
                    04 1A  02D0  1218          BGTRU   110$
     18 A5    00    90  02D2  1219          MOVB    #EVC$C_SCL_PRSN_OPC,WQE$L_EVL_PKT(R5) ; Set "operator initated"
     50    61 A2    90  02D6  1220  110$:    MOVB    RCB$B_STI(R2),R0                ; Get previous internal state
                 1A    10  02DA  1221          BSBB    200$                            ; Convert to EVL coding scheme
     1E A5    50    90  02DC  1222          MOVB    R0,WQE$B_EVL_DT1(R5)            ; Set previous executor state in event
              50    53 90  02E0  1223          MOVB    R3,R0                           ; Get new internal state
                    11 10  02E3  1224          BSBB    200$                            ; Convert to EVL coding scheme
     1F A5    50    90  02E5  1225          MOVB    R0,WQE$B_EVL_DT2(R5)            ; Set new executor state
                 FD14' 30  02E9  1226          BSBW    NET$EVT_INTRAW                  ; Log the event
              50    55 D0  02EC  1227          MOVL    R5,R0                           ; Get WQE address
```

```
                FDOE'  30  02EF  1228          BSBW    WQE$DEALLOCATE          ; Deallocate the WQE
            50    01   D0  02F2  1229          MOVL    #1,R0                   ; Successful
                       05  02F5  1230          RSB
                           02F6  1231
                           02F6  1232 ;
                           02F6  1233 ; Convert internal ACP state (R0) to EVL state (R0)
                           02F6  1234 ;
                           02F6  1235
                       51  DD  02F6  1236 200$:  PUSHL   R1                     ; Save registers
          51   0090'CF  9E  02F8  1237          MOVAB   EVL_STA_MAP,R1          ; Point to translation table
                       61  95  02FD  1238 210$:  TSTB    (R1)                    ; End of table?
                       0A  19  02FF  1239          BLSS    220$                    ; If so, use default state
                   50  61  91  0301  1240          CMPB    (R1),R0                 ; Does state match?
                       05  13  0304  1241          BEQL    220$                    ; Branch if so
                   51  02  C0  0306  1242          ADDL    #2,R1                   ; Skip to next entry
                       F2  11  0309  1243          BRB     210$
             50    01  A1  9A  030B  1244 220$:  MOVZBL  1(R1),R0                ; Return EVL state code
                       51 8ED0  030F  1245          POPL    R1                      ; Restore registers
                          05  0312  1246          RSB
```

```
                     0313  1248          .SBTTL  Event action routines
                     0313  1249 ;+
                     0313  1250 ; ACT_REVIVE    - Try to turn the node back on from the "shut" or "off" state
                     0313  1251 ; ACT_NODE_SHUT - Turn the local node to the "shut" state
                     0313  1252 ; ACT_NODE_OFF  - Turn the local node off
                     0313  1253 ; ACT_CLEANUP   - Break all logical links, turn off all lines
                     0313  1254 ; ACT_NOP       - Nop action routine
                     0313  1255 ; ACT_BUG       - Bug_check
                     0313  1256 ; ACT_ERROR     - Return error
                     0313  1257 ;
                     0313  1258 ;-
                     0313  1259          .SAVE_PSECT
                0000020E  1260          .PSECT  NET_LOCK_CODE,NOWRT,GBL
                     020E  1261
                     020E  1262 ACT_REVIVE:                              ; Try to abort the dismount procedure
                     020E  1263          DSBINT  #NET$C_IPL             ; Raise IPL to sync with NETDRIVER
           54 A2  B5 0214  1264          TSTW    RCB$W_MCOUNT(R2)       ; Does NETDRIVER know we're dismounting
              06  13 0217  1265          BEQL    10$                    ; If EQL then yes, we must allow the
                     0219  1266                                         ; dismount to complete
           54 A2  B6 0219  1267          INCW    RCB$W_MCOUNT(R2)       ; Cancel the dismount
        50    01  D0 021C  1268          MOVL    #1,R0                  ; Indicate success
                     021F  1269 10$:     ENBINT                         ; Restore IPL
            0111' 31 0222  1270          BRW     RETURN_NULL            ; Return to terminate event processing
                     0225  1271
                0000313  1272          .RESTORE_PSECT
                     0313  1273
                     0313  1274 ACT_NODE_SHUT:                          ; Turn local node to the "shut" state
            FF0F' 30 0313  1275          BSBW    NET$DECR_MCOUNT        ; Return ACP's claim on the RCB
                     0316  1276                                         ; Fall thru
                     0316  1277 ACT_BRDCST:
              2C 10 0316  1278          BSBB    MBX_NET_SHUT           ; Tell the world we're going away
              19 11 0318  1279          BRB     ACT_NOP                ; Return success and null new event
                     031A  1280
                     031A  1281 ACT_NODE_OFF:                           ; Turn local node off
            FF08' 30 031A  1282          BSBW    NET$DECR_MCOUNT        ; Return ACP's claim on the RCB
                     031D  1283                                         ; Fall thru
                     031D  1284 ACT_CLEANUP:                            ; Break all links
              25 10 031D  1285          BSBB    MBX_NET_SHUT           ; Tell the world we're going away
        55 0000'CF  D0 031F  1286          MOVL    NET$GL_PTR_UCB0,R5     ; Setup UCB pointer
        51   24 A2 D0 0324  1287          MOVL    RCB$L_PTR_LTB(R2),R1   ; Get LTB pointer
              06  13 0328  1288          BEQL    20$                    ; If EQL then no LTB
           50 08  9A 032A  1289          MOVZBL  #NETUPD$_ABOLNK,R0     ; Fct code is "abort all links"
            FE2D 30 032D  1290          BSBW    CALL_NETDRIVER         ; Call NETDRIVER to abort the links
            FCCD' 30 0330  1291 20$:    BSBW    NET$DLL_ALL_OFF        ; Pass to module which knows about lines
                     0333  1292                                         ; Fall thru
                     0333  1293 ACT_NOP:                                ; Nop Action routine
           50 01  90 0333  1294          MOVB    #1,R0                  ; Allow state change
                     0336  1295 RETURN_NULL:
                     0336  1296          ASSUME  ACP$C_EVT_NOP  EQ  0
              51  D4 0336  1297          CLRL    R1                     ; Signal done
                  05 0338  1298          RSB                            ;
                     0339  1299
                     0339  1300 ACT_ERROR:                              ; Return error
        50 0000'8F  3C 0339  1301          MOVZWL  #SS$_WRITLCK,R0        ; Indicate wrong state
              F6  11 033E  1302          BRB     RETURN_NULL            ; Return null new event
                     0340  1303
                     0340  1304 ACT_STALL:                              ; Stall to prevent further processing
```

```
0340  1305
0340  1306 ACT_BUG:
0340  1307          BUG_CHECK NETNOSTATE,FATAL      ; Bugcheck
```

```
                          0344  1309          .SBTTL  MBX_NET_SHUT - Broadcast shutdown message
                          0344  1310 ;++
                          0344  1311 ;
                          0344  1312 ; MBX_NET_SHUT - Broadcast msg that node is shutting down
                          0344  1313 ;
                          0344  1314 ;--
                          0344  1315 MBX_NET_SHUT:
              3F    BB    0344  1316          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save regs
        55  0000'CF    D0  0346  1317          MOVL    NET$GL_PTR_UCB0,R5      ; Get the master UCB address
              53    7C    034B  1318          CLRQ    R3                      ; Broadcast to everyone and no data
           52   3B    3C  034D  1319          MOVZWL  #MSG$_NETSHUT,R2        ; Mailbox message code
           50   0A    3C  0350  1320          MOVZWL  #NETUPD$_BRDCST,R0      ; This is the function
                FE07  30  0353  1321          BSBW    CALL_NETDRIVER          ; Call the driver
              3F    BA  0356  1322          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore regs
                    05  0358  1323          RSB                             ; Done
```

```
                     0359  1325          .SBTTL  NET$DECR_MCOUNT - Decrement ACP mount count
                     0359  1326 ;+
                     0359  1327 ; NET$DECR_MCOUNT - Decrement ACP mount count
                     0359  1328 ;
                     0359  1329 ; The ACP mount count is decremented.  If the mount count goes to zero,
                     0359  1330 ; then the "ACP dismounting" flag is set, causing the VMS executive to
                     0359  1331 ; prevent any further ACCESS QIO's from coming in.
                     0359  1332 ;
                     0359  1333 ; Inputs:
                     0359  1334 ;
                     0359  1335 ;       R2 = Address of RCB
                     0359  1336 ;
                     0359  1337 ; Outputs:
                     0359  1338 ;
                     0359  1339 ;       R0 destroyed.
                     0359  1340 ;-
                     0359  1341          .SAVE_PSECT
                00000225  1342          .PSECT  NET_LOCK_CODE,NOWRT,GBL
                     0225  1343
                     0225  1344 NET$DECR_MCOUNT::
          54 A2      B7  0225  1345          DECW    RCB$W_MCOUNT(R2)        ; Decrement mount count
             0A      14  0228  1346          BGTR    90$                     ; Exit if still positive
  50    0000'CF      D0  022A  1347          MOVL    NET$GL_PTR_UCB0,R0      ; Get address of base UCB
00 38 A0   15      E2  022F  1348          BBSS    #DEV$V_DMT,UCB$L_DEVCHAR(R0),90$ ; Make VMS stop sending ACCESS QIO'
                     05  0234  1349 90$:    RSB
                     0235  1350
                00000359  1351          .RESTORE
```

```
                     0359  1353          .SBTTL  NET$UPD_LOCAL - Update local state
                     0359  1354 ;+
                     0359  1355 ; NET$UPD_LOCAL - Update local state
                     0359  1356 ;
                     0359  1357 ; FUNCTIONAL DESCRIPTION:
                     0359  1358 ;
                     0359  1359 ; The contents of the new LNI CNF are assume to have been already checked
                     0359  1360 ; for consistency with the NDI data base.  The CNF values are used to update
                     0359  1361 ; the structure and content of the control blocks residing in non-paged pool.
                     0359  1362 ;
                     0359  1363 ; INPUTS:        R11     LNI CNR pointer
                     0359  1364 ;                 R10     New LNI CNF pointer
                     0359  1365 ;
                     0359  1366 ; OUTPUTS:       R11,R10 Clobbered
                     0359  1367 ;                 R9      Bit i.d. of bad parameter if LBC in R0
                     0359  1368 ;                 R8-R1   Clobbered
                     0359  1369 ;                 R0      Status
                     0359  1370 ;-
                     0359  1371 NET$UPD_LOCAL::                             ; Update local state
                     0359  1372          ;
                     0359  1373          ;   If the SEGMENT BUFFER SIZE is not specified, then zero
                     0359  1374          ;   the LNI cell.   Check to make sure that an explicit SBS
                     0359  1375          ;   value is always less than or equal to the BUS value.
                     0359  1376          ;
                     0359  1377          $GETFLD lni,l,bus                   ; Get forwarding buffer size
          56   58    D0  0364  1378          MOVL    R8,R6                   ; Save it
                     0367  1379          $GETFLD lni,l,sbs                   ; SEGMENT BUFFER SIZE specified?
    3C AA   58    B0  0372  1380          MOVW    R8,CNF$C_LENGTH -           ; Overwrite cell in CNF, so that SBS
                     0376  1381                  +LNI$W_SBS(R10)             ; cell is zeroed if not set
       08 50    E9  0376  1382          BLBC    R0,2$                       ; Branch if not specified
    58    56    D1  0379  1383          CMPL    R6,R8                       ; Is BUS < SBS?
          03    18  037C  1384          BGEQ    2$                          ; If not, everything is ok
        01AF    31  037E  1385          BRW     300$                        ; If so, report an error
                     0381  1386 2$:      ;
                     0381  1387          ;   For a number of LNI parameters, if the parameter is not "set",
                     0381  1388          ;   then zero the corresponding LNI field to make life easier later.
                     0381  1389          ;   This code relies on $GETFLD returning R8=0 if the parameter is
                     0381  1390          ;   not "set".
                     0381  1391          ;
                     0381  1392          $GETFLD lni,l,piq                   ; PIPELINE QUOTA
      53 AA   58    B0  038C  1393          MOVW    R8,CNF$C_LENGTH+LNI$W_PIQ(R10)
                     0390  1394          $GETFLD lni,l,mar                   ; MAX AREAS
      5A AA   58    90  039B  1395          MOVB    R8,CNF$C_LENGTH+LNI$B_MAR(R10)
                     039F  1396          $GETFLD lni,l,ali                   ; alias address (cluster address)
      5E AA   58    B0  03AA  1397          MOVW    R8,CNF$C_LENGTH+LNI$W_ALI(R10)
                     03AE  1398          ;
                     03AE  1399          ;
                     03AE  1400          ;   Setup nonpaged CNF image
                     03AE  1401          ;
                     03AE  1402          ;
    56   0004'CF    9E  03AE  1403          MOVAB   NEW_LNI_IMAGE,R6            ; Point to non-pageable CNF image
         0050 8F    28  03B3  1404          MOVC3   #LNI$C_LENGTH,-             ; Copy the basic new LNI block (no CNF
      66   24 AA        03B7  1405                  CNF$C_LENGTH(R10),(R6)      ; superstructure and no string storage)
    58   0000'CF    D0  03BA  1406          MOVL    NET$GL_PTR_VCB,R8           ; Get RCB pointer
                     03BF  1407          ;
                     03BF  1408          ;
                     03BF  1409          ;   Map the requested state into an event code
```

```
                                                     L 1
                     03BF  1410          ;
                     03BF  1411          ;
                     03BF  1412          $CNFFLD lni,l,sta,R9                ; Identify state field in case of error
52   0084'CF    9E   03C6  1413          MOVAB   OPR_EVT_MAP,R2              ; Get reguested state/event map
0000'CF   62    9A   03CB  1414  10$:    MOVZBL  (R2),ACP_L_EVT              ; Assume this is it
      07   82    91   03D0  1415          CMPB    (R2)+,#ACP$C_EVT_BUG        ; At end of table ?
           65    13   03D3  1416          BEQL    110$                        ; If EQL then invalid state value
   02 A6   82    91   03D5  1417          CMPB    (R2)+,LNI$B_STA(R6)         ; Does the entry match new state?
           F0    12   03D9  1418          BNEQ    10$                         ; If NEQ then keep trying
                     03DB  1419          ;
                     03DB  1420          ;
                     03DB  1421          ;   New node address may only be zero if the new state is OFF/INIT/SHUT
                     03DB  1422          ;
                     03DB  1423          ;
                     03DB  1424          $CNFFLD lni,l,add,R9                ; Identify parameter assuming error
           66    B5   03E2  1425          TSTW    LNI$W_ADD(R6)               ; Test the new local node address
           15    12   03E4  1426          BNEQ    30$                         ; If NEQ then check is unnecessary
01   0000'CF    D1   03E6  1427          CMPL    ACP_L_EVT,#ACP$C_OPR_INIT   ; Is the ACP initializing
           0E    13   03EB  1428          BEQL    30$                         ; If so, allow it
04   0000'CF    D1   03ED  1429          CMPL    ACP_L_EVT,#ACP$C_OPR_SHUT   ; Is the ACP shutting down?
           07    13   03F2  1430          BEQL    30$                         ; If so, allow it
05   0000'CF    D1   03F4  1431          CMPL    ACP_L_EVT,#ACP$C_OPR_OFF    ; Is the ACP shutting down?
           3F    12   03F9  1432          BNEQ    110$                        ; If NEQ then bad ADDRESS parameter
                     03FB  1433          ;
                     03FB  1434          ;
                     03FB  1435          ;   If the local address is already non-zero then it must not be
                     03FB  1436          ;   changed again.
                     03FB  1437          ;
                     03FB  1438          ;
   0E A8    66    B1   03FB  1439  30$:    CMPW    LNI$W_ADD(R6),RCB$W_ADDR(R8) ; Are the addresses the same?
           05    13   03FF  1440          BEQL    60$                         ; If EQL then okay
      0E A8    B5   0401  1441          TSTW    RCB$W_ADDR(R8)              ; Is the address currently zero?
           34    12   0404  1442          BNEQ    110$                        ; If NEQ report bad address, else okay
                     0406  1443  60$:    ;
                     0406  1444          ;   If a local alias address is being specified, make sure it is
                     0406  1445          ;   in the same area as in primary local address.
                     0406  1446          ;
                     0406  1447          $CNFFLD lni,l,ali,R9                ; Set field ID in case of error
      3A A6    B5   040D  1448          TSTW    LNI$W_ALI(R6)               ; Alias specified?
           0D    13   0410  1449          BEQL    100$                        ; Skip if not
           0A    EF   0412  1450          EXTZV   #TR4$V_ADDR_AREA,-          ; Get area of alias address
50    3A A6    06        0414  1451                  #TR4$S_ADDR_AREA,LNI$W_ALI(R6),R0
           0A    ED   0418  1452          CMPZV   #TR4$V_ADDR_AREA,-          ; Make sure it matches our area
   50   66    06        041A  1453                  #TR4$S_ADDR_AREA,LNI$W_ADD(R6),R0
           1B    12   041D  1454          BNEQ    110$                        ; If NEQ, report bad address
                     041F  1455  100$:    ;
                     041F  1456          ;   The EXECUTOR TYPE parameter is only allowed to be Phase IV
                     041F  1457          ;   routing, Phase IV area routing, Phase IV endnode or Phase III.
                     041F  1458          ;   In addition, TYPE is not allowed to be changed while there are
                     041F  1459          ;   active datalinks in operation.
                     041F  1460          ;
                     041F  1461          $CNFFLD lni,l,ety,R9                ; Set field ID in case of error
   50   03 A6    9A   0426  1462          MOVZBL  LNI$B_ETY(R6),R0            ; Get executor type
                     042A  1463          $DISPATCH R0,<-                     ; Allow the following values:
                     042A  1464                  <ADJ$C_PTY_PH3,120$>,-      ; Phase III routing
                     042A  1465                  <ADJ$C_PTY_PH4,120$>,-      ; Phase IV routing
                     042A  1466                  <ADJ$C_PTY_PH4N,120$>,-     ; Phase IV endnode
```

```
                        042A  1467                    <ADJ$C_PTY_AREA,120$>>  ; Phase IV area routing
          00F3    31    043A  1468 110$:    BRW       300$                    ; Otherwise, report an error
008A C8     50    91    043D  1469 120$:    CMPB      R0,RCB$B_ETY(R8)        ; Same as existing type?
              2A  13    0442  1470          BEQL      200$                    ; Skip check if no change
                        0444  1471
          28 A8 D5    0444  1472          TSTL      RCB$L_PTR_LPD(R8)       ; Check LPD vector
              1B  13    0447  1473          BEQL      190$                    ; If EQL then none
       52   5C A8 9A    0449  1474          MOVZBL    RCB$B_MAX_LPD(R8),R2    ; Get number of circuits
            51  01 D0    044D  1475          MOVL      #LPD$C_LOC_INX,R1       ; Start just after local LPD
                0E  11    0450  1476          BRB       180$
          28 B841 D5    0452  1477 160$:    TSTL      @RCB$L_PTR_LPD(R8)[R1]  ; Check if LPD slot active
                08  18    0456  1478          BGEQ      180$                    ; Branch if slot not in use
50      0000'8F   3C    0458  1479 170$:    MOVZWL    #SS$_WRITLCK,R0         ; Indicate "executor in wrong state"
              01E5  31    045D  1480          BRW       400$                    ; Report the error
  EE 51     52    F3    0460  1481 180$:    AOBLEQ    R2,R1,160$              ; Loop through all LPDs
                        0464  1482
                        0464  1483 190$:    ASSUME    CNR$L_FLINK EQ 0
51      0000'CF   D0    0464  1484          MOVL      NET$GL_CNR_PLI,R1       ; Get PLI root pointer
            51  61  D1    0469  1485          CMPL      CNR$L_FLINK(R1),R1      ; Is CNF list empty?
                EA  12    046C  1486          BNEQ      170$                    ; If not, error - all lines must be cleared
                        046E  1487 200$:    ;
                        046E  1488          ;   Do not allow MAXIMUM CIRCUITS to be increased if there are
                        046E  1489          ;   BRAs or BEAs active.  This is because there is a one-to-one
                        046E  1490          ;   correspondence between the LPD vector and the first NC ADJs.
                        046E  1491          ;
7E    0E A6   01    81    046E  1492          ADDB3     #1,LNI$B_MLN(R6),-(SP)  ; Get new Max circuits
                        0473  1493                                            ; plus one for local LPD
          5C A8   8E  91    0473  1494          CMPB      (SP)+,RCB$B_MAX_LPD(R8) ; Same as existing size?
                26  13    0477  1495          BEQL      230$                    ; Skip check if no change
              2C A8 D5    0479  1496          TSTL      RCB$L_PTR_ADJ(R8)       ; Check ADJ vector
                21  13    047C  1497          BEQL      230$                    ; Skip if none
          52   68 A8 3C    047E  1498          MOVZWL    RCB$W_MAX_ADJ(R8),R2    ; Get number of adjacencies
                1B  13    0482  1499          BEQL      230$                    ; Skip if none
          51   5C A8 9A    0484  1500          MOVZBL    RCB$B_MAX_LPD(R8),R1    ; Get number of circuits
                11  11    0488  1501          BRB       220$                    ; Start at NC+1
50      2C B841   D0    048A  1502 210$:    MOVL      @RCB$L_PTR_ADJ(R8)[R1],R0 ; Get ADJ address
  08 60     00    E1    048F  1503          BBC       #ADJ$V_INUSE,ADJ$B_STS(R0),220$ ; Branch if slot unused
50      0000'8F   3C    0493  1504          MOVZWL    #SS$_WRITLCK,R0         ; Indicate "executor in wrong state"
              01AA  31    0498  1505          BRW       400$                    ; Report the error
  EB 51     52    F3    049B  1506 220$:    AOBLEQ    R2,R1,210$              ; Loop through all ADJs
                        049F  1507 230$:    ;
                        049F  1508          ;
                        049F  1509          ;   MAX ADDRESS must be larger than any of the adjacent
                        049F  1510          ;   node addresses but not so large that a routing message
                        049F  1511          ;   could not be sent to the adjacent node.
                        049F  1512          ;
                        049F  1513          ;
                        049F  1514          $CNFFLD lni,l,mad,R9              ; Specify MAX ADDRESS in case error
          02    A5    04A6  1515          MULW3     #NET$C_TRCTL_CEL,-      ; Determine variable size of Phase III
       51   06 A6          04A8  1516                    LNI$W_MAD(R6),R1    ; routing message
            51  05    A0    04AB  1517          ADDW      #NET$C_TRCTL_OVR,R1     ; Add in fixed size of routing msg
                        04AE  1518          ;
                        04AE  1519          ;   Scan ADJ vector to determine the max Phase III partner's node
                        04AE  1520          ;   address and the minimum adjacent routing node's buffer size.
                        04AE  1521          ;
              2C A8 D5    04AE  1522          TSTL      RCB$L_PTR_ADJ(R8)       ; Check ADJ vector
                3F  13    04B1  1523          BEQL      280$                    ; If EQL then none
```

```
                54    68 A8    3C   04B3  1524                MOVZWL  RCB$W_MAX_ADJ(R8),R4     ; Get number of cells
             50    2C B844     D0   04B7  1525  240$:         MOVL    @RCB$[_PTR_ADJ(R8)[R4],R0 ; Get ADJ address
               2F 60    00     E1   04BC  1526                BBC     #ADJ$V_INUSE,ADJ$B_STS(R0),270$ ; Skip if slot not in use
               2B 60    01     E1   04C0  1527                BBC     #ADJ$V_RUN,ADJ$B_STS(R0),270$ ; Skip if circuit not up
                        0A     EF   04C4  1528                EXTZV   #TR4$V_ADDR_AREA,-        ; Get the area for adjacency
          52    04 A0    06          04C6  1529                        #TR4$S_ADDR_AREA,ADJ$W_PNA(R0),R2
                        07     13   04CA  1530                BEQL    250$                      ; If area = 0, then use our area
                        0A     ED   04CC  1531                CMPZV   #TR4$V_ADDR_AREA,-        ; Is this adjacency in our area?
             52    66    06          04CE  1532                        #TR4$S_ADDR_AREA,LNI$W_ADD(R6),R2
                        0C     12   04D1  1533                BNEQ    260$                      ; If not, then skip MAX ADDR check
                        00     EF   04D3  1534  250$:         EXTZV   #TR4$V_ADDR_DEST,-        ; Get the node within area
          53    04 A0    0A          04D5  1535                        #TR4$S_ADDR_DEST,ADJ$W_PNA(R0),R3
               06 A6    53     B1   04D9  1536                CMPW    R3,LNI$W_MAD(R6)          ; Does partner have a larger address?
                        51     1A   04DD  1537                BGTRU   300$                      ; If so, invalid MAX ADDRESS param
                        02     E1   04DF  1538  260$:         BBC     #ADJ$V_RTG,-              ; If partner non-routing,
                     0C 60          04E1  1539                        ADJ$B_STS(R0),270$        ; then ignore buffer size
                     01 A0    91   04E3  1540                CMPB    ADJ$B_PTYPE(R0),-         ; Is it a Phase III adjacency?
                              00    04E6  1541                        #ADJ$C_PTY_PH3
                              06    12   04E7  1542                BNEQ    270$                      ; If not, don't worry about routing msgs
               51    06 A0    B1   04E9  1543                CMPW    ADJ$W_BUFSIZ(R0),R1       ; Is partner's buffer too small
                              41    1F   04ED  1544                BLSSU   300$                      ; If LSSU, invalid MAX ADDRESS param
                     C5 54    F5   04EF  1545  270$:         SOBGTR  R4,240$                   ; Loop for each adjacency
                                   04F2  1546  280$:         ;
                                   04F2  1547                ;
                                   04F2  1548                ;   The FORWARDING BUFFER SIZE must be as least 192 bytes (max. size
                                   04F2  1549                ;   NSP connect initiate message plus route-header is 190 bytes  --
                                   04F2  1550                ;   192 was chosen since that was the DECnet-VAX version 1 minimum)
                                   04F2  1551                ;
                                   04F2  1552                ;   Also, check to make sure that SEGMENT BUFFER SIZE is larger than
                                   04F2  1553                ;   the minimum buffer size.
                                   04F2  1554                ;
                                   04F2  1555                ;
                                   04F2  1556                $CNFFLD lni,l,bus,R9              ; Identify param in case of error
                  16 A6    B1   04F9  1557                CMPW    LNI$W_BUS(R6),-           ; Does buffer meet the minimum size
                00C0 8F          04FC  1558                        #NET$C_MINBUFSIZ          ; requirements?
                        2F     1F   04FF  1559                BLSSU   300$                      ; If LSSU report invalid BUFFER SIZE
                               0501  1560                $CNFFLD lni,l,sbs,R9              ; Identify param in case of error
             50    18 A6    3C   0508  1561                MOVZWL  LNI$W_SBS(R6),R0          ; Get segment buffer size
                        07     13   050C  1562                BEQL    290$                      ; Branch if not set
          00C0 8F    50    B1   050E  1563                CMPW    R0,#NET$C_MINBUFSIZ       ; SBS big enough?
                        1B     1F   0513  1564                BLSSU   300$                      ; Error if not
                               0515  1565  290$:         ;
                               0515  1566                ;
                               0515  1567                ;   MAXIMUM VISITS must be at least as large as MAXIMUM HOPS
                               0515  1568                ;
                               0515  1569                ;
                               0515  1570                $CNFFLD lni,l,mvi,R9              ; Identify  MAX VISITS in case error
          0C A6    0D A6    91   051C  1571                CMPB    LNI$B_MVI(R6),LNI$B_MHO(R6) ; Is MAX VISITS gequ MAX HOPS
                        0D     1F   0521  1572                BLSSU   300$                      ; If LSSU report MAX VISITS is invalid
                               0523  1573                ;
                               0523  1574                ;
                               0523  1575                ;    DELAY FACTOR must be at least 16
                               0523  1576                ;
                               0523  1577                ;
                               0523  1578                $CNFFLD lni,l,dfa,R9              ; Identify DELAY FACTOR in case error
             2A A6    10   91   052A  1579                CMPB    #16,LNI$B_DFA(R6)         ; Is DELAY FACTOR large enough ?
                        06     1B   052E  1580                BLEQU   310$                      ; If GTRU report DELAY FACTOR invalid
```

```
                50   00'  3C  0530  1581 300$:   MOVZWL  S^#SS$_BADPARAM,R0      ; Indicate error
                   010F   31  0533  1582          BRW     400$
                          0536  1583 310$:   ;
                          0536  1584         ;
                          0536  1585         ;    Make sure that default access states have values
                          0536  1586         ;
                          0536  1587         ;
                   0180 8F   BB  0536  1588          PUSHR   #^M<R7,R8>              ; Save regs
                          053A  1589          $GETFLD lni,l_dpx                       ; Get current default proxy access value
                     07 50   E8  0545  1590          BLBS    R0,320$                 ; If LBS then okay
                58   00      9A  0548  1591          MOVZBL  #NMA$C_ACES_NONE,R8     ; Else setup default value
            2E A6   58       90  054B  1592          MOVB    R8,LNI$B_DPX(R6)        ; And in its non-pageable copy
                          054F  1593 320$:   $GETFLD lni,l_dac                       ; Get current default access value
                0A 50   E8  055A  1594          BLBS    R0,330$                 ; If LBS then okay
                58   03      9A  055D  1595          MOVZBL  #NMA$C_ACES_BOTH,R8     ; Else setup default value
                   FA9D'    30  0560  1596          BSBW    CNF$PUT_FIELD           ; Store in the new CNF entry
            2D A6   58       90  0563  1597          MOVB    R8,LNI$B_DAC(R6)        ; And in its non-pageable copy
                   0180 8F   BA  0567  1598 330$:   POPR    #^M<R7,R8>              ; Restore regs
                          056B  1599         ;
                          056B  1600         ;   If X.25 datalink mapping is enabled (by the presence of
                          056B  1601         ;   the EXECUTOR SUBADDRESSES parameter), then notify PSI that
                          056B  1602         ;   we want to handle a given range of incoming calls.
                          056B  1603         ;
                     00D8   30  056B  1604          BSBW    NET$DECLARE_PSI         ; Declare PSI process
                   07 50   E8  056E  1605          BLBS    R0,340$                 ; Branch if ok
            0000'8F   50   B1  0571  1606          CMPW    R0,#SS$_NOSUCHDEV       ; Is PSI not yet initialized?
                      06   12  0576  1607          BNEQ    409$                    ; If not, defer until actually needed
                          0578  1608 340$:   ;
                          0578  1609         ;
                          0578  1610         ;   Update non-paged control structures
                          0578  1611         ;
                          0578  1612         ;
                    FCBA'  30  0578  1613          BSBW    UPDATE_DATABASE         ; Change state, update database
                  03 50   E8  057B  1614          BLBS    R0,350$                 ; If LBS then okay
                   00C4   31  057E  1615 409$:   BRW     400$                    ; Else report error
            0000'CF   5A   D0  0581  1616 350$:   MOVL    R10,NET$GL_PTR_LNI      ; Save pointer to LNI CNF
                          0586  1617         ;
                          0586  1618         ;
                          0586  1619         ;   Update miscellaneous RCB fields.  Any races with respect to
                          0586  1620         ;   NETDRIVER regarding the update of the RCB fields are of no
                          0586  1621         ;   consequence since they are independent values and are not pointers.
                          0586  1622         ;   IPL cannot be raised here since the LNI block is not locked down.
                          0586  1623         ;
                          0586  1624         ;
         008A C8   03 A6   90  0586  1625          MOVB    LNI$B_ETY(R6),RCB$B_ETY(R8)     ; Store node type
               0E A8   66  B0  058C  1626          MOVW    LNI$W_ADD(R6),RCB$W_ADDR(R8)    ; Local address
         008D C8   3A A6   B0  0590  1627          MOVW    LNI$W_ALI(R6),RCB$W_ALIAS(R8)   ; Alias local address
                  06   0A   EF  0596  1628          EXTZV   #TR4$V_ADDR_AREA,#TR4$S_ADDR_AREA,-
                  50   66       0599  1629                  LNI$W_ADD(R6),R0                ; Extract our area address
               008B C8   50   90  059B  1630          MOVB    R0,RCB$B_HOMEAREA(R8)           ; Store in a convenient place
                          05A0  1631         ;
      50   2F A6   16 A6   A7  05A0  1632          DIVW3   LNI$W_BUS(R6),LNI$W_PIQ(R6),R0  ; Get packets in pipeline quota
               62 A8   50   90  05A6  1633          MOVB    R0,             RCB$B_ECL_RFLW(R8) ; Store in RCB
                       10   87  05AA  1634          DIVB3   #16,-
               64 A8   2A A6       05AC  1635                  LNI$B_DFA(R6),RCB$B_ECL_DFA(R8) ; Delay factor
               65 A8   2B A6   90  05B0  1636          MOVB    LNI$B_DWE(R6),RCB$B_ECL_DWE(R8) ; Delay weight
               63 A8   2C A6   90  05B5  1637          MOVB    LNI$B_RFA(R6),RCB$B_ECL_RFA(R8) ; Rexmt factor
```

```
                66 A8   2D A6   90  05BA  1638          MOVB    LNI$B_DAC(R6),RCB$B_ECL_DAC(R8) ; Default access state
                67 A8   2E A6   90  05BF  1639          MOVB    LNI$B_DPX(R6),RCB$B_ECL_DPX(R8) ; Default proxy access state
                                    05C4  1640
                76 A8   1E A6   B0  05C4  1641          MOVW    LNI$W_ITI(R6),RCB$W_TIM_CNI(R8) ; Inbound connect timer
                78 A8   20 A6   B0  05C9  1642          MOVW    LNI$W_OTI(R6),RCB$W_TIM_CNO(R8) ; Outbound connect timer
                74 A8   1C A6   B0  05CE  1643          MOVW    LNI$W_IAT(R6),RCB$W_TIM_IAT(R8) ; Inactivity timer
                                    05D3  1644
             5C A8   0E A6    01   81  05D3  1645          ADDB3   #1,LNI$B_MLN(R6),RCB$B_MAX_LPD(R8) ; Max circuits
                                    05D9  1646                                                  ; plus one for local LPD
                50   5C A8   9A  05D9  1647          MOVZBL  RCB$B_MAX_LPD(R8),R0                ; Get number of LPDs
          6A A8   50   28 A6   A1  05DD  1648          ADDW3   LNI$W_MBR(R6),R0,-                  ; # of "routing destinations"
                                    05E3  1649                  RCB$W_MAX_RTG(R8)               ; (NC + NBRA)
             6A A8   26 A6   A1  05E3  1650          ADDW3   LNI$W_MBE(R6),RCB$W_MAX_RTG(R8),- ; # of adjacencies
                     68 A8          05E8  1651                  RCB$W_MAX_ADJ(R8)               ; (NC + NBRA + NBEA)
                                    05EA  1652
          008C C8   36 A6   90  05EA  1653          MOVB    LNI$B_MAR(R6),RCB$B_MAX_AREA(R8) ; Max area address
            5A A8   06 A6   B0  05F0  1654          MOVW    LNI$W_MAD(R6),RCB$W_MAX_ADDR(R8); Max node addr
            5E A8   0D A6   B0  05F5  1655          MOVW    LNI$B_MVI(R6),RCB$B_MAX_VISIT(R8) ; Max visits
            58 A8   04 A6   B0  05FA  1656          MOVW    LNI$W_MLK(R6),RCB$W_MAX_LNK(R8) ; Max logical links
                                    05FF  1657          ;
                                    05FF  1658          ;   The total datalink buffer size is calculated here.  The
                                    05FF  1659          ;   size is computed by adding the EXECUTOR BUFFER SIZE (which
                                    05FF  1660          ;   includes the NSP header plus exactly 6 bytes), plus the
                                    05FF  1661          ;   CXB overhead used by datalink drivers.
                                    05FF  1662          ;
                   004C 8F   A1  05FF  1663          ADDW3   #CXB$C_OVERHEAD,-                   ; Total buffer size
             7E A8   16 A6        0603  1664                  LNI$W_BUS(R6),RCB$W_TOTBUFSIZ(R8)
                                    0607  1665          ;
                                    0607  1666          ;
                                    0607  1667          ;    Update state sensitive settings
                                    0607  1668          ;
                                    0607  1669          ;
                                    0607  1670          $DISPATCH TYPE=B,RCB$B_STI(R8),-            ; Case on internal state
                                    0607  1671          <-
                                    0607  1672                  <ACP$C_STA_N,   370$>,-          ; "on"
                                    0607  1673                  <ACP$C_STA_R,   370$>,-          ; "restrict"
                                    0607  1674                  <ACP$C_STA_S,   370$>,-          ; "shut"
                                    0607  1675                  <ACP$C_STA_F,   390$>,-          ; "off" - avoid setting MAX_PKT
                                    0607  1676                  -                               ;         to allow run-down
                                    0607  1677                  -                               ; else fall thru
                                    0607  1678          >
       0082 C8   08 A6   B0  0614  1679          MOVW    LNI$W_MBU(R6),RCB$W_MAX_PKT(R8) ; Max xmt packets
          50   00C0 8F   B0  061A  1680          MOVW    #NET$C_MINBUFSIZ,R0             ; Use smallest possible size
                   10   11  061F  1681          BRB     380$
       0082 C8   08 A6   B0  0621  1682 370$:    MOVW    LNI$W_MBU(R6),RCB$W_MAX_PKT(R8) ; Max xmt packets
             50   18 A6   3C  0627  1683          MOVZWL  LNI$W_SBS(R6),R0                ; Get segment buffer size
                   04   12  062B  1684          BNEQ    380$                            ; Branch if set
             50   16 A6   3C  062D  1685          MOVZWL  LNI$W_BUS(R6),R0                ; Else, use forwarding buffer size
                   0F   A3  0631  1686 380$:    SUBW3   #6+NSP$C_MAXHDR,-               ; ECL msg size = total msg size
             7C A8   50         0633  1687                  R0,RCB$W_ECLSEGSIZ(R8)          ; minus max route header size
                                    0636  1688          ;
                                    0636  1689          ;
                                    0636  1690          ;   Update the datalink control layer
                                    0636  1691          ;
                                    0636  1692          ;
                F9C7'   30  0636  1693 390$:    BSBW    NET$DLLUPDLNI
                09 50   E9  0639  1694          BLBC    R0,400$                         ; Br if error
```

```
                                                D  2
                     063C  1695          ;
                     063C  1696          ;   Start up timer that keeps 'hello' messages going
                     063C  1697          ;
          55   D4   063C  1698          CLRL    R5                              ; Indicate no WQE to deallocate
        FBA0   30   063E  1699          BSBW    NET$START_TIMER                 ; Startup the activity timer
  50  00'8F   9A   0641  1700          MOVZBL  #SS$_NORMAL,R0                  ; Timer is always successful
              05   0645  1701 400$:    RSB
                   0646  1702
```

```
                     0646  1704          .SBTTL  NET$DECLARE_PSI - Declare ourselves as a PSI process
                     0646  1705 ;+
                     0646  1706 ; NET$DECLARE_PSI - Declare ourselves as a PSI process to receive incoming calls
                     0646  1707 ;
                     0646  1708 ; Inputs:
                     0646  1709 ;
                     0646  1710 ;       R11 = LNI CNR address
                     0646  1711 ;       R10 = LNI CNF address
                     0646  1712 ;
                     0646  1713 ; Outputs:
                     0646  1714 ;
                     0646  1715 ;       R0 = Status code
                     0646  1716 ;
                     0646  1717 ;       R1 is destroyed.
                     0646  1718 ;-
                     0646  1719 NET$DECLARE_PSI::
 0180 8F     BB      0646  1720          PUSHR   #^M<R7,R8>                  ; Save registers
                     064A  1721          ;
                     064A  1722          ;   If X.25 datalink mapping is enabled (by the presence of
                     064A  1723          ;   the EXECUTOR SUBADDRESSES parameter), then notify PSI that
                     064A  1724          ;   we want to handle a given range of incoming calls.
                     064A  1725          ;
                     064A  1726          $GETFLD lni,l_sad                   ; Subaddresses specified?
   0F 50     E8      0655  1727          BLBS    R0,10$                      ; Branch if specified
 0026'CF     D5      0658  1728          TSTL    CURRENT_SAD                 ; Are we declared right now?
      03     13      065C  1729          BEQL    5$                          ; If not, then nothing to worry about
    00B7     30      065E  1730          BSBW    UNDECLARE_PSI               ; Remove existing declaration
50    01     D0      0661  1731 5$:      MOVL    #1,R0                       ; No subaddress - exit successful
    00AC     31      0664  1732          BRW     90$
                     0667  1733          ;
                     0667  1734          ;   If it hasn't changed, then don't bother to do anything
                     0667  1735          ;   (this prevents excessive QIOs when the network manager
                     0667  1736          ;   is constantly changing an executor parameter).
                     0667  1737          ;
      0026'CF    58  D1    0667  1738 10$:     CMPL    R8,CURRENT_SAD          ; Different than current declaration?
                 F3  13    066C  1739          BEQL    5$                      ; Branch if not
                     066E  1740          ;
                     066E  1741          ;   Remote the existing subaddress declaration.  This can only
                     066E  1742          ;   be done by deassigning the PSI channel, and reassigning it.
                     066E  1743          ;
          0026'CF    D5    066E  1744          TSTL    CURRENT_SAD             ; Are we declared right now?
                 03  13    0672  1745          BEQL    15$                     ; Branch if not
                 00A1  30    0674  1746          BSBW    UNDECLARE_PSI           ; Undeclare the subaddress declaration
                     0677  1747          ;
                     0677  1748          ;   Tell PSI that we want to handle the given range of calls
                     0677  1749          ;
                 7E  58  B0  0677  1750 15$:     MOVW    R8,-(SP)                ; Push lowest subaddress
                 7E  02  B0  067A  1751          MOVW    #PSI$C_NTD_SALO,-(SP)   ; Push item identifier
                 7E  06  B0  067D  1752          MOVW    #6,-(SP)                ; Push length of item
50    58    F0  8F  78  0680  1753          ASHL    #-16,R8,R0              ; Get high order word
                 7E  50  B0  0685  1754          MOVW    R0,-(SP)                ; Push highest subaddress
                 7E  03  B0  0688  1755          MOVW    #PSI$C_NTD_SAHI,-(SP)   ; Push item identifier
                 7E  06  B0  068B  1756          MOVW    #6,-(SP)                ; Push length of item
          334C3532  8F  DD  068E  1757          PUSHL   #^A"25L3"               ; Push the string "X25L3"
           7E    58  8F  90  0694  1758          MOVB    #^A"X",-(SP)
                 7E  05  90  0698  1759          MOVB    #5,-(SP)                ; Push byte count of X25L3 string
                 7E  01  B0  069B  1760          MOVW    #PSI$C_NTD_ACCLVL,-(SP) ; Push item identifier
```

```
                7E    0A    B0  069E  1761          MOVW    #10,-(SP)                       ; Push item length
                      5E    DD  06A1  1762          PUSHL   SP                              ; Construct descriptor of NTD
                      16    DD  06A3  1763          PUSHL   #22
                      7E    D4  06A5  1764          CLRL    -(SP)                           ; Construct NFB
                7E    15    90  06A7  1765          MOVB    #NFB$C_DECLNAME,-(SP)
                      5E    DD  06AA  1766          PUSHL   SP                              ; Construct descriptor of NFB
                      05    DD  06AC  1767          PUSHL   #5
              0000'CF       B5  06AE  1768          TSTW    NET$GW_X25_CHAN                 ; Have we tried to talk to PSI yet?
                      06    12  06B2  1769          BNEQ    30$                             ; If so, proceed
                 F949'      30  06B4  1770          BSBW    NET$GET_X25_CHAN                ; If not, try to grab PSI mutex
                56 50       E9  06B7  1771          BLBC    R0,50$                          ; Exit if error detected
              002A'CF       B5  06BA  1772  30$:     TSTW    PSI_DECL_CHAN                   ; Is there a "declare channel" to PSI?
                      18    12  06BE  1773          BNEQ    40$                             ; Branch if so
                            06C0  1774          $ASSIGN_S CHAN=PSI_DECL_CHAN,-              ; Assign a channel to PSI
                            06C0  1775                    DEVNAM=NET$GQ_X25_DEV,-
                            06C0  1776                    MBXNAM=NET$GQ_MBX_NAME
                38 50       E9  06D5  1777          BLBC    R0,50$                          ; Branch if unsuccessful
                50    5E    D0  06D8  1778  40$:     MOVL    SP,R0                           ; Point to NFB descriptor
             51    0D AE    9E  06DB  1779          MOVAB   8+5(SP),R1                      ; Point to NTD descriptor
                            06DF  1780          $QIOW_S CHAN=PSI_DECL_CHAN,-                ; Tell PSI to send us X.25 calls
                            06DF  1781                  FUNC=#IO$_ACPCONTROL,-
                            06DF  1782                  IOSB=IOSB,-
                            06DF  1783                  P1=(R0),-                       ; Address of NFB descriptor
                            06DF  1784                  P2=R1                           ; Address of NTD descriptor
                0D 50       E9  0700  1785          BLBC    R0,50$                          ; Branch if request failed
          50    001E'CF       3C  0703  1786          MOVZWL  IOSB,R0                         ; Get I/O status
                05 50       E9  0708  1787          BLBC    R0,50$                          ; Branch if I/O failed
          0026'CF    58       D0  070B  1788          MOVL    R8,CURRENT_SAD                  ; Store current subaddress declaration
                5E    2B      C0  0710  1789  50$:     ADDL    #8+5+8+22,SP                    ; Deallocate NFB and NTD
                0180 8F       BA  0713  1790  90$:     POPR    #^M<R7,R8>                      ; Restore registers
                            05  0717  1791          RSB                                     ; Exit with status
```

```
                     0718  1793          .SBTTL  UNDECLARE_PSI - Remove PSI declaration
                     0718  1794 ;+
                     0718  1795 ; UNDECLARE_PSI - Remove PSI subaddress declarations
                     0718  1796 ;
                     0718  1797 ; With the PSI software as it exists, the only method of removing a
                     0718  1798 ; subaddress declaration is to deassign the PSI channel.  This may
                     0718  1799 ; disrupt activity on the associated mailbox, but that's ok, since
                     0718  1800 ; the system manager must expect some disruption when the subaddress
                     0718  1801 ; range is changed.  We leave the channel deassigned, rather than
                     0718  1802 ; re-assigning it, since there may no longer be any need to have a
                     0718  1803 ; PSI channel if there are no more X.25 circuits, and the subaddress
                     0718  1804 ; parameter has been removed.  If there is a future need for a PSI
                     0718  1805 ; channel, it will be assigned just before using it.
                     0718  1806 ;
                     0718  1807 ; Inputs:
                     0718  1808 ;
                     0718  1809 ;       None
                     0718  1810 ;
                     0718  1811 ; Outputs:
                     0718  1812 ;
                     0718  1813 ;       None
                     0718  1814 ;-
                     0718  1815 UNDECLARE_PSI:
                     0718  1816          $DASSGN_S CHAN=PSI_DECL_CHAN    ; Remove X.25 declaration
002A'CF    B4        0724  1817          CLRW    PSI_DECL_CHAN           ; Indicate channel no longer active
0026'CF    D4        0728  1818          CLRL    CURRENT_SAD             ; Indicate nothing declared
           05        072C  1819          RSB
```

```
                                        .SBTTL  UPDATE_DATABASE - Update non-paged control blocks
                                ;+
                                ; UPDATE_DATABASE - Update nonpaged control blocks
                                ;
                                ; This routine updates/replaces the nonpaged control blocks as a result
                                ; of a change to the executor database.
                                ;
                                ; Inputs:
                                ;
                                ;       R11 = LNI CNR address
                                ;       R10 = New LNI CNF address
                                ;       R8 = RCB address
                                ;       R6 = Address of LNI nonpaged copy
                                ;
                                ; Outputs:
                                ;
                                ;       R0 = Status code
                                ;-
                                        .SAVE_PSECT
                        00000235        .PSECT  NET_LOCK_CODE,NOWRT,GBL
                                0235
                                0235    UPDATE_DATABASE:                        ; Build non-paged control blocks
                                0235
                        00000000        0235    LTB     =  0
                        00000004        0235    NDC     =  4
                        00000008        0235    LPD     =  8
                        0000000C        0235    OA      = 12
                        00000010        0235    ADJ     = 16
                        00000014        0235    AOA     = 20
                                0235
                         01     DD      0235    20$:    PUSHL   #1                      ; Mark end of deallocation list
                         7E     7C      0237            CLRQ    -(SP)                   ; Initialize storeage on stack for
                         7E     7C      0239            CLRQ    -(SP)                   ; new vector block addresses
                         7E     7C      023B            CLRQ    -(SP)
                                023D                    ;
                                023D                    ;
                                023D                    ;   Allocate and initialize whatever new control blocks are needed.
                                023D                    ;   These blocks cannot replace the old ones until the last operation
                                023D                    ;   which may possibly end in error has been performed.
                                023D                    ;
                                023D                    ;   Wherever possible, if an error condition arizes, the bit i.d. of
                                023D                    ;   the LNI parameter is returned in R9 and the error code in R0.
                                023D                    ;
                                023D                    ;
                          04ED' 30      023D            BSBW    BUILD_LTB               ; Build a new LTB
                        65 50   E9      0240            BLBC    R0,50$                  ; Br on error
                     6E    52   D0      0243            MOVL    R2,LTB(SP)              ; Store the new LTB address
                          0607' 30      0246            BSBW    BUILD_NDC               ; Else build a new vector
                        5C 50   E9      0249            BLBC    R0,50$                  ; Br on error
                  04 AE    52   D0      024C            MOVL    R2,NDC(SP)              ; Save its address
                          052D' 30      0250            BSBW    BUILD_LPD               ; Build the LPD vector
                        52 50   E9      0253            BLBC    R0,50$                  ; Br on error
                  08 AE    52   D0      0256            MOVL    R2,LPD(SP)              ; Store the LPD vector address
                          0618' 30      025A            BSBW    BUILD_OA                ; Build OA vector
                        48 50   E9      025D            BLBC    R0,50$                  ; Br on error
                  0C AE    52   D0      0260            MOVL    R2,OA(SP)               ; Save its address
                          0564' 30      0264            BSBW    BUILD_ADJ               ; Build the ADJ block vector
```

```
                                                     I  2
                     3E 50    E9  0267  1878                BLBC    R0,50$                  ; Br on error
                 10 AE   52   D0  026A  1879                MOVL    R2,ADJ(SP)              ; Store the ADJ block vector address
                       0629'  30  026E  1880                BSBW    BUILD_AOA               ; Build AOA vector
                     34 50    E9  0271  1881                BLBC    R0,50$                  ; Br on error
                 14 AE   52   D0  0274  1882                MOVL    R2,AOA(SP)              ; Save its address
                              0278  1883                ;
                              0278  1884                ;   Process the event
                              0278  1885                ;
                51   0000'CF   D0  0278  1886                MOVL    ACP_EVT,R1              ; Get the event
                     0D40 8F   BB  027D  1887                PUSHR   #^M<R6,R8,R10,R11>      ; Save regs
                        FFB2'  30  0281  1888                BSBW    PROC_EVT                ; Process event indicated in R1
                     0D40 8F   BA  0284  1889                POPR    #^M<R6,R8,R10,R11>      ; Restore regs
                       1D 50   E9  0288  1890                BLBC    R0,50$                  ; If LBC then event was aborted
                              028B  1891                ;
                              028B  1892                ;   Attach new control blocks to the RCB.  Get the data base mutex
                              028B  1893                ;   and raise IPL to synchronize with NETDRIVER.
                              028B  1894                ;
                01B0'CF   00  FB  028B  1895                CALLS   #0,LOCK_IODB            ; Get mutex
                     59   5E   D0  0290  1896                MOVL    SP,R9                   ; Save pointer to vector of blocks
                              0293  1897                DSBINT  #NET$C_IPL              ; Raise IPL
                          21  10  0299  1898                BSBB    REPLACE                 ; Replace old blocks with new blocks
                              029B  1899                ENBINT                          ; Restore IPL
                          50  DD  029E  1900                PUSHL   R0                      ; Save status
                01C7'CF   00  FB  02A0  1901                CALLS   #0,UNLOCK_IODB          ; Unlock the database
                          50 8ED0  02A5  1902                POPL    R0                      ; Restore status
                              02A8  1903                ;
                              02A8  1904                ;
                              02A8  1905                ;   Deallocate old or unused control blocks
                              02A8  1906                ;
                              02A8  1907                ;
            57    50    D0  02A8  1908  50$:    MOVL    R0,R7                   ; Save error code
                  50 8ED0  02AB  1909  100$:   POPL    R0                      ; Get next block
                     FB   13  02AE  1910          BEQL    100$                    ; If EQL then try next entry
                05 50     E8  02B0  1911          BLBS    R0,200$                 ; If LBS then at end of list
                   FD4A'  30  02B3  1912          BSBW    NET$DEALLOCATE          ; Deallocate the block pointed to by R0
                     F3   11  02B6  1913          BRB     100$                    ; Loop
            50    57    D0  02B8  1914  200$:   MOVL    R7,R0                   ; Restore error code
                        05  02BB  1915          RSB
```

```
                                02BC  1917 REPLACE:
                                02BC  1918          ;
                                02BC  1919          ;
                                02BC  1920          ;   Attach new LTB vector
                                02BC  1921          ;
                                02BC  1922          ;
                55    69    D0  02BC  1923          MOVL    LTB(R9),R5              ; Get new LTB address
                      1F    13  02BF  1924          BEQL    100$                    ; Branch if no new LTB to insert
          54    24    A8    D0  02C1  1925          MOVL    RCB$L_PTR_LTB(R8),R4    ; Get old LTB address
          24    A8    55    D0  02C5  1926          MOVL    R5,RCB$L_PTR_LTB(R8)    ; Attach new LTB
                69    54    D0  02C9  1927          MOVL    R4,LTB(R9)              ; Get old LTB address for deallocation
                      12    13  02CC  1928          BEQL    100$                    ; Br if none
       0C A5    0C    A4    D0  02CE  1929          MOVL    LTB$L_XWB(R4),LTB$L_XWB(R5) ; Fix up the linkage
          50    04    A4    3C  02D3  1930          MOVZWL  LTB$W_SLT_TOT(R4),R0    ; Get number of slots to copy
                50    04    C4  02D7  1931          MULL    #4,R0                   ; Get number of bytes to copy
          14    A4    50    28  02DA  1932          MOVC3   R0,LTB$L_SLOTS+4(R4),-  ; Copy the slots starting at slot #1
                14    A5        02DE  1933                      LTB$L_SLOTS+4(R5)   ; (slot #0 always contains a zero)
                                02E0  1934 100$:    ;
                                02E0  1935          ;
                                02E0  1936          ;   Attach new NDC vector
                                02E0  1937          ;
                                02E0  1938          ;
          55    04    A9    D0  02E0  1939          MOVL    NDC(R9),R5              ; Get new NDC address
                      1B    13  02E4  1940          BEQL    120$                    ; Branch if no new NDC block to insert
          54    34    A8    D0  02E6  1941          MOVL    RCB$L_PTR_NDC(R8),R4    ; Get old NDC address
          34    A8    55    D0  02EA  1942          MOVL    R5,RCB$L_PTR_NDC(R8)    ; Attach new NDC
          04    A9    54    D0  02EE  1943          MOVL    R4,NDC(R9)              ; Copy old NDC address for deallocation
                      0D    13  02F2  1944          BEQL    120$                    ; Br if none
          50    5A    A8    3C  02F4  1945          MOVZWL  RCB$W_MAX_ADDR(R8),R0   ; Get number of cells to copy
                50    1C    C4  02F8  1946          MULL    #NDC$C_LENGTH,R0        ; Get number of bytes to copy
    0C A5 0C    A4    50    28  02FB  1947          MOVC3   R0,12(R4),12(R5)        ; Copy valid cells, now that
                                0301  1948                                          ; NETDRIVER is interlocked
                                0301  1949 120$:    ;
                                0301  1950          ;
                                0301  1951          ;   Attach new LPD vector
                                0301  1952          ;
                                0301  1953          ;
          55    08    A9    D0  0301  1954          MOVL    LPD(R9),R5              ; Get new LPD vector
                      23    13  0305  1955          BEQL    135$                    ; If none, then leave LPD vector alone
          54    28    A8    D0  0307  1956          MOVL    RCB$L_PTR_LPD(R8),R4    ; Get old LPD vector address
                      03    13  030B  1957          BEQL    130$                    ; Branch if none
                54    08    C2  030D  1958          SUBL    #8,R4                   ; Get address of real LPD vector
          08    A9    54    D0  0310  1959 130$:    MOVL    R4,LPD(R9)              ; Copy old LPD vector address
                                0314  1960                                          ; for later deallocation
       28 A8    08    A5    9E  0314  1961          MOVAB   8(R5),RCB$L_PTR_LPD(R8) ; Attach new LPD vector
                      54    D5  0319  1962          TSTL    R4                      ; Any old LPD vector?
                      0D    13  031B  1963          BEQL    135$                    ; If not, skip copy
          50    5C    A8    9A  031D  1964          MOVZBL  RCB$B_MAX_LPD(R8),R0    ; Get old number of LPDs
                50    04    C4  0321  1965          MULL    #4,R0                   ; Compute size of old LPD vector
    0C A5 0C    A4    50    28  0324  1966          MOVC    R0,12(R4),12(R5)        ; Copy valid pointers, now that
                                032A  1967                                          ; NETDRIVER is interlocked
                                032A  1968
                                032A  1969 135$:    ;
                                032A  1970          ;
                                032A  1971          ;   Attach local LPD block to new LPD vector
                                032A  1972          ;
                                032A  1973          ;
```

```
                53   0000'CF   D0  032A  1974          MOVL    NET$GL_LOC_LPD,R3       ; Get address of local LPD
                  51     01    D0  032F  1975          MOVL    #LPD$C_LOC_INX,R1       ; Get local LPD index
             28 B841     53    D0  0332  1976          MOVL    R3,@RCB$L_PTR_LPD(R8)[R1] ; Store address into new LPD vector
                                   0337  1977          ;
                                   0337  1978          ;
                                   0337  1979          ;   Attach new OA vector
                                   0337  1980          ;
                                   0337  1981          ;
                55    0C A9   D0  0337  1982          MOVL    OA(R9),R5               ; Get new OA address
                      54   13  033B  1983          BEQL    140$                    ; Branch if no new OA to insert
                54    1C A8   D0  033D  1984          MOVL    RCB$L_PTR_OA(R8),R4     ; Get old OA vector address
                      03   13  0341  1985          BEQL    138$                    ; Branch if none
                   54   0C   C2  0343  1986          SUBL    #12,R4                  ; Get address of real OA vector
             1C A8    0C A5   9E  0346  1987 138$:   MOVAB   12(R5),RCB$L_PTR_OA(R8) ; Point to first entry in vector
                0C A9   54   D0  034B  1988          MOVL    R4,OA(R9)               ; Copy old OA vector address
                      0D   13  034F  1989          BEQL    139$                    ; for later deallocation
                50    5A A8   3C  0351  1990          MOVZWL  RCB$W_MAX_ADDR(R8),R0   ; Get number of cells to copy
                   50   02   C4  0355  1991          MULL    #2,R0                   ; Compute number of bytes to copy
          0C A5    0C A4   50   28  0358  1992          MOVC3   R0,12(R4),12(R5)        ; Copy valid cells, now that
                                   035E  1993                                          ; NETDRIVER is interlocked
                                   035E  1994 139$:    ;
                                   035E  1995          ;   Update path descriptor to local node
                                   035E  1996          ;
                      00   EF  035E  1997          EXTZV   #TR4$V_ADDR_DEST,-      ; Get old local address
                   50    0E A8   0A   0360  1998                  #TR4$S_ADDR_DEST,RCB$W_ADDR(R8),R0
                      1C B840   94   0364  1999          CLRB    @RCB$L_PTR_OA(R8)[R0]   ; Make unreachable in case local the
                                   0368  2000                                          ; local address is being changed
                      00   EF  0368  2001          EXTZV   #TR4$V_ADDR_DEST,-      ; Get old local alias address
                50    008D C8   0A   036A  2002                  #TR4$S_ADDR_DEST,RCB$W_ALIAS(R8),R0
                      1C B840   94   036F  2003          CLRB    @RCB$L_PTR_OA(R8)[R0]   ; Make unreachable in case local the
                                   0373  2004                                          ; alias is being changed
                        51   01   D0  0373  2005          MOVL    #LPD$C_LOC_INX,R1       ; Get local LPD/ADJ index
                   1C B8    51   B0  0376  2006          MOVW    R1,@RCB$L_PTR_OA(R8)    ; Node "0" is a synonym for the local addres
                      00   EF  037A  2007          EXTZV   #TR4$V_ADDR_DEST,-      ; Get new local address
                50    66    0A   037C  2008                  #TR4$S_ADDR_DEST,LNI$W_ADD(R6),R0
                   1C B840   51   B0  037F  2009          MOVW    R1,@RCB$L_PTR_OA(R8)[R0] ; Point OA for local node to local LPD
                      00   EF  0384  2010          EXTZV   #TR4$V_ADDR_DEST,-      ; Get new local alias address
             50    3A A6    0A   0386  2011                  #TR4$S_ADDR_DEST,LNI$W_ALI(R6),R0
                      05   13  038A  2012          BEQL    140$                    ; Skip if no alias specified
                   1C B840   51   B0  038C  2013          MOVW    R1,@RCB$L_PTR_OA(R8)[R0] ; Point OA for alias node to local LPD
                                   0391  2014 140$:    ;
                                   0391  2015          ;   Attach new AOA vector
                                   0391  2016          ;
                55    14 A9   D0  0391  2017          MOVL    AOA(R9),R5              ; Get new AOA address
                      3D   13  0395  2018          BEQL    144$                    ; Branch if no new OA to insert
                54    20 A8   D0  0397  2019          MOVL    RCB$L_PTR_AOA(R8),R4    ; Get old AOA vector address
                      03   13  039B  2020          BEQL    142$                    ; Branch if none
                   54   0C   C2  039D  2021          SUBL    #12,R4                  ; Get address of real AOA vector
             20 A8    0C A5   9E  03A0  2022 142$:   MOVAB   12(R5),RCB$L_PTR_AOA(R8) ; Point to first entry in vector
                14 A9   54   D0  03A5  2023          MOVL    R4,AOA(R9)              ; Copy old AOA vector address
                      0E   13  03A9  2024          BEQL    143$                    ; for later deallocation
                50    008C C8   9A  03AB  2025          MOVZBL  RCB$B_MAX_AREA(R8),R0   ; Get number of cells to copy
                   50   02   C4  03B0  2026          MULL    #2,R0                   ; Compute number of bytes to copy
          0C A5    0C A4   50   28  03B3  2027          MOVC3   R0,12(R4),12(R5)        ; Copy valid cells, now that
                                   03B9  2028                                          ; NETDRIVER is interlocked
                                   03B9  2029 143$:    ;
                                   03B9  2030          ;   Update path descriptor to local area
```

```
                          03B9  2031          ;
                    0A    EF  03B9  2032          EXTZV   #TR4$V_ADDR_AREA,-       ; Get old local area
50   0E A8    06          03BB  2033                  #TR4$S_ADDR_AREA,RCB$W_ADDR(R8),R0
      20 B840  94          03BF  2034          CLRB    @RCB$L_PTR_AOA(R8)[R0]  ; Make unreachable in case local the
                          03C3  2035                                          ; local address is being changed
                    0A    EF  03C3  2036          EXTZV   #TR4$V_ADDR_AREA,-       ; Get new local area
   50   66    06          03C5  2037                  #TR4$S_ADDR_AREA,LNI$W_ADD(R6),R0
      51    01    D0  03C8  2038          MOVL    #LPD$C_LOC_INX,R1        ; Get local LPD/ADJ index
   20 B840    51    B0  03CB  2039          MOVW    R1,@RCB$L_PTR_AOA(R8)[R0] ; Point AOA for local area to local LPD
     20 B8    51    B0  03D0  2040          MOVW    R1,@RCB$L_PTR_AOA(R8)   ; Area "0" is a synonym for the local area
                          03D4  2041  144$:   ;
                          03D4  2042          ;   Attach new ADJ block vector
                          03D4  2043          ;
      55   10 A9    D0  03D4  2044          MOVL    ADJ(R9),R5               ; Get new ADJ address
                 33    13  03D8  2045          BEQL    149$                     ; Branch if no new ADJ to insert
      54   2C A8    D0  03DA  2046          MOVL    RCB$L_PTR_ADJ(R8),R4     ; Get old ADJ block vector address
                 03    13  03DE  2047          BEQL    145$                     ; Branch if none
         54    08    C2  03E0  2048          SUBL    #8,R4                    ; Get address of real ADJ vector
      10 A9    54    D0  03E3  2049  145$:   MOVL    R4,ADJ(R9)               ; Copy old ADJ vector address
                          03E7  2050                                          ; for later deallocation
   2C A8    08 A5    9E  03E7  2051          MOVAB   8(R5),RCB$L_PTR_ADJ(R8) ; Point to first entry in vector
                 54    D5  03EC  2052          TSTL    R4                       ; Any old ADJ vector?
                 1D    13  03EE  2053          BEQL    149$                     ; Branch if not
      51   68 A8    3C  03F0  2054          MOVZWL  RCB$W_MAX_ADJ(R8),R1     ; Get previous number of ADJ slots
         51    0D    C4  03F4  2055          MULL    #ADJ$C_LENGTH,R1         ; Compute # bytes which are still valid
      50   04 A4    D0  03F7  2056          MOVL    4(R4),R0                 ; Get old number of adjacencies in block
   50   0C A440    DE  03FB  2057          MOVAL   12(R4)[R0],R0            ; Point to first old ADJ(1)
      52   04 A5    D0  0400  2058          MOVL    4(R5),R2                 ; Get number of new adjacencies in block
   52   0C A542    DE  0404  2059          MOVAL   12(R5)[R2],R2            ; Point to first new ADJ(1)
   62   60   51    28  0409  2060          MOVC    R1,(R0),(R2)             ; Copy old adjacencies to new
                          040D  2061  149$:   ;
                          040D  2062          ;   Initialize local adjacency block
                          040D  2063          ;
         2C A8    D5  040D  2064          TSTL    RCB$L_PTR_ADJ(R8)        ; Is there an ADJ vector?
                 1E    13  0410  2065          BEQL    150$                     ; Skip if none
         50   01    D0  0412  2066          MOVL    #LPD$C_LOC_INX,R0        ; Get local ADJ index
   50    2C B840    D0  0415  2067          MOVL    @RCB$L_PTR_ADJ(R8)[R0],R0 ; Get local ADJ address
06 A0    7FFF  8F    B0  041A  2068          MOVW    #32767,ADJ$W_BUFSIZ(R0) ; Remove restrictions on block size
                 03    90  0420  2069          MOVB    #ADJ$M_INUSE!ADJ$M_RUN,-         ; Set slot in use
                 60        0422  2070                  ADJ$B_STS(R0)                    ; and mark it up
   01 A0    03 A6    90  0423  2071          MOVB    LNI$B_ETY(R6),ADJ$B_PTYPE(R0)    ; Set executor node type
      04 A0    66    B0  0428  2072          MOVW    LNI$W_ADD(R6),ADJ$W_PNA(R0)      ; Set new local address
      02 A0    01    90  042C  2073          MOVB    #LPD$C_LOC_INX,ADJ$B_LPD_INX(R0); Set LPD index of circuit
         50    01    90  0430  2074  150$:   MOVB    #1,R0                    ; Set success flag
                 05    0433  2075          RSB                              ; Done
                          0434  2076
                    0000072D  2077          .RESTORE_PSECT
```

```
                                        .SBTTL  BUILD_LTB - Build logical link table
                                ;+
                                ; BUILD_LTB     - Build Logical Link Table
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; The LTB contains a vector of pointers to the logical link context blocks
                                ; (XWBs).  When a given cell in the dispatch vector is not pointing to an XWB
                                ; its low bit is set and its high word contains the last 'address' to be used
                                ; for this slot.
                                ;
                                ; The low order portion of a link 'address' is its index into the vector.  The
                                ; high order portion is a sequence number used to prevent cross-talk between
                                ; links when the slots are reused.
                                ;
                                ; The first slot in the vector corresonds to link index zero and is unused.
                                ; The vector is terminated with a longword -1 followed by a longword zero.
                                ; This is done in order to allow NETDRIVER to optimize its search for a
                                ; free vector slot.
                                ;
                                ; The total number of slots in the vector is greater than the maximum number
                                ; of allowed links in order to prevent sequence numbers from cycling too
                                ; rapidly when the current number of links is close to the maximum number of
                                ; links.  This assumes that links slots are assigned in round-robin fashion
                                ; rather than first-available-first-used starting from the top.
                                ;
                                ; If there is no current LTB, or if the old LTB is too small to support the
                                ; maximum links, then a new table is allocated and initialized.
                                ;
                                ; INPUTS:       R10     Ptr to new LNI CNF
                                ;               R8      Ptr to RCB
                                ;               R6      Ptr to start of non-pageable new LNI image
                                ;               R5-R0   Scratch
                                ;
                                ; OUTPUTS:      R11,R10 Preserved
                                ;               R9      Bit i.d. used to qualify the error
                                ;               R8-R6   Preserved
                                ;               R2      New LTB pointer or zero if no new LTB needed
                                ;                       Invalid if R0 has low bit clear.
                                ;               R0      Status
                                ;
                                ;-
                                BUILD_LTB:                                      ; Build logical link table
                        52  D4          CLRL    R2                              ; Assume no new LTB is needed
                    55  04 A6  3C      MOVZWL  LNI$W_MLK(R6),R5                ; Get new max links
                        55  1E  C0      ADDL    #NSP$C_EXT_LNK,R5               ; Add in extra slots
                    51  24 A8  D0      MOVL    RCB$L_PTR_LTB(R8),R1            ; Get current LTB
                           06  13      BEQL    15$                             ; If EQL then none
                    04 A1  55  B1      CMPW    R5,LTB$W_SLT_TOT(R1)            ; Can current LTB be used ?
                           35  1B      BLEQU   25$                             ; If LEQU then yes
                                       ;
                                       ;
                                       ;   Allocate a new LTB.  Note that the +12 below is for the two marker
                                       ;   slots at the end of the LTB and for slot #0 (at offset LTB$L_SLOTS)
                                       ;   which is not counted in LTB$W_SLT_TOT and which is never used.
                                       ;
                                       ;
```

```
                                                     N 2
51   1C A245   DE  0742  2136 15$:    MOVAL   LTB$L_SLOTS+12(R2)[R5],R1 ; Get total LTB size - assumes R2=0
                 0747  2137           $CNFFLD lni,l,mlk,R9              ; Specify 'max links' in case there's
                 074E  2138                                             ; insufficent memory
       F8AF'     30  074E  2139           BSBW    NET$ALONPAGED         ; Get buffer
     2B 50       E9  0751  2140           BLBC    R0,30$                ; Br if error
                 0754  2141           ;
                 0754  2142           ;
                 0754  2143           ;   Initialize  LTB
                 0754  2144           ;
                 0754  2145           ;
     10 A2       9E  0754  2146           MOVAB   LTB$L_SLOTS(R2),-     ;
           62        0757  2147                   LTB$L_SLT_NXT(R2)     ; Setup "next slot" candidate
  04 A2    55       B0  0758  2148           MOVW    R5,LTB$W_SLT_TOT(R2)  ; Setup total slots - does not include
                 075C  2149                                             ; the two marker slots at the end,
                 075C  2150                                             ; or slot number zero.
     0C A2       D4  075C  2151           CLRL    LTB$L_XWB(R2)         ; Terminate the XWB chain
51   1C A245     DE  075F  2152           MOVAL   LTB$L_SLOTS+12(R2)[R5],R1 ; Point just past slot vector
           71    D4  0764  2153           CLRL    -(R1)                 ; End of slot marker
     71    01     CE  0766  2154           MNEGL   #1,-(R1)              ; End of slot marker
     71    55     B0  0769  2155 20$:      MOVW    R5,-(R1)              ; Enter slot index value
     71    01     B0  076C  2156           MOVW    #1,-(R1)              ; Enter slot available flag
       F7  55     F5  076F  2157           SOBGTR  R5,20$                ; Continue for entire vector
           71    D4  0772  2158           CLRL    -(R1)                 ; Slot zero is not usable
     51    52     D0  0774  2159           MOVL    R2,R1                 ; Copy LTB pointer
       04  A6     B0  0777  2160 25$:      MOVW    LNI$W_MLK(R6),-       ; Setup the number of slots that may be
     06    A1           077A  2161                   LTB$W_SLT_LMT(R1)     ; in use at any given time.
     50    01     90  077C  2162           MOVB    #1,R0                 ; Set low bit for success
                 05  077F  2163 30$:      RSB
```

```
                                0780  2165          .SBTTL  BUILD_LPD - Build the LPD vector
                                0780  2166 ;+
                                0780  2167 ; BUILD_LPD     - Build the LPD vector
                                0780  2168 ;
                                0780  2169 ; FUNCTIONAL DESCRIPTION:
                                0780  2170 ;
                                0780  2171 ; The maximum allowed datalinks can be modified only if the there is no current
                                0780  2172 ; LPD vector.  This restriction may be relaxed in future releases.
                                0780  2173 ;
                                0780  2174 ; The LPD vector is allocated and intialized.
                                0780  2175 ;
                                0780  2176 ; INPUTS:        R10     Ptr to new LNI CNF
                                0780  2177 ;                 R8      Ptr to RCB
                                0780  2178 ;                 R6      Ptr to start of non-pageable new LNI image
                                0780  2179 ;                 R5-R0   Scratch
                                0780  2180 ;
                                0780  2181 ; OUTPUTS:       R11,R10 Preserved
                                0780  2182 ;                 R9      LNI bit i.d. used to qualify any errors
                                0780  2183 ;                 R8-R6   Preserved
                                0780  2184 ;                 R2      New LPD pointer or zero if no new LPD needed
                                0780  2185 ;                         Invalid if R0 has low bit clear.
                                0780  2186 ;                 R0      Status
                                0780  2187 ;
                                0780  2188 ;-
                                0780  2189 BUILD_LPD:                                    ; Build LPD vector
    50    01    90              0780  2190          MOVB    #1,R0                       ; Assume no new LPD is needed
          52    D4              0783  2191          CLRL    R2                          ;
 55     0E A6    9A              0785  2192          MOVZBL  LNI$B_MLN(R6),R5            ; Get maximum lines (circuits) allowed
          55    D6              0789  2193          INCL    R5                          ; Add one for local LPD
 5C A8    55    91              078B  2194          CMPB    R5,RCB$B_MAX_LPD(R8)        ; Same as current maximum?
          39    13              078F  2195          BEQL    90$                         ; If so, then nothing to do
          2B    14              0791  2196          BGTR    10$                         ; Branch if increasing size
       28 A8    D5              0793  2197          TSTL    RCB$L_PTR_LPD(R8)           ; Is there a current LPD vector?
          26    13              0796  2198          BEQL    10$                         ; If none, then go ahead
    51    5C A8    9A           0798  2199          MOVZBL  RCB$B_MAX_LPD(R8),R1        ; Get current number of LPDs
 53       51    55    C3        079C  2200          SUBL3   R5,R1,R3                    ; Get number of LPDs to be removed
          28 B841    D5        07A0  2201 5$:       TSTL    @RCB$L_PTR_LPD(R8)[R1]      ; Is LPD slot in use?
             0E    18           07A4  2202          BGEQ    8$                          ; OK if slot not in use
 50    0000'8F    3C           07A6  2203          MOVZWL  #SS$_WRITLCK,R0             ; Error - cannot remove active slot
                   07AB  2204          $CNFFLD lni,l,mln,R9                ; Set qualifying parameter
             16    11           07B2  2205          BRB     90$                         ; Return with error
             51    D7           07B4  2206 8$:       DECL    R1                          ; Decrement LPD index
          E7 53    F5           07B6  2207          SOBGTR  R3,5$                       ; Loop thru all LPDs to be removed
       50    01    D0           07B9  2208          MOVL    #1,R0                       ; It's ok - current block can be used
             0C    11           07BC  2209          BRB     90$                         ; Exit with success
                   07BE  2210
       54    04    9A           07BE  2211 10$:      MOVZBL  #4,R4                       ; Specify cell size
          0102    30           07C1  2212          BSBW    COM_BLD_CO                  ; Allocate and init LPD
                   07C4  2213          ;
                   07C4  2214          ;
                   07C4  2215          ;   The following code is executed for each cell in the vector
                   07C4  2216          ;
                   07C4  2217          ;       R4  Cell address
                   07C4  2218          ;       R5  Cell index  - there is no cell with index zero
                   07C4  2219          ;
                   07C4  2220          ;
64  55    0100 8F    A1           07C4  2221          ADDW3   #^X<0100>,R5,(R4)           ; Store current path index & seq. no
```

```
05  07CA  2222 90$:     RSB
```

```
                                07CB  2224          .SBTTL  BUILD_ADJ - Build the ADJ vector
                                07CB  2225 ;+
                                07CB  2226 ; BUILD_ADJ     - Build the ADJ vector
                                07CB  2227 ;
                                07CB  2228 ; FUNCTIONAL DESCRIPTION:
                                07CB  2229 ;
                                07CB  2230 ; The maximum allowed adjacencies can be modified only if the there is no
                                07CB  2231 ; current ADJ vector.  This restriction may be relaxed in future releases.
                                07CB  2232 ;
                                07CB  2233 ; The ADJ vector is allocated and intialized.
                                07CB  2234 ;
                                07CB  2235 ; INPUTS:        R10     Ptr to new LNI CNF
                                07CB  2236 ;                R8      Ptr to RCB
                                07CB  2237 ;                R6      Ptr to start of non-pageable new LNI image
                                07CB  2238 ;                R5-R0   Scratch
                                07CB  2239 ;
                                07CB  2240 ; OUTPUTS:       R11,R10 Preserved
                                07CB  2241 ;                R9      LNI bit i.d. used to qualify any errors
                                07CB  2242 ;                R8-R6   Preserved
                                07CB  2243 ;                R2      New LPD pointer or zero if no new LPD needed
                                07CB  2244 ;                        Invalid if R0 has low bit clear.
                                07CB  2245 ;                R0      Status
                                07CB  2246 ;
                                07CB  2247 ;-
                                07CB  2248 BUILD_ADJ:                                   ; Build ADJ vector
     50    01    90             07CB  2249          MOVB    #1,R0                      ; Assume no new structure is needed
           52    D4             07CE  2250          CLRL    R2                         ;   "   "   "   "   "   "
     55    0E A6    9A          07D0  2251          MOVZBL  LNI$B_MLN(R6),R5           ; Get maximum circuits allowed
           55    D6             07D4  2252          INCL    R5                         ; Add one for local LPD
     55    28 A6    A0          07D6  2253          ADDW    LNI$W_MBR(R6),R5           ; Add number of broadcast routers
     55    26 A6    A0          07DA  2254          ADDW    LNI$W_MBE(R6),R5           ; Add number of broadcast endnodes
     68 A8    55    B1          07DE  2255          CMPW    R5,RCB$W_MAX_ADJ(R8)       ; Same as current maximum?
              6B    13          07E2  2256          BEQL    90$                        ; If so, then nothing to do
              2E    14          07E4  2257          BGTR    10$                        ; Branch if increasing size
        2C A8       D5          07E6  2258          TSTL    RCB$L_PTR_ADJ(R8)          ; Is there a current ADJ vector?
              29    13          07E9  2259          BEQL    10$                        ; If none, then go ahead
     51    68 A8    3C          07EB  2260          MOVZWL  RCB$W_MAX_ADJ(R8),R1       ; Get current number of ADJs
53   51    55       C3          07EF  2261          SUBL3   R5,R1,R3                   ; Get number of ADJs to be removed
50   2C B841       D0          07F3  2262 5$:       MOVL    @RCB$L_PTR_ADJ(R8)[R1],R0  ; Get ADJ address
  0E 60    00       E1          07F8  2263          BBC     #ADJ$V_INUSE,ADJ$B_STS(R0),8$ ; OK if slot not in use
50   0000'8F       3C          07FC  2264          MOVZWL  #SS$_WRITLCK,R0            ; Error - cannot remove active slot
                               0801  2265          $CNFFLD lni,f,mbr,R9               ; Set qualifying parameter
              45    11          0808  2266          BRB     90$                        ; Return with error
              51    D7          080A  2267 8$:       DECL    R1                         ; Decrement ADJ index
        E4 53       F5          080C  2268          SOBGTR  R3,5$                      ; Loop thru all ADJs to be removed
     50    01       D0          080F  2269          MOVL    #1,R0                      ; It's ok - current block can be used
              3B    11          0812  2270          BRB     90$                        ; Exit with success
                               0814  2271
     54    0D       9A          0814  2272 10$:      MOVZBL  #ADJ$C_LENGTH,R4           ; Specify cell size
51   54    55       C5          0817  2273          MULL3   R5,R4,R1                   ; Get total vector size
50   04    55       C5          081B  2274          MULL3   R5,#4,R0                   ; Compute size of pointer vector
51   0C A140       9E          081F  2275          MOVAB   12(R1)[R0],R1              ; Add in VMS header + pointer vector
           51       DD          0824  2276          PUSHL   R1                         ; Save it
        F7D7'       30          0826  2277          BSBW    NET$ALONPGD_Z              ; Allocate and zero the block
           51 8ED0              0829  2278          POPL    R1                         ; Restore R1
        20 50       E9          082C  2279          BLBC    R0,90$                     ; Br on error
     62    54       D0          082F  2280          MOVL    R4,(R2)                    ; Save size of each cell
```

```
                                                E 3
                     04 A2    55   D0  0832  2281          MOVL    R5,4(R2)                ; Save number of adjacencies in vector
                  53    0C A245   DE  0836  2282          MOVAL   12(R2)[R5],R3           ; Point to last pointer + 1
                     50    6241   9E  083B  2283          MOVAB   (R2)[R1],R0             ; Get ptr to first byte past last cell
                              55   D5  083F  2284          TSTL    R5                      ; Any cells ?
                              09   13  0841  2285          BEQL    40$                     ; If so, continue
                           50    54   C2  0843  2286 30$:   SUBL    R4,R0                   ; Skip to next block
                           73    50   D0  0846  2287          MOVL    R0,-(R3)                ; Initialize pointer to actual block
                              F7 55   F5  0849  2288          SOBGTR  R5,30$                  ; Loop for each cell
                           50    01   D0  084C  2289 40$:   MOVL    #1,R0                   ; Indicate success
                                      05  084F  2290 90$:   RSB                             ; Return to his caller
```

```
                     0850  2292          .SBTTL  BUILD_NDC - Build the Node counter vector
                     0850  2293 ;+
                     0850  2294 ; BUILD_NDC     - Build the Node counter vector
                     0850  2295 ;
                     0850  2296 ; FUNCTIONAL DESCRIPTION:
                     0850  2297 ;
                     0850  2298 ; If the maximum node address is being increased then a new vector must be
                     0850  2299 ; allocated.
                     0850  2300 ;
                     0850  2301 ; INPUTS:        R10     Ptr to new LNI CNF
                     0850  2302 ;                 R8      Ptr to RCB
                     0850  2303 ;                 R6      Ptr to start of non-pageable new LNI image
                     0850  2304 ;                 R5-R0   Scratch
                     0850  2305 ;
                     0850  2306 ; OUTPUTS:       R11,R10 Preserved
                     0850  2307 ;                 R9      LNI field bit i.d. used to qualify any errors
                     0850  2308 ;                 R8-R6   Preserved
                     0850  2309 ;                 R2      New NDC vector pointer or zero if new one isn't needed
                     0850  2310 ;                         Invalid if R0 has low bit clear.
                     0850  2311 ;                 R0      Status
                     0850  2312 ;
                     0850  2313 BUILD_NDC:                                   ; Build node counter vector
55   06 A6    3C     0850  2314          MOVZWL  LNI$W_MAD(R6),R5            ; Get new max address
   54   1C    D0     0854  2315          MOVL    #NDC$C_LENGTH,R4            ; Get size of each cell
   50   01    90     0857  2316          MOVB    #1,R0                       ; Assume new NDC vector is not needed
        52    D4     085A  2317          CLRL    R2                          ;  "     "   "   "  "   "   "
51   34 A8    D0     085C  2318          MOVL    RCB$L_PTR_NDC(R8),R1        ; Get current NDC vector
        06    13     0860  2319          BEQL    10$                         ; If EQL then none
55   5A A8    B1     0862  2320          CMPW    RCB$W_MAX_ADDR(R8),R5       ; Can this NDC vector be used?
        0C    1E     0866  2321          BGEQU   20$                         ; If GEQU then yes
        55    D6     0868  2322 10$:     INCL    R5                          ; Account for cell with index 0
      0059    30     086A  2323          BSBW    COM_BLD_CO                  ; Allocate and init the vector
                     086D  2324          ;
                     086D  2325          ;
                     086D  2326          ;   The following code is called once for each cell in the vector
                     086D  2327          ;   with:
                     086D  2328          ;
                     086D  2329          ;       R5  Cell index  - there is an index 0
                     086D  2330          ;       R4  Cell address
                     086D  2331          ;
                     086D  2332          ;
00000000'GF    D0    086D  2333          MOVL    G^EXE$GL_ABSTIM,-           ; Enter time last zeroed
          64         0873  2334                  NDC$L_ABS_TIM(R4)
                05   0874  2335 20$:     RSB
```

```
                     0875  2337          .SBTTL  BUILD_OA - Build Output Adjacency Vector
                     0875  2338 ;+
                     0875  2339 ; BUILD_OA      - Build Output Adjacency Vector
                     0875  2340 ;
                     0875  2341 ; FUNCTIONAL DESCRIPTION:
                     0875  2342 ;
                     0875  2343 ; The Output Adjacency Vector is used to determine the default adjacency to
                     0875  2344 ; be used to get to  a given node.  The index is the node address, the vector
                     0875  2345 ; cell contains the ADJ index.  The first vector cell corresponds to node
                     0875  2346 ; address 0.
                     0875  2347 ;
                     0875  2348 ; A new OA vector must be allocated whenever the maximum supported node address
                     0875  2349 ; is increased.
                     0875  2350 ;
                     0875  2351 ; INPUTS:        R10     Ptr to new LNI CNF
                     0875  2352 ;                 R8      Ptr to RCB
                     0875  2353 ;                 R6      Ptr to start of non-pageable new LNI image
                     0875  2354 ;                 R5-R0   Scratch
                     0875  2355 ;
                     0875  2356 ; OUTPUTS:       R11,R10 Preserved
                     0875  2357 ;                 R9      Bit i.d. used to qualify any errors
                     0875  2358 ;                 R8-R6   Preserved
                     0875  2359 ;                 R2      New LTB pointer or zero if no new LTB needed
                     0875  2360 ;                         Invalid if R0 has low bit clear.
                     0875  2361 ;                 R0      Status
                     0875  2362 ;
                     0875  2363 ;-
                     0875  2364 BUILD_OA:                                    ; Init Output Adjacency vector
          52   D4    0875  2365          CLRL    R2                         ; Assume no new vector is needed
     50   01   90    0877  2366          MOVB    #1,R0                      ;    "   "  "   "   "  "   "
55   06 A6   3C    087A  2367          MOVZWL  LNI$W_MAD(R6),R5           ; Get number of vector cells required
          55   D6    087E  2368          INCL    R5                         ; Add one for OA(0)
     54   02   D0    0880  2369          MOVL    #2,R4                      ; Indicate cell size
51   1C A8   D0    0883  2370          MOVL    RCB$L_PTR_OA(R8),R1        ; Get current vector
          0D   13    0887  2371          BEQL    10$                        ; If EQL then none
55   5A A8   B1    0889  2372          CMPW    RCB$W_MAX_ADDR(R8),R5      ; Can this vector be used ?
          0A   1E    088D  2373          BGEQU   20$                        ; If GEQU then yes
               088F  2374          $CNFFLD lni,l,mad,R9                   ; Further errors would be due to not
               0896  2375                                                 ; enough memory left since 'max addr'
               0896  2376                                                 ; is too large
        002D   30    0896  2377 10$:     BSBW    COM_BLD_CO                 ; Call co-routine to allocate block
                     0899  2378 ;
                     0899  2379 ;
                     0899  2380 ;   The following code is executed for each vector cell with:
                     0899  2381 ;
                     0899  2382 ;       R5  Cell index  - there is no index 0
                     0899  2383 ;       R4  Cell address
                     0899  2384 ;
                     0899  2385 ;
                05   0899  2386 20$:     RSB                                ; There is no cell intialization
                     089A  2387                                             ; required.
```

```
                                        .SBTTL  BUILD_AOA - Build Area Output Adjacency Vector
                                ;+
                                ; BUILD_AOA     - Build Area Output Adjacency Vector
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; The Area Output Adjacency Vector is used to determine the default adjacency to
                                ; be used to get to a given area.  The index is the area address, the vector
                                ; cell contains the ADJ index.  The first vector cell corresponds to area
                                ; number 0.
                                ;
                                ; A new AOA vector must be allocated whenever the maximum supported area address
                                ; is increased.
                                ;
                                ; INPUTS:       R10     Ptr to new LNI CNF
                                ;               R8      Ptr to RCB
                                ;               R6      Ptr to start of non-pageable new LNI image
                                ;               R5-R0   Scratch
                                ;
                                ; OUTPUTS:      R11,R10 Preserved
                                ;               R9      Bit i.d. used to qualify any errors
                                ;               R8-R6   Preserved
                                ;               R2      New LTB pointer or zero if no new LTB needed
                                ;                       Invalid if R0 has low bit clear.
                                ;               R0      Status
                                ;
                                ;-
                                BUILD_AOA:                                      ; Init Area Output Adjacency vector
                        52  D4  CLRL    R2                                      ; Assume no new vector is needed
                    50  01  90  MOVB    #1,R0                                   ;  "  "  "  "  "  "  "
                03  03  A6  91  CMPB    LNI$B_ETY(R6),#ADJ$C_PTY_AREA ; Are we an area router?
                        20  12  BNEQ    20$                                     ; If not, don't create any AOA
                55  36  A6  9A  MOVZBL  LNI$B_MAR(R6),R5                       ; Get number of vector cells required
                        55  D6  INCL    R5                                      ; Add one for AOA(0)
                    54  02  D0  MOVL    #2,R4                                   ; Indicate cell size
                51  20  A8  D0  MOVL    RCB$L_PTR_AOA(R8),R1                    ; Get current vector
                        0E  13  BEQL    10$                                     ; If EQL then none
            55  008C  C8  91  CMPB    RCB$B_MAX_AREA(R8),R5                   ; Can this vector be used ?
                        0A  1E  BGEQU   20$                                     ; If GEQU then yes
                            08BB        $CNFFLD lni,l,mar,R9                    ; Further errors would be due to not
                            08C2                                                ; enough memory left since 'max area'
                            08C2                                                ; is too large
                    0001    30  08C2    10$:    BSBW    COM_BLD_CO              ; Call co-routine to allocate block
                            08C5                ;
                            08C5                ;
                            08C5                ;   The following code is executed for each vector cell with:
                            08C5                ;
                            08C5                ;       R5  Cell index  - there is no index 0
                            08C5                ;       R4  Cell address
                            08C5                ;
                            08C5                ;
                        05  08C5    20$:    RSB                                 ; There is no cell intialization
                            08C6                                                ; required.
```

```
                                                I  3
                     08C6  2443          .SBTTL  COM_BLD_CO - Common build vector co-routine
                     08C6  2444 ;+
                     08C6  2445 ; COM_BLD_CO    - Common build vector co-routine.
                     08C6  2446 ;
                     08C6  2447 ; FUNCTIONAL DESCRIPTION:
                     08C6  2448 ;
                     08C6  2449 ; This routine allocates a zeroed block of non-paged pool to be used as a
                     08C6  2450 ; vector of cells.  The calling routine is returned to for each cell in the
                     08C6  2451 ; vector.
                     08C6  2452 ;
                     08C6  2453 ; Note that this routine is not a co-routine in the true sense of the word.
                     08C6  2454 ; After the last cell is initialized, the return is to the caller's caller.
                     08C6  2455 ;
                     08C6  2456 ; INPUTS:        R5      Number of cells
                     08C6  2457 ;                 R4      Cell size
                     08C6  2458 ;
                     08C6  2459 ; CALL BACK:     R5      Cell index (assumes first index is zero)
                     08C6  2460 ;                 R4      Pointer to cell
                     08C6  2461 ;                 R3      Scratch
                     08C6  2462 ;                 R2      Vector address
                     08C6  2463 ;                 R1      Scratch
                     08C6  2464 ;                 R0      Scratch
                     08C6  2465 ;
                     08C6  2466 ; OUTPUTS:       R2      Vector address if successful
                     08C6  2467 ;                 R0      Status
                     08C6  2468 ;-
                     08C6  2469 COM_BLD_CO:                                  ; Common build vector co-routine
           51  54  55  C5  08C6  2470            MULL3   R5,R4,R1            ; Get total vetor size
               51  0C  C0  08CA  2471            ADDL    #12,R1              ; Add in standard VMS header
                   51  DD  08CD  2472            PUSHL   R1                  ; Save it
                   F72E'   30  08CF  2473            BSBW    NET$ALONPGD_Z       ; Allocate and zero the block
                   51 8ED0  08D2  2474            POPL    R1                  ; Restore R1
                1B 50  E9  08D5  2475            BLBC    R0,30$              ; Br on error
             62    54  3C  08D8  2476            MOVZWL  R4,(R2)             ; Save cell size
          04 A2    55  3C  08DB  2477            MOVZWL  R5,4(R2)            ; Save number of cells
           54  6241  9E  08DF  2478            MOVAB   (R2)[R1],R4         ; Get ptr to first byte past last cell
                55  D5  08E3  2479            TSTL    R5                  ; Any cells ?
                09  13  08E5  2480            BEQL    20$                 ; If so, continue
             54 62  C2  08E7  2481 10$:        SUBL    (R2),R4             ; Advance to top of cell
                00 BE  16  08EA  2482            JSB     @(SP)               ; Call back caller - note not (SP)+
                F7 55  F5  08ED  2483            SOBGTR  R5,10$              ; Loop for each cell
             50    01  D0  08F0  2484 20$:        MOVL    #1,R0               ; Indicate success
                   8E  D5  08F3  2485 30$:        TSTL    (SP)+               ; Pop caller's address
                       05  08F5  2486            RSB                         ; Return to his caller
                           08F6  2487
                           08F6  2488 .END    NET$INITIALIZE
```

```
$$T1                    =  00000001
$$_NSPMSG               =  00000000
$$_TR3MSG               =  00000000
$$_TR4MSG               =  00000000
ACP$AL_ACTION              00000060 RG     02
ACP$AW_STA_TAB             00000000 RG     02
ACP$C_EVENTS            =  00000008
ACP$C_EVT_BUG           =  00000007
ACP$C_EVT_NOP           =  00000000
ACP$C_LPD_LOC           =  00000006
ACP$C_MAX_EVT           =  00000007
ACP$C_OPR_INIT          =  00000001
ACP$C_OPR_OFF           =  00000005
ACP$C_OPR_ON            =  00000002
ACP$C_OPR_RSTR          =  00000003
ACP$C_OPR_SHUT          =  00000004
ACP$C_STATES            =  00000006
ACP$C_STA_.             =  00000005
ACP$C_STA_F             =  00000004
ACP$C_STA_H             =  00000005
ACP$C_STA_I             =  00000000
ACP$C_STA_N             =  00000001
ACP$C_STA_R             =  00000002
ACP$C_STA_S             =  00000003
ACP_L_EVT                  00000000 R      03
ACT_BRDCST                 00000316 R      07
ACT_BUG                    00000340 R      07
ACT_CLEANUP                0000031D R      07
ACT_ERROR                  00000339 R      07
ACT_NODE_OFF               0000031A R      07
ACT_NODE_SHUT              00000313 R      07
ACT_NOP                    00000333 R      07
ACT_REVIVE                 0000020E R      08
ACT_STALL                  00000340 R      07
ADJ                     =  00000010
ADJ$B_LPD_INX           =  00000002
ADJ$B_PTYPE             =  00000001
ADJ$B_STS               =  00000000
ADJ$C_LENGTH            =  0000000D
ADJ$C_PTY_AREA          =  00000003
ADJ$C_PTY_PH3           =  00000000
ADJ$C_PTY_PH4           =  00000004
ADJ$C_PTY_PH4N          =  00000005
ADJ$M_INUSE             =  00000001
ADJ$M_RUN               =  00000002
ADJ$V_INUSE             =  00000000
ADJ$V_RTG               =  00000002
ADJ$V_RUN               =  00000001
ADJ$W_BUFSIZ            =  00000006
ADJ$W_PNA               =  00000004
AOA                     =  00000014
AQB$B_ACPTYPE           =  00000015
AQB$B_MNTCNT            =  0000000B
AQB$B_STATUS            =  00000014
AQB$B_TYPE              =  0000000A
AQB$C_LENGTH            =  0000001C
AQB$C_XLNG              =  00000020
AQB$K_NET               =  00000004
AQB$L_ACPPID            =  0000000C
AQB$L_ACPQBL            =  00000004
AQB$L_ACPQFL            =  00000000
AQB$L_LINK              =  00000010
AQB$M_UNIQUE            =  00000001
AQB$W_SIZE              =  00000008
BEGLCR                     00000000 R      05
BIT...                  =  00000006
BUG$_NETNOSTATE            ********   X    07
BUG$_NETSYSSRV             ********   X    07
BUILD_ADJ                  000007CB R      07
BUILD_AOA                  0000089A R      07
BUILD_LPD                  00000780 R      07
BUILD_LTB                  0000072D R      07
BUILD_NDC                  00000850 R      07
BUILD_OA                   00000875 R      07
CALL_NETDRIVER             0000015D RG     07
CCB$L_UCB               =  00000000
CNF$C_LENGTH            =  00000024
CNF$GET_FIELD              ********   X    07
CNF$PUT_FIELD              ********   X    07
CNF$_ADVANCE            =  00000000
CNF$_QUIT               =  00000002
CNF$_TAKE_CURR          =  00000003
CNF$_TAKE_PREV          =  00000001
CNR$L_FLINK             =  00000000
COM_BLD_CO                 000008C6 R      07
CRB$L_INTD              =  00000024
CTL$GL_PCB                 ********   X    07
CTL$T_USERNAME             ********   X    07
CURRENT_SAD                00000026 R      03
CXB$C_OVERHEAD          =  0000004C
DDB$L_UCB               =  00000004
DEV$M_DMT               =  00200000
DEV$M_MNT               =  00080000
DEV$V_DMT               =  00000015
DEV$V_MNT               =  00000013
DISMOUNT                   00000130 R      08
DLE_DESC                   000000AB R      02
DYN$C_AQB               =  00000003
DYN$C_TQE               =  0000000F
DYN$C_VCB               =  00000011
ENDLCR                     00000000 R      06
EVC$C_SCL_LNS           =  00000080
EVC$C_SCL_PRSN_NOR      =  00000001
EVC$C_SCL_PRSN_OPC      =  00000000
EVL_STA_MAP                00000090 R      02
EXE$GL_ABSTIM              ********     X  08
INIT_AQB                   0000003F R      08
INIT_LPD                   000000BE R      08
INIT_NETDRIVER             00000146 R      07
INIT_RCB                   0000006F R      08
INIT_TQE                   000000E0 R      08
IO$_ACPCONTROL             ********     X  07
IOC$GL_AQBLIST             ********     X  08
IOC$GL_MUTEX               ********     X  07
```

```
IOC$VERIFYCHAN          ********  X   07
IOSB                    0000001E R      03
IPL$_SYNCH              =  00000008
JIB$T_USERNAME          =  0000000C
LKWSET_ADDR             0000010D R      02
LNG                     =  00000170
LNI$B_DAC               =  0000002D
LNI$B_DFA               =  0000002A
LNI$B_DPX               =  0000002E
LNI$B_DWE               =  0000002B
LNI$B_ETY               =  00000003
LNI$B_MAR               =  00000036
LNI$B_MHO               =  0000000C
LNI$B_MLN               =  0000000E
LNI$B_MVI               =  0000000D
LNI$B_RFA               =  0000002C
LNI$B_STA               =  00000002
LNI$C_LENGTH            =  00000050
LNI$C_STA_INIT          =  00000004
LNI$C_STA_OFF           =  00000001
LNI$C_STA_ON            =  00000000
LNI$C_STA_RSTR          =  00000003
LNI$C_STA_SHUT          =  00000002
LNI$W_ADD               =  00000000
LNI$W_ALI               =  0000003A
LNI$W_BUS               =  00000016
LNI$W_IAT               =  0000001C
LNI$W_ITI               =  0000001E
LNI$W_MAD               =  00000006
LNI$W_MBE               =  00000026
LNI$W_MBR               =  00000028
LNI$W_MBU               =  00000008
LNI$W_MLK               =  00000004
LNI$W_OTI               =  00000020
LNI$W_PIQ               =  0000002F
LNI$W_SBS               =  00000018
LOCK_IODB               000001B0 RG     07
LOG$C_PROCESS           =  00000002
LPD                     =  00000008
LPD$B_PTH_INX           =  00000020
LPD$B_XMT_IPL           =  0000001F
LPD$B_XMT_SRL           =  0000001E
LPD$C_LENGTH            =  0000006A
LPD$C_LOC_INX           =  00000001
LPD$C_XLNG              =  00000070
LPD$M_RBF               =  00000040
LPD$M_RUN               =  00000010
LPD$M_XBF               =  00000020
LPD$Q_REQ_WAIT          =  00000000
LPD$W_STS               =  00000022
LTB                     =  00000000
LTB$L_SLOTS             =  00000010
LTB$L_SLT_NXT           =  00000000
LTB$L_XWB               =  0000000C
LTB$W_SLT_LMT           =  00000006
LTB$W_SLT_TOT           =  00000004
MBX_NET_SRUT            00000344 R      07
MOUNT                   00000000 R      08
MSG$_NETSHUT            =  0000003B
NDB$C_MSG_SHUT          =  00000002
NDB$C_MSG_START         =  00000001
NDC                     =  00000004
NDC$C_LENGTH            =  0000001C
NDC$L_ABS_TIM           =  00000000
NET$ALONPAGED             ********    X   07
NET$ALONPGD_Z             ********    X   08
NET$CREATE_MBX            ********    X   07
NET$C_ACT_TIMER         =  0000001E
NET$C_EFN_ASYN          =  00000002
NET$C_EFN_WAIT          =  00000001
NET$C_IPL               =  00000008
NET$C_MAXACCFLD         =  00000027
NET$C_MAXLINNAM         =  0000000F
NET$C_MAXLNK            =  000003FF
NET$C_MAXNODNAM         =  00000006
NET$C_MAXOBJNAM         =  0000000C
NET$C_MAX_AREAS         =  0000003F
NET$C_MAX_LINES         =  00000040
NET$C_MAX_NCB           =  0000006E
NET$C_MAX_NODES         =  000003FF
NET$C_MAX_OBJ           =  000000FF
NET$C_MAX_WQE           =  00000014
NET$C_MINBUFSIZ         =  000000C0
NET$C_TID_ACT           =  00000003
NET$C_TID_RUS           =  00000001
NET$C_TID_XRT           =  00000002
NET$C_TRCTL_CEL         =  00000002
NET$C_TRCTL_OVR         =  00000005
NET$C_UTLBUFSIZ         =  00001000
NET$DEALLOCATE            ********      X   08
NET$DECLARE_PSI           00000646 RG     07
NET$DECR_MCOUNT           00000225 RG     08
NET$DEC_TRANS             00000175 RG     07
NET$DISPATCH              ********      X   07
NET$DLLUPDLNI             ********      X   07
NET$DLL_ALL_OFF           ********      X   07
NET$EVT_INTRAW            ********      X   07
NET$GET_X25_CHAN          ********      X   07
NET$GL_CNR_PLI            ********      X   07
NET$GL_DLE_UCB            00000006 RG     03
NET$GL_DLE_UCBO           0000000A RG     03
NET$GL_LOC_LPD            00000000 R      04
NET$GL_MY_POOL            0000000E RG     03
NET$GL_NET_UCB            ********      X   07
NET$GL_PTR_AQB            ********      X   08
NET$GL_PTR_LNI            ********      X   07
NET$GL_PTR_UCBO           ********      X   07
NET$GL_PTR_VCB            ********      X   08
NET$GQ_MBX_NAME           ********      X   07
NET$GQ_VERSION            ********      X   07
NET$GQ_X25_DEV            ********      X   07
NET$GW_DLECHAN            00000004 R      03
NET$GW_NETCHAN            ********      X   07
NET$GW_X25_CHAN           ********      X   07
```

```
NET$INITIALIZE                  00000000 RG    07
NET$INIT_ROUTING                ******** X     07
NET$INI_CONFIG                  ******** X     07
NET$JNX_CO                      ******** X     07
NET$KILL_MBX                    ******** X     08
NET$LOCLPD_DOWN                 00000228 RG    07
NET$MBX_QIO                     ******** X     07
NET$M_MAXLNKMSK               = 000003FF
NET$OPCOM                       ******** X     07
NET$START_TIMER                 000001E1 R     07
NET$UPD_LOCAL                   00000359 RG    07
NETUPD$_ABOLNK                = 00000008
NETUPD$_BRDCST                = 0000000A
NETUPD$_DLL_ON                = 00000005
NET_DESC                        0000009E R     02
NET_NAME                        000000B7 R     02
NEW_LNI_IMAGE                   00000004 R     04
NFB$C_DECLNAME                = 00000015
NFB$C_LNI_ADD                 = 01010010
NFB$C_LNI_ALI                 = 01010033
NFB$C_LNI_BUS                 = 01010024
NFB$C_LNI_DAC                 = 01010028
NFB$C_LNI_DFA                 = 01010016
NFB$C_LNI_DPX                 = 01010029
NFB$C_LNI_ETY                 = 0101001A
NFB$C_LNI_MAD                 = 0101001E
NFB$C_LNI_MAR                 = 0101002D
NFB$C_LNI_MBR                 = 0101002F
NFB$C_LNI_MLK                 = 01010015
NFB$C_LNI_MLN                 = 0101001F
NFB$C_LNI_MVI                 = 01010022
NFB$C_LNI_PIQ                 = 0101002A
NFB$C_LNI_SAD                 = 0101001D
NFB$C_LNI_SBS                 = 01010032
NFB$C_LNI_STA                 = 01010014
NMA$C_ACES_BOTH               = 00000003
NMA$C_ACES_NONE               = 00000000
NMA$C_STATE_OFF               = 00000001
NMA$C_STATE_ON                = 00000000
NMA$C_STATE_RES               = 00000003
NMA$C_STATE_SHU               = 00000002
NODE_DESC                       000000BB R     02
NSP$$$_QUAL_ACK               = 00000000
NSP$$$_QUAL_ALTFLW            = 00000000
NSP$$$_QUAL_DATA              = 00000000
NSP$$$_QUAL_FLW               = 00000000
NSP$$$_QUAL_INF               = 00000000
NSP$$$_QUAL_MSG               = 00000000
NSP$$$_QUAL_SRV               = 00000000
NSP$C_EXT_LNK                 = 0000001E
NSP$C_FLW_DATA                = 00000000
NSP$C_FLW_INT                 = 00000001
NSP$C_FLW_NOP                 = 00000000
NSP$C_FLW_XOFF                = 00000001
NSP$C_FLW_XON                 = 00000002
NSP$C_HSZ_ACK                 = 00000007
NSP$C_HSZ_CA                  = 00000003
NSP$C_HSZ_CC                  = 00000064
NSP$C_HSZ_CD                  = 000000F0
NSP$C_HSZ_CI                  = 000000F0
NSP$C_HSZ_DATA                = 00000009
NSP$C_HSZ_DC                  = 00000016
NSP$C_HSZ_DI                  = 00000016
NSP$C_HSZ_INT                 = 00000009
NSP$C_HSZ_LS                  = 00000009
NSP$C_INF_V31                 = 00000001
NSP$C_INF_V32                 = 00000000
NSP$C_INF_V33                 = 00000002
NSP$C_MAXADR                  = 00000009
NSP$C_MSG_CA                  = 00000024
NSP$C_MSG_CC                  = 00000028
NSP$C_MSG_CI                  = 00000018
NSP$C_MSG_DATA                = 00000000
NSP$C_MSG_DC                  = 00000048
NSP$C_MSG_DI                  = 00000038
NSP$C_MSG_DTACK               = 00000004
NSP$C_MSG_INT                 = 00000030
NSP$C_MSG_LIACK               = 00000014
NSP$C_MSG_LS                  = 00000010
NSP$C_SRV_MFC                 = 00000002
NSP$C_SRV_NFC                 = 00000000
NSP$C_SRV_REQ                 = 00000001
NSP$C_SRV_SFC                 = 00000001
NSP$M_ACK_NAK                 = 00001000
NSP$M_ACK_NUM                 = 00000FFF
NSP$M_ACK_VALID               = 00008000
NSP$M_DATA_BOM                = 00000020
NSP$M_DATA_EOM                = 00000040
NSP$M_DATA_OVFW               = 00000080
NSP$M_FLW_CHAN                = 0000000C
NSP$M_FLW_DRV                 = 000000F0
NSP$M_FLW_INT                 = 00000020
NSP$M_FLW_INUSE               = 00000010
NSP$M_FLW_LISUB               = 00000004
NSP$M_FLW_MODE                = 00000003
NSP$M_FLW_SP1                 = 00000008
NSP$M_FLW_SP2                 = 00000040
NSP$M_FLW_SP3                 = 00000080
NSP$M_FLW_XOFF                = 00000001
NSP$M_FLW_XON                 = 00000002
NSP$M_INF_VER                 = 00000003
NSP$M_MSG_INT                 = 00000020
NSP$M_MSG_LI                  = 00000010
NSP$M_SRV_01                  = 00000003
NSP$M_SRV_EXT                 = 00000080
NSP$M_SRV_FLW                 = 0000000C
NSP$M_SRV_REQ                 = 000000F3
NSP$M_SRV_SP1                 = 00000070
NSP$R_QUAL                    = 00000000
NSP$S_ACK_NUM                 = 0000000C
NSP$S_ACK_SP2                 = 00000002
NSP$S_DATA_SP                 = 00000005
NSP$S_FLW_CHAN                = 00000002
NSP$S_FLW_DRV                 = 00000004
```

```
NSP$S_FLW_MODE          =  00000002
NSP$S_INF_VER           =  00000002
NSP$S_MSG_SP1           =  00000004
NSP$S_NSPMSG            =  00000005
NSP$S_QUAL              =  00000005
NSP$S_QUAL_ACK          =  00000002
NSP$S_QUAL_ALTFLW       =  00000001
NSP$S_QUAL_DATA         =  00000001
NSP$S_QUAL_FLW          =  00000001
NSP$S_QUAL_INF          =  00000001
NSP$S_QUAL_MSG          =  00000005
NSP$S_QUAL_SRV          =  00000001
NSP$S_SRV_01            =  00000002
NSP$S_SRV_FLW           =  00000002
NSP$S_SRV_SP1           =  00000003
NSP$V_ACK_NAK           =  0000000C
NSP$V_ACK_NUM           =  00000000
NSP$V_ACK_SP2           =  0000000D
NSP$V_ACK_VALID         =  0000000F
NSP$V_DATA_BOM          =  00000005
NSP$V_DATA_EOM          =  00000006
NSP$V_DATA_OVFW         =  00000007
NSP$V_DATA_SP           =  00000000
NSP$V_FLW_CHAN          =  00000002
NSP$V_FLW_DRV           =  00000004
NSP$V_FLW_INT           =  00000005
NSP$V_FLW_INUSE         =  00000004
NSP$V_FLW_LISUB         =  00000002
NSP$V_FLW_MODE          =  00000000
NSP$V_FLW_SP1           =  00000003
NSP$V_FLW_SP2           =  00000006
NSP$V_FLW_SP3           =  00000007
NSP$V_FLW_XOFF          =  00000000
NSP$V_FLW_XON           =  00000001
NSP$V_INF_VER           =  00000000
NSP$V_MSG_INT           =  00000005
NSP$V_MSG_LI            =  00000004
NSP$V_MSG_SP1           =  00000000
NSP$V_SRV_01            =  00000000
NSP$V_SRV_EXT           =  00000007
NSP$V_SRV_FLW           =  00000002
NSP$V_SRV_SP1           =  00000004
NSP$W_DSTLNK            =  00000001
NSP$W_SRCLNK            =  00000003
NUM_LINES               =  00000041
NUM_NODES               =  00000400
OA                      =  0000000C
OPR_EVT_MAP               00000084 R      02
PCB$L_JIB               =  00000080
PCB$L_PHD               =  0000006C
PCB$L_PID               =  00000060
PCB$L_UIC               =  000000BC
PHD$Q_PRIVMSK           =  00000000
POOL_LENGTH             =  000186A0
PR$_IPL                    ********   X    08
PRIVMSK                    000000CB R      02
PROC_EVT                   00000236 R      07
PSI$C_NTD_ACCLVL        =  00000001
PSI$C_NTD_SAHI          =  00000003
PSI$C_NTD_SALO          =  00000002
PSI_DECL_CHAN              0000002A R      03
RCB$B_ACT_TIMER         =  0000008F
RCB$B_ECL_DAC           =  00000066
RCB$B_ECL_DFA           =  00000064
RCB$B_ECL_DPX           =  00000067
RCB$B_ECL_DWE           =  00000065
RCB$B_ECL_RFA           =  00000063
RCB$B_ECL_RFLW          =  00000062
RCB$B_ETY               =  0000008A
RCB$B_HOMEAREA          =  0000008B
RCB$B_MAX_AREA          =  0000008C
RCB$B_MAX_LPD           =  0000005C
RCB$B_MAX_VISIT         =  0000005E
RCB$B_STI               =  00000061
RCB$B_TYPE              =  0000000A
RCB$C_LENGTH            =  000000AE
RCB$C_XLNG              =  000000B0
RCB$L_ABS_TIM           =  00000090
RCB$L_ACP_UCB           =  00000014
RCB$L_AQB               =  00000010
RCB$L_PTR_ADJ           =  0000002C
RCB$L_PTR_AOA           =  00000020
RCB$L_PTR_LPD           =  00000028
RCB$L_PTR_LTB           =  00000024
RCB$L_PTR_NDC           =  00000034
RCB$L_PTR_OA            =  0000001C
RCB$L_PTR_TQE           =  00000030
RCB$Q_CXB_FREE          =  000000A0
RCB$Q_IRP_FREE          =  00000000
RCB$Q_IRP_WAIT          =  0000004C
RCB$Q_LOC_RCV           =  0000003C
RCB$Q_LOC_XMT           =  00000044
RCB$W_ADDR              =  0000000E
RCB$W_ALIAS             =  0000008D
RCB$W_CUR_PKT           =  00000080
RCB$W_ECLSEGSIZ         =  0000007C
RCB$W_MAX_ADDR          =  0000005A
RCB$W_MAX_ADJ           =  00000068
RCB$W_MAX_LNK           =  00000058
RCB$W_MAX_PKT           =  00000082
RCB$W_MAX_RTG           =  0000006A
RCB$W_MCOUNT            =  00000054
RCB$W_TIM_CNI           =  00000076
RCB$W_TIM_CNO           =  00000078
RCB$W_TIM_IAT           =  00000074
RCB$W_TOTBUFSIZ         =  0000007E
RCB$W_TRANS             =  0000000C
REPLACE                    000002BC R      08
RETURN_NULL                00000336 R      07
SAVE_STA_TAB               0000001A R      03
SCH$LOCKQ                  ********     X  07
SCH$UNLOCK                 ********     X  07
SIZ...                  =  00000001
SS$_BADPARAM               ********     X  07
```

```
SS$_DEVMOUNT                    ********   X    08
SS$_NORMAL                      ********   X    07
SS$_NOSUCHDEV                   ********   X    07
SS$_WRITLCK                     ********   X    07
STARTUP                         00000041 R         07
SYS$ASSIGN                      ********  GX    07
SYS$CANTIM                      ********  GX    08
SYS$CMKRNL                      ********  GX    07
SYS$CRELOG                      ********  GX    07
SYS$DASSGN                      ********  GX    08
SYS$DELLOG                      ********  GX    08
SYS$DELPRC                      ********  GX    07
SYS$EXPREG                      ********  GX    07
SYS$GQ_VERSION                  ********   X    07
SYS$LKWSET                      ********  GX    07
SYS$QIOW                        ********  GX    07
SYS$SETDDIR                     ********   X    07
SYSDISK                         000000D3 R         02
SYSMGR                          000000F7 R         02
SYSTEM_ROOT                     000000E3 R         02
TQE$B_TYPE                    = 0000000A
TQE$C_LENGTH                  = 00000030
TQE$C_XLNG                    = 00000030
TR$C_MAXHDR                   = 0000001C
TR$C_NI_ALLEND1               = 040000AB
TR$C_NI_ALLEND2               = 00000000
TR$C_NI_ALLROU1               = 030000AB
TR$C_NI_ALLROU2               = 00000000
TR$C_NI_PREFIX                = 000400AA
TR$C_NI_PROT                  = 00000360
TR$C_PRI_ECL                  = 0000001F
TR$C_PRI_RTHRU                = 0000001F
TR3$$_QUAL_MSG                = 00000000
TR3$$$_QUAL_RTFLG             = 00000000
TR3$C_ASZ_DATA                = 00000006
TR3$C_MSG_DATA                = 00000002
TR3$C_MSG_HELLO               = 00000005
TR3$C_MSG_INIT                = 00000001
TR3$C_MSG_NOP2                = 00000008
TR3$C_MSG_ROUT                = 00000007
TR3$C_MSG_STR2                = 00000058
TR3$C_MSG_VERF                = 00000003
TR3$M_MSG_CTL                 = 00000001
TR3$M_MSG_RTH                 = 00000002
TR3$M_RTFLG_PH2               = 00000040
TR3$M_RTFLG_RQR               = 00000008
TR3$M_RTFLG_RTS               = 00000010
TR3$R_QUAL                    = 00000000
TR3$S_QUAL                    = 00000001
TR3$S_QUAL_MSG                = 00000001
TR3$S_QUAL_RTFLG              = 00000001
TR3$S_RTFLG_012               = 00000003
TR3$S_TR3MSG                  = 00000001
TR3$V_MSG_CTL                 = 00000000
TR3$V_MSG_RTH                 = 00000001
TR3$V_RTFLG_012               = 00000000
TR3$V_RTFLG_5                 = 00000005
TR3$V_RTFLG_7                 = 00000007
TR3$V_RTFLG_PH2               = 00000006
TR3$V_RTFLG_RQR               = 00000003
TR3$V_RTFLG_RTS               = 00000004
TR4$$$_QUAL_ADDR              = 00000000
TR4$$$_QUAL_RTFLG             = 00000000
TR4$$$_QUAL_SCLASS            = 00000000
TR4$C_BCE_MID1                = 040000AB
TR4$C_BCE_MID2                = 00000000
TR4$C_BCR_MID1                = 030000AB
TR4$C_BCR_MID2                = 00000000
TR4$C_BCT3MULT                = 00000008
TR4$C_END_NODE                = 00000003
TR4$C_HIORD                   = 000400AA
TR4$C_HSZ_DATA                = 00000015
TR4$C_MSG_BCEHEL              = 0000000D
TR4$C_MSG_BCRHEL              = 0000000B
TR4$C_MSG_LDATA               = 00000006
TR4$C_MSG_RDATA               = 00000002
TR4$C_PRO_TYPE                = 00000360
TR4$C_RTR_LVL1                = 00000002
TR4$C_RTR_LVL2                = 00000001
TR4$C_T3MULT                  = 00000002
TR4$C_VER_HIB                 = 00000000
TR4$C_VER_LOWW                = 00000002
TR4$M_ADDR_AREA               = 0000FC00
TR4$M_ADDR_DEST               = 000003FF
TR4$M_RTFLG_INI               = 00000020
TR4$M_RTFLG_LNG               = 00000004
TR4$M_RTFLG_RQR               = 00000008
TR4$M_RTFLG_RTS               = 00000010
TR4$R_QUAL                    = 00000000
TR4$S_ADDR_AREA               = 00000006
TR4$S_ADDR_DEST               = 0000000A
TR4$S_QUAL                    = 00000002
TR4$S_QUAL_ADDR               = 00000002
TR4$S_QUAL_RTFLG              = 00000001
TR4$S_QUAL_SCLASS             = 00000001
TR4$S_RTFLG_01                = 00000002
TR4$S_RTFLG_VER               = 00000002
TR4$S_SCLASS_57               = 00000003
TR4$S_TR4MSG                  = 00000002
TR4$V_ADDR_AREA               = 0000000A
TR4$V_ADDR_DEST               = 00000000
TR4$V_RTFLG_01                = 00000000
TR4$V_RTFLG_INI               = 00000005
TR4$V_RTFLG_LNG               = 00000002
TR4$V_RTFLG_RQR               = 00000003
TR4$V_RTFLG_RTS               = 00000004
TR4$V_RTFLG_VER               = 00000006
TR4$V_SCLASS_1                = 00000001
TR4$V_SCLASS_57               = 00000005
TR4$V_SCLASS_BC               = 00000004
TR4$V_SCLASS_LS               = 00000002
TR4$V_SCLASS_METR             = 00000000
TR4$V_SCLASS_SUBA             = 00000003
UCB$L_CRB                     = 00000024
```

```
UCB$L_DDB                       =  00000028
UCB$L_DEVCHAR                   =  00000038
UCB$L_LINK                      =  00000030
UCB$L_VCB                       =  00000034
UIC                             =  00010004
UNDECLARE_PSI                      00000718 R      07
UNLOCK_IODB                        000001C7 RG     07
UPDATE_DATABASE                    00000235 R      08
USERNAME                           00000107 R      02
USERNAMLEN                      =  00000006
VEC$L_START                     =  0000001C
WQE$ALLOCATE                       ********    X   07
WQE$B_EVL_DT1                   =  0000001E
WQE$B_EVL_DT2                   =  0000001F
WQE$CANCEL_TIM                     ********    X   07
WQE$C_QUAL_ACT                  =  00000004
WQE$C_SUB_ACP                   =  00000001
WQE$DEALLOCATE                     ********    X   07
WQE$L_EVL_PKT                   =  00000018
WQE$RESET_TIM                      ********    X   07
WQE$W_EVL_CODE                  =  0000001C
XMSK                            =  0000000F
_$$                             =  00000000
_$ENT                           =  00000001
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_PURE	00000115 (277.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
NET_IMPURE	0000002C (44.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
NET_LOCK_IMPURE	00000054 (84.)	04 (4.)	NOPIC USR CON REL GBL NOSHR EXE RD WRT NOVEC LONG
NET_LOCK_A	00000000 (0.)	05 (5.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG
NET_LOCK_Z	00000000 (0.)	06 (6.)	NOPIC USR CON REL GBL NOSHR EXE RD WRT NOVEC BYTE
NET_CODE	000008F6 (2294.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
NET_LOCK_CODE	00000434 (1076.)	08 (8.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.11	00:00:00.82
Command processing	189	00:00:01.24	00:00:04.23
Pass 1	1031	00:00:41.35	00:00:54.83
Symbol table sort	0	00:00:05.73	00:00:06.27
Pass 2	615	00:00:09.26	00:00:12.12
Symbol table output	0	00:00:00.48	00:00:00.96
Psect synopsis output	3	00:00:00.05	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1870	00:00:58.22	00:01:19.29

The working set limit was 2000 pages.
231306 bytes (452 pages) of virtual memory were used to buffer the intermediate code.
There were 210 pages of symbol table space allocated to hold 3848 non-local and 133 local symbols.
2488 source lines were read in Pass 1, producing 38 object records in Pass 2.
84 pages of virtual memory were used to define 73 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SHRLIB]EVCDEF.MLB;1	1
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	1
_255DUA28:[NETACP.OBJ]NET.MLB;1	18
_255DUA28:[SYS.OBJ]LIB.MLB;1	19
_255DUA28:[SYSLIB]STARLET.MLB;2	22
TOTALS (all libraries)	62

4158 GETS were required to define 62 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NETACPTRN/OBJ=OBJ$:NETACPTRN MSRC$:NETACPTRN/UPDATE=(ENH$:NETACPTRN)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$
```

NETUSR
SDL

LIBTAIL
B32

XWBDEF
SDL

NETDEFS
MAR

PSIUSR
SDL

NETDRVMAC
MAR

NODRIVER
LIS

NSPMSGDEF
SDL

LIBHEAD
B32

NET
LIS

NETMACROS
MAR

NETACPTRN
LIS

NETCNF
LIS

NETCNFACT
LIS

NETCNFDLL
LIS