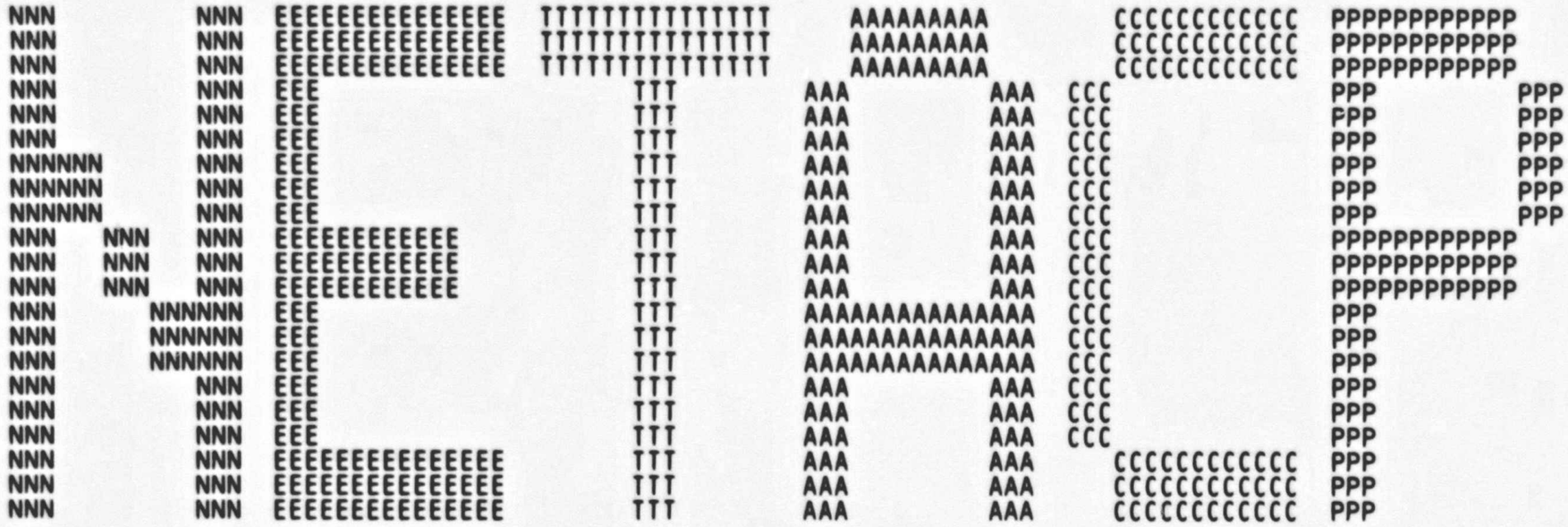
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN 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

FILEID**NETDRVXPT

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV  XX      XX  PPPPPPPP    TTTTTTTTTT
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV  XX      XX  PPPPPPPP    TTTTTTTTTT
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  XX      XX  PP      PP      TT
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  XX      XX  PP      PP      TT
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV    XX  XX    PP      PP      TT
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV    XX  XX    PP      PP      TT
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV      XX      PPPPPPPP        TT
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV      XX      PPPPPPPP        TT
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV    XX  XX    PP              TT
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV    XX  XX    PP              TT
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV      XX      XX  PP              TT      ....
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV      XX      XX  PP              TT      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV        XX      XX  PP              TT      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV        XX      XX  PP              TT      ....

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  NETDRVXPT - NETDRIVER Transport (Routing) Layer
0000     2          .IDENT  'V04-000'
0000     3
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:
0000    29 ;
0000    30 ;       VAX/VMS NETDRIVER
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       This module implements the DECnet Transport packet switching function.
0000    35 ;
0000    36 ; AUTHOR:
0000    37 ;
0000    38 ;       A.ELDRIDGE      1-May-82
0000    39 ;
0000    40 ; MODIFIED BY:
0000    41 ;
0000    42 ;       V03-039 RNG0039         Rod Gamache     24-Mar-1984
0000    43 ;               Enable check of ACP activity timer. Disable all transmit
0000    44 ;               operations if the NETACP process has stalled.
0000    45 ;
0000    46 ;       V03-038 PRB0318         Paul Beck       8-Mar-1984  18:19
0000    47 ;               Add TEST_ADJ to return true/false indication of whether a
0000    48 ;               node address represents a node which is one hop distant.
0000    49 ;
0000    50 ;       V03-037 RNG0037         Rod Gamache             02-Mar-1984
0000    51 ;               Disable check of ACP activity timer.
0000    52 ;
0000    53 ;       V03-036 ADE0001         Alan D. Eldridge        14-Feb-1984
0000    54 ;               Remove all trace of the "DLE" support.
0000    55 ;               Add count of entries added to AQB work queue.
0000    56 ;
0000    57 ;       V03-035 RNG0035         Rod Gamache     27-Jan-1984
```

```
0000     58 ;                  Fix problem with Transport not resetting the CXB type
0000     59 ;                  when system resources are being depleted.
0000     60 ;
0000     61 ;          V03-034 RNG0034         Rod Gamache     14-Nov-1983
0000     62 ;                  Fix problem in connecting a Phase IV endnode to a
0000     63 ;                  Phase III node, don't build a Phase IV route header
0000     64 ;                  on packets transmitted.
0000     65 ;                  Fix PSI problem that crashes system when the system
0000     66 ;                  resources (CXBs) are being depleted.
0000     67 ;
0000     68 ;          V03-033 RNG0033         Rod Gamache     11-Jul-1983
0000     69 ;                  Add support for cluster group address.
0000     70 ;
0000     71 ;          V03-032 TMH0032         Tim Halvorsen   08-Jun-1983
0000     72 ;                  Fix erroneous check which prevented reception of Phase II
0000     73 ;                  route headers (currently only known to be sent by DECnet-2020).
0000     74 ;                  Fix case where garbaged message which looks like a data msg
0000     75 ;                  is received on a point-to-point circuit which hasn't yet been
0000     76 ;                  node inited.  We were assuming that the ADJ was valid and
0000     77 ;                  crashing when referencing the ADJ block.
0000     78 ;
0000     79 ;          V03-031 RNG0031         Rod Gamache     01-Jun-1983
0000     80 ;                  Fix solicit to PH3N, which was preventing any logical links
0000     81 ;                  to an adjacent PH3N node.
0000     82 ;
0000     83 ;          V03-030 TMH0030         Tim Halvorsen   26-May-1983
0000     84 ;                  Fix setting of Intra-NI flag.  We were always setting
0000     85 ;                  the flag, even in the route-thru case, which told endnodes
0000     86 ;                  that nodes were on the NI, even when they weren't,
0000     87 ;                  and causing connectivity problems.
0000     88 ;                  Replace code which sets the Intra-NI flag 0/1 by figuring
0000     89 ;                  out who the source and destination are.  The replaced code
0000     90 ;                  uses a simple test of input=output LPD to clear the intra-NI
0000     91 ;                  flag and assumes that all other nodes originate their NI
0000     92 ;                  packets with the flag set.  (This code was written in the
0000     93 ;                  previous modification, but left commented out).
0000     94 ;
0000     95 ;          V03-029 RNG0029         Rod Gamache     05-May-1983
0000     96 ;                  Only enter node addresses into the CACHE which are
0000     97 ;                  received with the Inta-Ethernet bit set. Remove all
0000     98 ;                  settings of the Intra-Ethernet bit (NEW CODE WRITTEN,
0000     99 ;                  BUT ACTUAL REMOVAL IS DEFERRED). Fix route through code
0000    100 ;                  on endnodes to simply return the packet, rather than
0000    101 ;                  generate a Packet Format Error.
0000    102 ;
0000    103 ;          V03-028 RNG0028         Rod Gamache     02-May-1983
0000    104 ;                  Fix the RTS code for sending to Phase III nodes from other
0000    105 ;                  areas. Clean up reception of Broadcast Endnode Hellos.
0000    106 ;
0000    107 ;          V03-027 RNG0027         Rod Gamache     30-Apr-1983
0000    108 ;                  Don't send messages from other areas to Phase III endnodes.
0000    109 ;                  Check BIT6 in route header flags byte (must be zero).
0000    110 ;                  Update LISTENER TIMER on hello message only if it is a
0000    111 ;                  Broadcast Circuit.
0000    112 ;                  Don't send message to Endnode if the destination address is
0000    113 ;                  not the Endnode's.
0000    114 ;
```

```
0000   115 ;        V03-026 RNG0026         Rod Gamache     20-Apr-1983
0000   116 ;                Do not send the area number in hello messages to Phase
0000   117 ;                III nodes. Fix sending hello messages on endnodes.
0000   118 ;
0000   119 ;        V03-025 RNG0025         Rod Gamache     01-Apr-1983
0000   120 ;                Only check HIORD when delivering a packet to the ECL
0000   121 ;                layer or when converting the packet to short format.
0000   122 ;                Only set the Intra-NI flag header bit when: the message
0000   123 ;                is received from the sender and the output ADJ is the
0000   124 ;                destination and the input LPD and output LPD are the
0000   125 ;                same BC circuit. Also only set when the Input and Output
0000   126 ;                areas are the same as our own (multi-area NIs).
0000   127 ;                Do not allow messages from other areas to be sent to
0000   128 ;                Phase III routers.
0000   129 ;
0000   130 ;        V03-024 RNG0024         Rod Gamache     14-Mar-1983
0000   131 ;                Start building the XPT pad bytes for datalinks that
0000   132 ;                require padding.
0000   133 ;                Do not use AOA vector if we are an isolated area router.
0000   134 ;                Make the reachability code a general subroutine.
0000   135 ;                Conform to change in RHEL and EHEL Hello Timer field.
0000   136 ;
0000   137 ;        V03-023 RNG0023         Rod Gamache     10-Mar-1983
0000   138 ;                Make XPT pad byte count inclusive of the byte count
0000   139 ;                byte.
0000   140 ;
0000   141 ;        V03-022 TMH0022         Tim Halvorsen   14-Feb-1983
0000   142 ;                Get datalink buffer size from cell in the LPD rather
0000   143 ;                than computing it from RCB value.  This allows different
0000   144 ;                datalinks to have different buffer sizes because of their
0000   145 ;                different size Transport route headers.
0000   146 ;                If NSP requests a transmit to a specific LPD, and gives
0000   147 ;                a remote node address (not a loopback address) as well,
0000   148 ;                then lookup the correct ADJ and use that, rather than
0000   149 ;                sending the message to an arbitrary BC adjacency.
0000   150 ;                Add code to parse the variable length pad field at front
0000   151 ;                of received messages.
0000   152 ;
0000   153 ;        V03-021 TMH0021         Tim Halvorsen   21-Jan-1983
0000   154 ;                Fix route-thru not to destroy the address of the LPD we
0000   155 ;                initially received the packet on, so that any errors in
0000   156 ;                return-to-sender are logged with a consistent LPD address.
0000   157 ;                Change all checks for endnodes to use $DISPATCH macro to
0000   158 ;                include Phase III endnode case.
0000   159 ;                Fix support of loop nodes over broadcast circuits on which
0000   160 ;                our node is the designated router. Also fix loop nodes on
0000   161 ;                endnodes which have the LPD set to loopback.
0000   162 ;
0000   163 ;        V03-020 RNG0020         Rod Gamache     18-Jan-1983
0000   164 ;                Cleanup the cache timeout handling to work properly in all
0000   165 ;                cases.
0000   166 ;
0000   167 ;        V03-019 TMH0019         Tim Halvorsen   18-Jan-1983
0000   168 ;                Fix bug in endnode solicit, so that messages destined
0000   169 ;                for ourself don't go to the designated router.
0000   170 ;                Exclude RTS messages from addition to the endnode cache,
0000   171 ;                since in an RTS message, the source address isn't really
```

```
0000   172 ;                valid.
0000   173 ;
0000   174 ;        V03-018 RNG0018         Rod Gamache     11-Jan-1983
0000   175 ;                Move cache handling routine to Route Header processing
0000   176 ;                routine. Fix Endnode problem for connecting to node 0
0000   177 ;                when the only circuit is turned off. Use symbols for
0000   178 ;                computing number of nodes to scan in a 1 second interval.
0000   179 ;                Add code to deallocate the LPD CACHE table.
0000   180 ;
0000   181 ;        V03-017 RNG0017         Rod Gamache     06-Jan-1983
0000   182 ;                Fix cache table handling and fix RTS code for route-thru
0000   183 ;                case.
0000   184 ;
0000   185 ;        V03-016 RNG0016         Rod Gamache     30-Nov-1982
0000   186 ;                Fix MOP LOOPBACK to not build a route header.
0000   187 ;                Add the ENDNODE CACHE to ENDNODE support.
0000   188 ;                Do not decrement IRPCNT when queuing CRD message
0000   189 ;                to NETACP, so that LPD activity is stopped until
0000   190 ;                the CRD message is received and processed by NETACP.
0000   191 ;
0000   192 ;        V03-015 RNG0015         Rod Gamache     29-Nov-1982
0000   193 ;                Fix massive bugs in LOOPBACK code.
0000   194 ;
0000   195 ;        V03-014 RNG0014         Rod Gamache     07-Oct-1982
0000   196 ;                Add support for Phase IV area routing.
0000   197 ;                Fix bug in processing of Phase II route headers,
0000   198 ;                which caused the source address in the CXB to be
0000   199 ;                left zero, causing replies to be sent to the wrong
0000   200 ;                node.
0000   201 ;                Fix two bugs which prevented STATE SHUT from working.
0000   202 ;                Use new long format data message header. Add return
0000   203 ;                to sender path for NSP.
0000   204 ;
0000   205 ;        V03-013 RNG0013         Rod Gamache     24-Sep-1982
0000   206 ;                Add support for Phase IV endnodes.
0000   207 ;
0000   208 ;        V03-012 TMH0012         Tim Halvorsen   14-Sep-1982
0000   209 ;                Fix CRC16 checks to avoid CRC instruction if the message
0000   210 ;                length is 0-2, and signal an error immediately (short
0000   211 ;                message size).
0000   212 ;                Don't pre-allocate IRPs up to the "maximum buffers"
0000   213 ;                limit, but instead only allocate IRPs when you need
0000   214 ;                them.
0000   215 ;                On each timer tick, dynamically reduce the size of
0000   216 ;                the IRP_FREE list, so that the list slowly reacts
0000   217 ;                to reduced traffic through the node, and always converges
0000   218 ;                to the optimum number of IRPs needed.
0000   219 ;                Add support for journalling Transport I/O.
0000   220 ;
0000   221 ;        V03-011 RNG0004         Rod N. Gamache  08-Sep-1982
0000   222 ;                Fix sending of Phase II NOP messages, to not skip the 6
0000   223 ;                bytes of header.
0000   224 ;
0000   225 ;        V03-010 RNG0003         Rod N. Gamache  02-Sep-1982
0000   226 ;                Fix all error returns to NETACP to return the packet
0000   227 ;                size. Set up ADJ pointer in WQE before checking the
0000   228 ;                CRC on X.25 circuits.
```

```
0000    229 ;
0000    230 ;       V03-009 RNG0002         Rod N. Gamache  20-Aug-1982
0000    231 ;               If we are the designated router on a Broadcast Circuit,
0000    232 ;               then send a "Broadcast Endnode Hello" message when the
0000    233 ;               "Broadcast Router Hello" message is sent.
0000    234 ;
0000    235 ;       V03-008 RNG0001         Rod N. Gamache  13-Jul-1982
0000    236 ;               Add Phase IV support to transport.
0000    237 ;
0000    238 ;--
```

```
                     0000    240 ;
                     0000    241 ;  EXTERNAL SYMBOLS
                     0000    242 ;
                     0000    243         $ADJDEF                 ; Adjacency control block definitions
                     0000    244         $AQBDEF                 ; ACP Queue Block
                     0000    245         $CADEF                  ; Conditionally turn on performance monitoring
                     0000    246         $CXBDEF                 ; Network receive block definitions
                     0000    247         $DYNDEF                 ; Block type definitions
                     0000    248         $FKBDEF                 ; Fork Block Definitions
                     0000    249         $IPLDEF                 ; Define interrupt priority levels
                     0000    250         $IRPDEF                 ; I/O Request Packet
                     0000    251         $VADEF                  ; Virtual address symbols
                     0000    252         $XMDEF                  ; DMC-11 Driver symbols
                     0000    253
                     0000    254         $NETSYMDEF              ; Miscellaneous symbols
                     0000    255         $NETMSGDEF              ; ACP receive buffer symbols
                     0000    256         $NETUPDDEF              ; LPD 'update' function codes
                     0000    257         $NSPMSGDEF              ; NSP and TR message definitions
                     0000    258
                     0000    259         $CXBEXTDEF              ; NETDRIVER extensions to the CXB
                     0000    260
                     0000    261         $LPDDEF                 ; Logical Path Descriptor
                     0000    262         $RCBDEF                 ; Routining Control Block
                     0000    263         $WQEDEF                 ; Work Queue Element
                     0000    264
                     0000    265
                     0000    266 ;
                     0000    267 ;  LOCAL SYMBOLS
                     0000    268 ;
00000004             0000    269         RETRY_TIMER     = 4     ; Error retry time on hello msg transmission
                     0000    270                                 ;  or listener timeout notification failure
00000024             0000    271         HELLO_MSG_SIZE  = 34+2  ; Size of worst case hello msg + 2 spare bytes
                     0000    272                                 ;    Fixed size of BC router hello msg is 27
                     0000    273                                 ;    Fixed size of BC endnode hello msg is 34
                     0000    274                                 ;    Fixed size of non-BC hello msg is 6
0000000A             0000    275         XPT_C_CACHETIMER = 10   ; Check cache timeout every 10 seconds
00000046             0000    276         XPT_C_CACHETIMEOUT = 70 ; Purge cache entry after 70 seconds of inactivity
00000400             0000    277         MAX_NODES       = 1024  ; Node data base has 1024 nodes maximum
00000100             0000    278         NODES_PER_PASS  = 256   ; Nodes to process per pass (1 second interval)
00000000             0000    279         NODE_SHIFT      = 0     ; Shift value for nodes per pass (initial value)
                     0000    280         ;
                     0000    281         ; Compute real node shift value.
                     0000    282         ; Calculate NODE_SHIFT as LOG base 2 of NODES_PER_PASS
                     0000    283         ;
00000100             0000    284         TEMP = NODES_PER_PASS   ; Initialize temporary value
                     0000    285         .REPT   10              ; Repeat for 2**10 (1024) max value
                     0000    286         .IIF LT TEMP-2, .MEXIT  ; Exit if all done
                     0000    287         TEMP=TEMP@-1            ; Else, shift again
                     0000    288         NODE_SHIFT=NODE_SHIFT+1 ; Compute log
                     0000    289         .ENDR                   ; Go again
                     0000    290
                     0000    291
00000040             0000    292         IRP$Q_STATION = IRP$Q_NT_PRVMSK
                     0000    293
00000001             0000    294         JNX$$$ = 1              ; Enables journalling
                     0000    295
                     0000    296 ;
```

```
                     0000   297 ; MACROS
                     0000   298 ;
                     0000   299 .MACRO  INCPMS  PMS_CELL                    ; Increment PMS cell
                     0000   300                                             ;
                     0000   301         .IF DF  CA$_MEASURE                 ;
                     0000   302         .IF NE  CA$_MEASURE                 ; Conditional assembly
                     0000   303                 INCL G^PMS$GL_'PMS_CELL'    ; Bump the counter
                     0000   304         .ENDC                               ;
                     0000   305         .ENDC                               ;
                     0000   306 .ENDM   INCPMS                              ;
                     0000   307
                     0000   308
                00000000   309         .PSECT  $$$115_DRIVER,LONG,EXE,RD,WRT   ; Goto code PSECT
                     0000   310
                     0000   311 ;
                     0000   312 ; Define polynomial table for calculating CRC16 on X.25 datagrams.
                     0000   313 ;
00000000             0000   314 CRC16:  .LONG   ^X00000000
0000CC01             0004   315         .LONG   ^X0000CC01
0000D801             0008   316         .LONG   ^X0000D801
00001400             000C   317         .LONG   ^X00001400
0000F001             0010   318         .LONG   ^X0000F001
00003C00             0014   319         .LONG   ^X00003C00
00002800             0018   320         .LONG   ^X00002800
0000E401             001C   321         .LONG   ^X0000E401
0000A001             0020   322         .LONG   ^X0000A001
00006C00             0024   323         .LONG   ^X00006C00
00007800             0028   324         .LONG   ^X00007800
0000B401             002C   325         .LONG   ^X0000B401
00005000             0030   326         .LONG   ^X00005000
00009C01             0034   327         .LONG   ^X00009C01
00008801             0038   328         .LONG   ^X00008801
00004400             003C   329         .LONG   ^X00004400
                     0040   330
```

```
                                K 2
0040   332          .SBTTL  TR$UPDATE       - Initiate receive sequence on data link
0040   333
0040   334 ;+
0040   335 ;  TR$UPDATE  - Update according to datalink state transition
0040   336 ;
0040   337 ;
0040   338 ;  For R0 = NETUPD$_DLL_ON
0040   339 ;
0040   340 ;  Allocate and initialize a "receive" IRP for a particular LPD and introduce
0040   341 ;  it into the network pool.  This operation happens once each time an LPD
0040   342 ;  becomes available for network traffic. If we are an endnode, allocate the
0040   343 ;  endnode cache table for the LPD.
0040   344 ;
0040   345 ;
0040   346 ;  For R0 = NETUPD$_REACT_RCV
0040   347 ;
0040   348 ;  A suspended receive IRP may be reactivated.  This interface is used to
0040   349 ;  restart the receiver which was stalled due to a receive buffer needing
0040   350 ;  to be passed to NETACP while the XM$V_STS_BUFFAIL bit was set in the IRP.
0040   351 ;  NETACP attaches the receive buffer to IRP$L_SVAPTE before calling this
0040   352 ;  routine.
0040   353 ;
0040   354 ;
0040   355 ;  For R0 = NETUPD$_SEND_HELLO
0040   356 ;
0040   357 ;  The NETACP wishes to inform other uses of the establishment of 2-way
0040   358 ;  communication on a broadcast circuit. The TRANSPORT layer with send out
0040   359 ;  a HELLO message immediately instead of waiting for the HELLO TIMER.
0040   360 ;
0040   361 ;  For R0 = NETUPD$_TEST_ADJ
0040   362 ;
0040   363 ;  The NETACP wants to know if a node specified by a node address can be found
0040   364 ;  in the endnode cache (i.e. is it one hop distant?).
0040   365 ;
0040   366 ;
0040   367 ;  INPUTS:      R5      NETDRIVER UCB pointer
0040   368 ;               R4,R3   Scratch
0040   369 ;               R2      RCB pointer
0040   370 ;               R1      LPD pointer
0040   371 ;               R0      Dispatch code (scratch)
0040   372 ;
0040   373 ;  OUTPUTS:     R5      Preserved
0040   374 ;               R4,R3   Garbage
0040   375 ;               R2,R1   Preserved
0040   376 ;               R0      LBS if successful, else LBC
0040   377 ;
0040   378 ;-
0040   379 TR$UPDATE::                                          ; Update LPD
0040   380         $DISPATCH R0,TYPE=W,-                        ; Dispatch on fuction request
0040   381         <-
0040   382                 <NETUPD$_DLL_ON  INIT_RCV>,-         ; Datalink starting
0040   383                 <NETUPD$_REACT_RCV REACT_RCV>,-      ; Reactivate a receiver
0040   384                 <NETUPD$_SEND_HELLO SEND_HELLO>,-    ; Send a hello msg
0040   385                 <NETUPD$_GET_ADJ GET_OUT_ADJ>,-      ; Get ADJ address for output
0040   386                 <NETUPD$_TEST_ADJ TEST_ADJ>,-        ; Test if node is 1 hop away
0040   387         >
50  D4  005A   388         CLRL    R0                           ; All others - indicate error
```

```
                 05  005C   389          RSB                                     ; Return to caller
                     005D   390
                     005D   391 TEST_ADJ:
          03CA 8F    BB  005D   392          PUSHR   #^M<R1,R3,R6,R7,R8,R9>          ; Save registers
05        008A C2    91  0061   393          CMPB    RCB$B_ETY(R2),#ADJ$C_PTY_PH4N   ; Is this an endnode?
             25      12  0066   394          BNEQ    10$                             ; If NEQ, bug (shouldn't be called)
             53      D4  0068   395          CLRL    R3                              ; No LPD wanted here
           052C      30  006A   396          BSBW    TR$GET_ADJ                      ; Get the output ADJ
         1D  50      E9  006D   397          BLBC    R0,10$                          ; If LBC, not even reachable
             59      D5  0070   398          TSTL    R9                              ; paranoia check
             19      13  0072   399          BEQL    10$                             ; If no ADJ, not reachable
                     0074   400          ;
                     0074   401          ;  R8 -> LPD, R9 -> ADJ for the path to this node.
                     0074   402          ;
                     0074   403          ;  Non-broadcast circuits: we can compare the node address with the
                     0074   404          ;  address in the ADJ to see if we're one hop away.
                     0074   405          ;
                     0074   406          ;  Broadcast circuits: Search the cache for the node address.
                     0074   407          ;
          50   01    D0  0074   408          MOVL    #1,R0                           ; Assume node is adjacent
             58      D5  0077   409          TSTL    R8                              ; Make sure we have LPD
             12      13  0079   410          BEQL    10$                             ; If EQL, not adjacent
08 22 A8     0A      E0  007B   411          BBS     #LPD$V_BC,LPD$W_STS(R8),5$      ; If BS, it's a broadcast ckt
   04 A9     54      B1  0080   412          CMPW    R4,ADJ$W_PNA(R9)                ; If not, does address match?
             09      13  0084   413          BEQL    20$                             ; If EQL, node is adjacent
             05      11  0086   414          BRB     10$                             ; Else, yes: adjacent node
           0685      30  0088   415 5$:      BSBW    SCAN_CACHE                      ; Search cache for this LPD
             02      11  008B   416          BRB     20$                             ; If LBS, found in cache
             50      D4  008D   417 10$:     CLRL    R0                              ; Not in cache
          03CA 8F    BA  008F   418 20$:     POPR    #^M<R1,R3,R6,R7,R8,R9>          ; Restore registers
                 05  0093   419          RSB
                     0094   420
                     0094   421                  .ENABL  LSB
                     0094   422
                     0094   423 GET_OUT_ADJ:                                     ; Find the output adjacency
          00C2 8F    BB  0094   424          PUSHR   #^M<R1,R6,R7>                   ; Save registers
             04FE    30  0098   425          BSBW    TR$GET_ADJ                      ; Get the output ADJ
          00C2 8F    BA  009B   426          POPR    #^M<R1,R6,R7>                   ; Restore registers
                 05  009F   427          RSB                                     ; Return to caller with status
                     00A0   428
                     00A0   429 SEND_HELLO:                                      ; Force sending a hello msg
          03FE 8F    BB  00A0   430          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
          58   51    D0  00A4   431          MOVL    R1,R8                           ; Copy LPD address
          5E   18    C2  00A7   432          SUBL    #FKB$C_LENGTH,SP                ; Create context block on stack
          55   5E    D0  00AA   433          MOVL    SP,R5                           ; Point to fork block
             0299    30  00AD   434          BSBW    TALKER                          ; Send hello message
          5E   18    C0  00B0   435          ADDL    #FKB$C_LENGTH,SP                ; Reset stack pointer
          03FE 8F    BA  00B3   436          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
             00EC    31  00B7   437          BRW     100$                            ; Exit with status
                     00BA   438
                     00BA   439 REACT_RCV:                                       ; Reactivate stalled receiver
             26      BB  00BA   440          PUSHR   #^M<R1,R2,R5>                   ; Save regs
                     00BC   441          ;
55        32 A1      D0  00BC   442          MOVL    LPD$L_RCV_IRP(R1),R5            ; Get IRP
          32 A1      D4  00C0   443          CLRL    LPD$L_RCV_IRP(R1)               ; No longer attached to LPD
                     00C3   444          ;
                     00C3   445          ;
```

```
                        00C3    446          ;   Say "success" and "zero bytes transferred" in IOST1.  These
                        00C3    447          ;   conditions will cause the buffer and IRP to be sent back to the
                        00C3    448          ;   datalink driver without signalling any further errors and without
                        00C3    449          ;   re-interpreting the buffer contents.
                        00C3    450          ;
                        00C3    451          ;
                        00C3    452          ASSUME  IRP$L_IOST2  EQ  4+IRP$L_IOST1
    38 A5   00'  7D     00C3    453          MOVQ    S^#SS$_NORMAL,IRP$L_IOST1(R5)   ; Enter "success" and "no bytes
                        00C7    454                                                  ; transferred".  Zero IOST2
    51   2C A5  D0      00C7    455          MOVL    IRP$L_SVAPTE(R5),R1             ; Get CXB address
    24 A5   51  D0      00CB    456          MOVL    R1,IRP$L_IOSB(R5)               ; Reset the CXB address here
              04  13     00CF    457          BEQL    1$                              ; Br if none
    0A A1     1B  90     00D1    458          MOVB    #DYN$C_CXB,CXB$B_TYPE(R1)       ; Else, reset the buffer type
        0C B5     16     00D5    459 1$:      JSB     @IRP$L_PID(R5)                  ; Recycle the buffer
                        00D8    460                                                  ;
           26     BA     00D8    461          POPR    #^M<R1,R2,R5>                   ; Restore regs
         00C9     31     00DA    462          BRW     100$                            ; Take common exit
                        00DD    463
                        00DD    464
                        00DD    465 INIT_RCV:                                        ; Queue receive to data link
 54 22 A1    0A  E1     00DD    466          BBC     #LPD$V_BC,LPD$W_STS(R1),3$      ; Br if not a broadcast circuit
  008A C2    05  91     00E2    467          CMPB    #ADJ$C_PTY_PH4N,RCB$B_ETY(R2)   ; Are we a real Endnode?
             4D  12     00E7    468          BNEQ    3$                              ; Br if not
    01   20 A1  91     00E9    469          CMPB    LPD$B_PTH_INX(R1),#LPD$C_LOC_INX; Is this for the "Local" LPD?
             47  13     00ED    470          BEQL    3$                              ; Br if yes - no CACHE needed
           66 A1  D5     00EF    471          TSTL    LPD$L_CACHE(R1)                 ; Is the CACHE already allocated?
             42  12     00F2    472          BNEQ    3$                              ; Br if yes - all set
                        00F4    473          ;
                        00F4    474          ;       Allocate the ENDNODE CACHE. The size of each entry is 4 bytes.
                        00F4    475          ;       The number of entries will be the maximum number of entries +
                        00F4    476          ;       some extra for the DRT and others attempting to connect.
                        00F4    477          ;
    7E   51  7D         00F4    478          MOVQ    R1,-(SP)                        ; Save registers
 51   58 A2  3C         00F7    479          MOVZWL  RCB$W_MAX_LNK(R2),R1            ; Get number of entries needed
    51   04  C4         00FB    480          MULL    #4,R1                           ; Calculate 4 bytes per entry
    51   1C  C0         00FE    481          ADDL    #<4*4>+12,R1                    ; Make room for some extra
                        0101    482                                                  ;  entries plus struct header
00000000'GF  16         0101    483          JSB     G^EXE$ALONONPAGED               ; Try to allocate the CACHE
    53   52  D0         0107    484          MOVL    R2,R3                           ; Save CACHE address (if good)
    54   51  D0         010A    485          MOVL    R1,R4                           ; Save CACHE size
    51   8E  7D         010D    486          MOVQ    (SP)+,R1                        ; Restore registers
      34 50  E9         0110    487          BLBC    R0,209$                         ; Br if error
                        0113    488          ;
                        0113    489          ;       Initialize the CACHE. Set the structure type, size and
                        0113    490          ;       zero the rest of the entries. The cache is as follows:
                        0113    491          ;
                        0113    492          ;   +---------------+
                        0113    493          ;   |               |   4 bytes unused
                        0113    494          ;   |               |
                        0113    495          ;   +---------------+
                        0113    496          ;   |      Time     |   2 bytes for time to update cache.
                        0113    497          ;   +---------------+
                        0113    498          ;   |     Number    |   2 bytes for number of entries in the Endnode
                        0113    499          ;   |       of      |   Cache.
                        0113    500          ;   |    Entries    |
                        0113    501          ;   +---------------+
                        0113    502          ;   |               |   2 bytes for size of structure
```

```
                                0113   503          ;   +---------------+
                                0113   504          ;   |               |   2 bytes for type of structure
                                0113   505          ;   +---------------+
                                0113   506          ;   |               |   LPD$L_CACHE points here.
                                0113   507          ;   .    body of     .   Each entry contains 2 bytes of address in
                                0113   508          ;   .     cache      .   low word, followed by 2 bytes of time last
                                0113   509          ;   .                .   used.
                                0113   510          ;   .                .
                                0113   511          ;   |               |
                                0113   512          ;   +---------------+
                                0113   513          ;
                                0113   514          ;
                          3E    BB    0113   515          PUSHR   #^M<R1,R2,R3,R4,R5>                     ; Save registers
      63   54   00   63      00    2C    0115   516          MOVC5   #0,(R3),#0,R4,(R3)                      ; Zero the structure
                          3E    BA    011B   517          POPR    #^M<R1,R2,R3,R4,R5>                     ; Restore registers
                50   54      0C    C3    011D   518          SUBL3   #12,R4,R0                               ; Get size of CACHE - header
                     50      04    C6    0121   519          DIVL    #4,R0                                   ; Calculate number of entries
                             83    D4    0124   520          CLRL    (R3)+                                   ; Skip first longword
                          83 0A    B0    0126   521          MOVW    #XPT_C_CACHETIMER,(R3)+                 ; Initialize CACHE timer period
                          83 50    B0    0129   522          MOVW    R0,(R3)+                                ; Set # of entries in cache
                          83 54    B0    012C   523          MOVW    R4,(R3)+                                ; Set size of structure
                          83 17    B0    012F   524          MOVW    #DYN$C_NET,(R3)+                        ;
                  66 A1   53       D0    0132   525          MOVL    R3,LPD$L_CACHE(R1)                      ; Save address of CACHE table
                                0136   526          ;
                                0136   527          ;       Queue initial receive to datalink
                                0136   528          ;
                55   07CF'CF      9E    0136   529 3$:      MOVAB   W^TR$RCV_BIO_DATA,R5                    ; Setup IRP return address
              0A 22 A1    06     E0    013B   530          BBS     #LPD$V_RBF,LPD$W_STS(R1),10$            ; If BS then reads are buffered
                55   072B'CF      9E    0140   531          MOVAB   W^TR$RCV_DIO_DATA,R5                    ; Setup IRP return address
                          63     10    0145   532 5$:      BSBB    INIT_CXB_FREE                           ; Init Free CXB queue
                     5F 50       E9    0147   533 209$:     BLBC    R0,200$                                 ; If LBC then report error
                                014A   534 10$:     ;
                                014A   535          ;
                                014A   536          ;   Common IRP setup
                                014A   537          ;
                                014A   538          ;
          50    0000'8F         3C    014A   539          MOVZWL  #SS$_DEVACTIVE,R0                       ; Assume error
                                014F   540          ASSUME  LPD$V_ACTIVE  EQ  0                     ;
             56 22 A1           E8    014F   541          BLBS    LPD$W_STS(R1),200$                      ; Br if already active
                   0FF2         30    0153   542          BSBW    TR$ALLOC_IRP                            ; Allocate the IRP
                50 50           E9    0156   543          BLBC    R0,200$                                 ; Br on error
                   0C A2        B6    0159   544          INCW    RCB$W_TRANS(R2)                         ; Account for IRP
           54      0C A3        9E    015C   545          MOVAB   IRP$L_PID(R3),R4                        ; Setup ptr to build IRP
                                0160   546
                                0160   547          ASSUME  IRP$L_AST       EQ  4+IRP$L_PID
                                0160   548          ASSUME  IRP$L_ASTPRM    EQ  4+IRP$L_AST
                                0160   549          ASSUME  IRP$L_WIND      EQ  4+IRP$L_ASTPRM
                                0160   550          ASSUME  IRP$L_UCB       EQ  4+IRP$L_WIND
                                0160   551          ASSUME  LPD$L_UCB       EQ  4+LPD$L_WIND
                                0160   552
             84    55           D0    0160   553          MOVL    R5,(R4)+                                ; Move return address into PID
          84    20 A1           3C    0163   554          MOVZWL  LPD$W_PTH(R1),(R4)+                     ; Enter LPD i.d. into AST
             84    52           D0    0167   555          MOVL    R2,(R4)+                                ; Enter RCB ptrs into ASTPRM
          84    0C A1           7D    016A   556          MOVQ    LPD$L_WIND(R1),(R4)+                    ; Enter WIND and UCB ptrs
                                016E   557
                                016E   558          ASSUME  IRP$W_FUNC      EQ  4+IRP$L_UCB
                                016E   559          ASSUME  IRP$B_EFN       EQ  2+IRP$W_FUNC
```

```
                                016E    560          ASSUME  IRP$B_PRI       EQ  1+IRP$B_EFN
                                016E    561          ASSUME  IRP$L_IOSB      EQ  1+IRP$B_PRI
                                016E    562          ASSUME  IRP$W_CHAN      EQ  4+IRP$L_IOSB
                                016E    563
                          84    7C    016E    564          CLRQ    (R4)+                           ; Clear FUNC,EFN,PRI,IOSB
                    84    14 A1   AE    0170    565          MNEGW   LPD$W_CHAN(R1),(R4)+           ; Enter CHAN
                                0174    566          ;
                                0174    567          ;
                                0174    568          ;   If the LPD does direct I/O on receives, remove the buffer from the
                                0174    569          ;   RCB CXB free queue and attach it to the IRP.  For buffered I/O
                                0174    570          ;   receiver's, no buffer is allocated since that is a datalink
                                0174    571          ;   function -- due to possible buffer size requirements, only those
                                0174    572          ;   buffers allocated by the datalink receiver are ever returned to the
                                0174    573          ;   datalink via the IRP (e.g., during BUFFAIL).
                                0174    574          ;
                                0174    575          ;
                                0174    576          ASSUME  IRP$W_STS       EQ  2+IRP$W_CHAN
                                0174    577          ASSUME  IRP$L_SVAPTE    EQ  2+IRP$W_STS
                                0174    578          ASSUME  IRP$W_BOFF      EQ  4+IRP$L_SVAPTE
                                0174    579          ASSUME  IRP$W_BCNT      EQ  2+IRP$W_BOFF
                                0174    580
                          84    03    B0    0174    581          MOVW    #IRP$M_FUNC!IRP$M_BUFIO,(R4)+   ; Setup STS for read functions
                    13 22 A1    06    E0    0177    582          BBS     #LPD$V_RBF,LPD$W_STS(R1),30$    ; If BS then reads are buffered
                       FE A4    01    AA    017C    583          BICW    #IRP$M_BUFIO,-2(R4)             ; Setup for direct I/O
                     50    00A0 D2    0F    0180    584          REMQUE  @RCB$Q_CXB_FREE(R2),R0          ; Get CXB
                                04    1C    0185    585          BVC     20$                             ; If VC then got one
                                      0187    586          BUG_CHECK  NETNOSTATE,FATAL             ; Queue should have been
                                      018B    587                                                  ; non-empty at this point
                          24 A3 50    D0    018B    588 20$:     MOVL    R0,IRP$L_IOSB(R3)               ;
                                84    7C    018F    589 30$:     CLRQ    (R4)+                           ; Clear SVAPTE, BOFF, W_BCNT
                FE A4       3FFF 8F   B0    0191    590          MOVW    #^X<3FFF>,-2(R4)                ; Setup W_BCNT
                                      0197    591
                                      0197    592          ASSUME  IRP$L_BCNT      EQ  0+IRP$W_BCNT
                                      0197    593          ASSUME  IRP$L_IOST1     EQ  6+IRP$L_BCNT
                                      0197    594          ASSUME  IRP$L_IOST2     EQ  4+IRP$L_IOST1
                                      0197    595
                                84    D4    0197    596          CLRL    (R4)+                           ; Clear high word of L_BCNT
                                      0199    597                                                  ; and next reserved word
                          84    00' 7D    0199    598          MOVQ    S^#SS$_NORMAL,(R4)+             ; Enter "success" and "no bytes
                                      019C    599                                                  ; xferred into IOST1 -- this is
                                      019C    600                                                  ; the standard method for
                                      019C    601                                                  ; admitting IRP's into the
                                      019C    602                                                  ; receive cycle.
                             1C A1    96    019C    603          INCB    LPD$B_IRPCNT(R1)                ; Account for IRP to be queued
                       22 A1    01    A8    019F    604          BISW    #LPD$M_ACTIVE,LPD$W_STS(R1)     ; Mark LPD active
                          0F48       30    01A3    605 50$:     BSBW    POST                            ; Start the cycle by sending
                                      01A6    606                                                  ; the IRP thru IOPOST
                          50    01    D0    01A6    607 100$:    MOVL    #1,R0                           ; Indicate success
                                      05    01A9    608 200$:    RSB
                                      01AA    609                          .DSABL LSB
                                      01AA    610
                                      01AA    611
                                      01AA    612 INIT_CXB_FREE:                                   ; Init free CXB queue
                                      01AA    613          ;
                                      01AA    614          ;
                                      01AA    615          ;   If the CXB lookaside list used for circuits using Direct I/O
                                      01AA    616          ;   reads is empty then allocate a single CXB and insert it on the
```

```
                         01AA   617          ;   queue.
                         01AA   618          ;
                         01AA   619          ;
                    0E   BB  01AA   620          PUSHR   #^M<R1,R2,R3>                ; Save regs
                         01AC   621          ;
                50  01   D0  01AC   622          MOVL    #1,R0                        ; Assume queue is non-empty
             53 00A0 C2  9E  01AF   623          MOVAB   RCB$Q_CXB_FREE(R2),R3        ; Get queue header address
                63  53   D1  01B4   624          CMPL    R3,(R3)                      ; Any free CXB's ?
                    1A   12  01B7   625          BNEQ    10$                          ; If NEQ then yes
                    50   D4  01B9   626          CLRL    R0                           ; Assume ACP not "up" yet
                  61 A2  95  01BB   627          TSTB    RCB$B_STI(R2)                ;&symbol  Can we trust the buffer size
                    13   13  01BE   628          BEQL    10$                          ; If EQL then no
             51  7E A2  3C  01C0   629          MOVZWL  RCB$W_TOTBUFSIZ(R2),R1       ; Get buffer size assuming 6 byte
                         01C4   630                                                   ; route header
                51  16   C0  01C4   631          ADDL    #TR$C_MAXHDR-6,R1            ; Adjust to account for largest
                         01C7   632                                                   ; possible route header (NI)
                51  02   A0  01C7   633          ADDW    #2,R1                        ; Add 2 extra bytes just in case this
                         01CA   634                                                   ; is a X.25 DLM datalink
                   0FAA  30  01CA   635          BSBW    TR$ALLOCATE                  ; Allocate a CXB
                 03 50   E9  01CD   636          BLBC    R0,10$                       ; If LBC then allocation failure
                93  62   0E  01D0   637          INSQUE  (R2),@(R3)+                  ; Insert CXB on the queue
                         01D3   638          ;
                    0E   BA  01D3   639 10$:     POPR    #^M<R1,R2,R3>                ; Restore regs
                         05  01D5   640          RSB                                  ; Return status in R0
                         01D6   641
```

```
                                01D6   643          .SBTTL  TR$KILL_LOC_LPD - Attempt to shutdown Local LPD
                                01D6   644
                                01D6   645 ;+
                                01D6   646 ;  TR$KILL_LOC_LPD - Attemp to shutdown Local LPD
                                01D6   647 ;
                                01D6   648 ;
                                01D6   649 ;  This routine is called when the network is shutting down.  It checks to
                                01D6   650 ;  see if the "local LPD" has run-down.  If so, it notifies the NETACP and
                                01D6   651 ;  deactivates the local LPD.
                                01D6   652 ;
                                01D6   653 ;
                                01D6   654 ;  INPUTS:       R2      RCB address
                                01D6   655 ;
                                01D6   656 ;  OUTPUTS:      R3      Garbage
                                01D6   657 ;                R2      Preserved
                                01D6   658 ;                R1      Garbage
                                01D6   659 ;                R0      Low bit set if the local LPD is deactivated
                                01D6   660 ;                        Low bit clear otherwise
                                01D6   661 ;
                                01D6   662 ;                All other registers are unchanged.
                                01D6   663 ;
                                01D6   664 ;-
                                01D6   665 TR$KILL_LOC_LPD::                            ; Deactivate local LPD
                    01F0 8F     BB 01D6   666          PUSHR   #^M<R4,R5,R6,R7,R8>     ; Save regs
                                01DA   667                                             ;
                    0082 C2     B4 01DA   668          CLRW    RCB$W_MAX_PKT(R2)       ; Force IRP queue to empty
                       0F25     30 01DE   669          BSBW    TR$ADJUST_IRP           ; Purge it
                         50     D4 01E1   670          CLRL    R0                      ; Assume we must wait
                    0080 C2     B5 01E3   671          TSTW    RCB$W_CUR_PKT(R2)       ; Empty yet?
                         29     12 01E7   672          BNEQ    30$                     ; If not, postpone shutdown
                                01E9   673                                             ;
                  55    3C B2    0F 01E9   674          REMQUE  @RCB$Q_LOC_RCV(R2),R5   ; Get Local receive IRP
                         23     1D 01ED   675          BVS     30$                     ; If VS then its not there
                50   00A0 D2    0F 01EF   676 10$:     REMQUE  @RCB$Q_CXB_FREE(R2),R0  ; Get free CXB
                         08     1D 01F4   677          BVS     20$                     ; If VS then none
                 00000000'GF    16 01F6   678          JSB     G^COM$DRVDEALMEM        ; Deallocate it
                         F1     11 01FC   679          BRB     10$                     ; Loop
                                01FE   680
                                01FE   681 20$:     ASSUME  IRP$L_IOST2  EQ 4+IRP$L_IOST1
                                01FE   682
                       38 A5    7C 01FE   683          CLRQ    IRP$L_IOST1(R5)         ; Clear all status bits -- low bit
                                0201   684                                             ; clear in IOST1 signals I/O error
                       24 A5    D4 0201   685          CLRL    IRP$L_IOSB(R5)          ; No buffer to deallocate
                    58    01    D0 0204   686          MOVL    #LPD$C_LOC_INX,R8       ; Get "local" LPD index
                58  28 B248    D0 0207   687          MOVL    @RCB$L_PTR_LPD(R2)[R8],R8 ; Get the "local" LPD address
                     0D4F      30 020C   688          BSBW    TR_RTRN_IRP             ; Shut down the LPD
                    50    01    D0 020F   689          MOVL    #1,R0                   ; Indicate success
                                0212   690                                             ;
                    01F0 8F     BA 0212   691 30$:     POPR    #^M<R4,R5,R6,R7,R8>     ; Restore regs
                                05 0216   692          RSB                             ; Return status in R0
                                0217   693
```

```
                        0217   695          .SBTTL  TR$TIMER        - Process Transport layer clock tick
                        0217   696
                        0217   697 ;+
                        0217   698 ;  TR$TIMER  -  Process Transport layer clock tick
                        0217   699 ;
                        0217   700 ;
                        0217   701 ;  This routine is called at IPL$_NET every time the network clock ticks.  The
                        0217   702 ;  action here is to process the "Talker" and "Listener" timers on each LPD.
                        0217   703 ;
                        0217   704 ;
                        0217   705 ;  INPUTS:       R2      RCB address
                        0217   706 ;
                        0217   707 ;  OUTPUTS:      R3      Garbage
                        0217   708 ;                R2      Preserved
                        0217   709 ;                R1      Garbage
                        0217   710 ;                R0      Garbage
                        0217   711 ;
                        0217   712 ;                All other registers are unchanged.
                        0217   713 ;
                        0217   714 ;-
                        0217   715 TR$TIMER::                                       ; Called each network clock tick
07F4 8F    BB           0217   716          PUSHR   #^M<R2,R4,R5,R6,R7,R8,R9,R10>   ; Save regs
                        021B   717          ;
                        021B   718          ;  Check to make sure NETACP is still active before doing any more work
                        021B   719          ;  Skip check on Endnodes.
                        021B   720          ;
                        021B   721          $DISPATCH RCB$B_ETY(R2),TYPE=B,-        ; CASE on LOCAL node type
                        021B   722          <-                                      ;
                        021B   723                  <ADJ$C_PTY_PH4N 3$>,-           ; Phase IV endnode
                        021B   724                  <ADJ$C_PTY_PH3N 3$>,-           ; Phase III endnode
                        021B   725          >
                        022B   726          ;
                        022B   727          ;  All others, except endnodes check ACP activity timer.
                        022B   728          ;
        008F C2    95   022B   729          TSTB    RCB$B_ACT_TIMER(R2)             ; Is timer already stopped?
              06   13   022F   730          BEQL    1$                              ; Br if yes
        008F C2    97   0231   731          DECB    RCB$B_ACT_TIMER(R2)             ; Else, decrement timer
              03   14   0235   732          BGTR    3$                              ; Br if okay
           00E1    31   0237   733 1$:      BRW     120$                            ; Else, clear active bit
                        023A   734                                                  ;  and leave now
                        023A   735 3$:       ;
                        023A   736          ;  On each tick, we reduce the IRP free packet list by 1 IRP,
                        023A   737          ;  so that the list dynamically (and slowly) reacts to reduced
                        023A   738          ;  traffic needs, and converges to an optimum size.
                        023A   739          ;
                        023A   740          SETBIT  #RCB$V_ACT,RCB$B_STATUS(R2)     ; Make sure everyone knows
                        023F   741                                                  ;  NETACP is still active.
0A      0080 C2    91   023F   742          CMPB    RCB$W_CUR_PKT(R2),#10           ; Don't let list get too small
              13   1B   0244   743          BLEQU   5$                              ; Skip if list is getting small
  50    00 B2      0F   0246   744          REMQUE  @RCB$Q_IRP_FREE(R2),R0          ; See if there is a free IRP
              0D   1D   024A   745          BVS     5$                              ; Skip if none
 00000000'GF     16   024C   746          JSB     G^COM$DRVDEALMEM                ; Deallocate the IRP
        0080 C2  B7   0252   747          DECW    RCB$W_CUR_PKT(R2)               ; and adjust packet count
           0C A2 B7   0256   748          DECW    RCB$W_TRANS(R2)                 ; Here too
                        0259   749 5$:       ;
                        0259   750          ;  Scan all LPDs for talker and listener
                        0259   751          ;
```

```
                5E   18   C2  0259   752          SUBL    #FKB$C_LENGTH,SP                ; Create context block on stack
                                0025C  753                                                  ; for the "TALKER" routine
            59   5C A2   9A  0025C  754          MOVZBL  RCB$B_MAX_LPD(R2),R9            ; Get number of LPDs
                    58   13  0260   755          BEQL    30$                             ; If EQL then none
                                0262   756          ASSUME  LPD$C_LOC_INX EQ 1
                57   01   9A  0262   757          MOVZBL  #LPD$C_LOC_INX,R7               ; Initialize index
                    4F   11  0265   758          BRB     20$                             ; Start at LOCAL+1
            58   28 B247   D0  0267   759 10$:    MOVL    @RCB$L_PTR_LPD(R2)[R7],R8       ; Get LPD address
                    48   18  026C   760          BGEQ    20$                             ; Branch if slot not used
            50   66 A8     D0  026E   761          MOVL    LPD$L_CACHE(R8),R0              ; Get the CACHE table for this
                                0272   762                                                  ;  LPD
                    26   13  0272   763          BEQL    15$                             ; Br if none
                                0274   764          ;
                                0274   765          ;   Handle CACHE timer
                                0274   766          ;
                F8 A0   B7  0274   767          DECW    -8(R0)                          ; Is it time to check the cache?
                    21   14  0277   768          BGTR    15$                             ; Br if not - skip cache work
            F8 A0    0A B0  0279   769          MOVW    #XPT_C_CACHETIMER,-8(R0)        ; Else, reset cache timer
            55   FA A0   3C  027D   770          MOVZWL  -6(R0),R5                       ; Get # of entries in cache
        00000046 8F     C3  0281   771          SUBL3   #XPT_C_CACHETIMEOUT,-           ; Get Absolute system time
51      00000000'GF             0287   772                  G^EXE$GL_ABSTIM,R1             ;  minus cache timeout period
                    80   B5  028D   773 13$:    TSTW    (R0)+                           ; Skip node address
                80  51   B1  028F   774          CMPW    R1,(R0)+                        ; Is current time > entrytime +
                                0292   775                                                  ;  cache timeout period
                    03   1B  0292   776          BLEQU   14$                             ; Br if not, entry still valid
                FC A0   D4  0294   777          CLRL    -4(R0)                          ; Else, flush the cache entry
                F3 55   F5  0297   778 14$:    SOBGTR  R5,13$                          ;
                                029A   779
    17 22 A8    04   E1  029A   780 15$:    BBC     #LPD$V_RUN,LPD$W_STS(R8),20$    ; If BC then no need to talk
                                029F   781          ;
                                029F   782          ;
                                029F   783          ;   Process talker timer
                                029F   784          ;
                                029F   785          ;       The talker timer cell is located in the LPD data base.
                                029F   786          ;
                  16 A8     B7  029F   787          DECW    LPD$W_TIM_TLK(R8)               ; Tick the talk timer
                     12     14  02A2   788          BGTR    20$                             ; Not expired if GTR
              16 A8    04   B0  02A4   789          MOVW    #RETRY_TIMER,LPD$W_TIM_TLK(R8)  ; Set for retry
                                02A8   790                                                  ; if TALKER resource failure
                 55   5E     D0  02A8   791          MOVL    SP,R5                           ; Setup fork block address
                 0384 8F     BB  02AB   792          PUSHR   #^M<R2,R7,R8,R9>                ; Save vulnerable regs
                    0097     30  02AF   793          BSBW    TALKER                          ; Send a "hello" message
                 0384 8F     BA  02B2   794          POPR    #^M<R2,R7,R8,R9>                ; Restore regs
              AD 57    59     F3  02B6   795 20$:    AOBLEQ  R9,R7,10$                       ; Loop for each cell
                                02BA   796
                 5E   18     C0  02BA   797 30$:    ADDL    #FKB$C_LENGTH,SP                ; Restore the stack
                                02BD   798          ;
                                02BD   799          ;
                                02BD   800          ;   Process listener timer
                                02BD   801          ;
                                02BD   802          ;       The listener timer cell is located in the ADJ data base.
                                02BD   803          ;       We will only process a maximum of 256 ADJs in a one second
                                02BD   804          ;       time interval.
                                02BD   805          ;
            51   68 A2   3C  02BD   806          MOVZWL  RCB$W_MAX_ADJ(R2),R1            ; Get number of adjacencies
                    53   13  02C1   807          BEQL    100$                            ; If EQL then none
        57   00A8 C2     9A  02C3   808          MOVZBL  RCB$B_LSN_ADJ(R2),R7            ; Get current index multiplier
```

```
                                02C8    809                             ;  for processing this time
            57   57   08   78  02C8    810         ASHL    #NODE_SHIFT,R7,R7           ; Get index of where to start
                      02   12  02CC    811         BNEQ    35$                         ; Br if non-zero - okay
                               02CE    812         ASSUME  LPD$C_LOC_INX EQ 1
                           57  D6  02CE    813         INCL    R7                          ; Else, start at "Local"
                               02D0    814 35$:    ;
                               02D0    815         ;   Calculate where to finish processing in this time interval.
                               02D0    816         ;
        53  57   00000100 8F   C1  02D0    817         ADDL3   #NODES_PER_PASS,R7,R3       ; Assume current maximum
                               02D8    818                                             ;   is current + NODES_PER_PASS
                           51  53  D1  02D8    819         CMPL    R3,R1                       ; Is current maximum greater
                               02DB    820                                             ;   than the absolute maximum?
                           03  1B  02DB    821         BLEQU   37$                         ; Br if no - okay
                           53  51  D0  02DD    822         MOVL    R1,R3                       ; Else, set maximum to MAX_ADJ
                               02E0    823 37$:    ;
                               02E0    824         ;   Update multiplier for next time thru.
                               02E0    825         ;
            51  000000FF 8F   C0  02E0    826         ADDL    #NODES_PER_PASS-1,R1        ; Calculate maximum index
             58   51    F8 8F   78  02E7    827         ASHL    #-NODE_SHIFT,R1,R8          ;  to use for this pass
                    00A8 C2   96  02EC    828         INCB    RCB$B_LSN_ADJ(R2)           ; Update next time path
                58  00A8 C2   91  02F0    829         CMPB    RCB$B_LSN_ADJ(R2),R8        ; Modulo NODES_PER_PASS
                           1B  1F  02F5    830         BLSSU   50$                         ;       ...
                    00A8 C2   94  02F7    831         CLRB    RCB$B_LSN_ADJ(R2)           ;       ...
                           15  11  02FB    832         BRB     50$                         ; Start at LOCAL+1
            59   2C B247       D0  02FD    833 40$:    MOVL    @RCB$L_PTR_ADJ(R2)[R7],R9   ; Get ADJ address
              0C 69    03       E1  0302    834         BBC     #ADJ$V_LSN,ADJ$B_STS(R9),50$ ; Br if listen timer not ticking
              0A A9    58       A2  0306    835         SUBW    R8,ADJ$W_TIM_LSN(R9)        ; Tick the listener timer
                       06   1A  030A    836         BGTRU   50$                         ; Not expired if NEQ
                               030C    837         ;
                               030C    838         ;   Listener timer has expired - queue WQE to AQB to signal event
                               030C    839         ;
              0A A9    04       B0  030C    840         MOVW    #RETRY_TIMER,ADJ$W_TIM_LSN(R9) ; Retry if LISTENER
                               0310    841                                             ; resource failure
                       10   10  0310    842         BSBB    LISTENER                    ; Listener has timed out
                               0312    843         ;
            E7 57      53   F3  0312    844 50$:    AOBLEQ  R3,R7,40$                   ; Loop for each cell
                               0316    845         ;
                07F4 8F        BA  0316    846 100$:   POPR    #^M<R2,R4,R5,R6,R7,R8,R9,R10> ; Restore regs
                               031A    847         RSB
                               031B    848         ;
                               031B    849         ;   NETACP is no longer active, it must have stalled.
                               031B    850         ;
                               031B    851         ;
                               031B    852 120$:   CLRBIT  #RCB$V_ACT,RCB$B_STATUS(R2) ; Clear the ACP active bit
                       F4   11  0320    853         BRB     100$                        ; Return
                               0322    854         ;
                               0322    855 LISTENER:                                   ; Listener timer has expired
                7E     52   7D  0322    856         MOVQ    R2,-(SP)                    ; Save regs
                               0325    857         ;
                               0325    858         ASSUME  IRP$C_LENGTH  GE  WQE$C_LENGTH ;
            51   C4 8F        9A  0325    859         MOVZBL  #IRP$C_LENGTH,R1            ; Setup buffer size
                  0E4B       30  0329    860         BSBW    TR$ALLOCATE                 ; Get the buffer
                16 50        E9  032C    861         BLBC    R0,50$                      ; If LBC then didn't get one
                55   52      D0  032F    862         MOVL    R2,R5                       ; Copy buffer for subr call
                52   6E      D0  0332    863         MOVL    (SP),R2                     ; Restore RCB address
          12 A5    02 A9     B0  0335    864         MOVW    ADJ$W_LPD(R9),WQE$W_REQIDT(R5) ; Return ADJ's LPD index
             20 A5    57     B0  033A    865         MOVW    R7,WQE$W_ADJ_INX(R5)        ; Save ADJ index
```

```
                10 A5    09   90  033E   866          MOVB    #NETMSG$C_LSN,WQE$B_EVT(R5)     ; Setup "listner" event
                     0CFE    30  0342   867          BSBW    TR$GIVE_TO_ACP                  ; Pass it to the ACP
                             0345   868                                                      ;
                   52    8E   7D  0345   869 50$:     MOVQ    (SP)+,R2                        ; Restore regs
                             05  0348   870 EXIT:     RSB                                     ; Done
                             0349   871
                             0349   872 TALKER:                                             ; Talker timer has expired
                             0349   873             ;
                             0349   874             ;
                             0349   875             ;   Fork block on stack (ptr in R5) provides context for the next call.
                             0349   876             ;   The call to SOL_NW must be done with:
                             0349   877             ;
                             0349   878             ;
                             0349   879             ;   INPUTS:     R8      LPD address
                             0349   880             ;               R7,R6   Scratch
                             0349   881             ;               R5      Fork block address
                             0349   882             ;                       The FPC,FR3,FR4 fields are all scratch
                             0349   883             ;               R4      Scratch
                             0349   884             ;               R3      IRP address
                             0349   885             ;               R2      RCB address
                             0349   886             ;               R1,R0   Scratch
                             0349   887             ;
                             0349   888             ;   OUTPUTS:    R0      Status
                             0349   889             ;               R1,R4,R6,R7,R9 are destroyed.
                             0349   890             ;
                     59      D4  0349   891          CLRL    R9                              ; No adjacency required
                  014D       30  034B   892          BSBW    SOL_NW                          ; Get permission to xmit
                             034E   893                                                      ; -- don't wait if no resources
                   F7 50     E9  034E   894          BLBC    R0,EXIT                         ; If LBC, permission denied
                             0351   895          ASSUME  TR4$C_BCE_MID2 EQ 0
                             0351   896          ASSUME  TR4$C_BCR_MID2 EQ 0
                     44 A3   D4  0351   897          CLRL    IRP$Q_STATION+4(R3)             ; Clear high portion of address
                  54    52   D0  0354   898          MOVL    R2,R4                           ; Save RCB address
                51    2E A8   D0  0357   899          MOVL    LPD$L_RTR_LIST(R8),R1           ; Get ROUTER LIST
                        03    13  035B   900          BEQL    5$                              ; Br if none
                     51 61    9A  035D   901          MOVZBL  (R1),R1                         ; Else, get size of router list
51    00000070 8F    C0  0360   902 5$:     ADDL    #CXB$C_OVERHEAD-                ; Add in CXB size
                             0367   903                    +HELLO_MSG_SIZE,R1              ;  plus fixed size of hello msg
                             0367   904                                                    ; (this is worst case msg size)
                     0E0D    30  0367   905          BSBW    TR$ALLOCATE                     ; Allocate the buffer
                     DB 50   E9  036A   906          BLBC    R0,EXIT                         ; If LBC then failed
                  56    52   D0  036D   907          MOVL    R2,R6                           ; Setup CXB address
                51    48 A6   9E  0370   908          MOVAB   CXB$C_HEADER(R6),R1             ; Setup message ptr
                  57    51   D0  0374   909          MOVL    R1,R7                           ; Make a copy
                             0377   910             ;
                             0377   911             ;   If the circuit is a broadcast circuit, then the PTYPE in the
                             0377   912             ;   'main' ADJ is always unknown. Therefore on broadcast circuits
                             0377   913             ;   we will have to build either the Broadcast Hello message for
                             0377   914             ;   routers or endnodes. Otherwise, for non-broadcast circuits we
                             0377   915             ;   will build the hello message based upon the ADJ$B_PTYPE field.
                             0377   916             ;
3D 22 A8      0A    E0  0377   917          BBS     #LPD$V_BC,LPD$W_STS(R8),20$     ; Br if broadcast circuit, we
                             037C   918                                                    ;  will use LPD$B_ETY for case
                  50    20 A8   9A  037C   919          MOVZBL  LPD$B_PTH_INX(R8),R0            ; Get the ADJ index (same as
                             0380   920                                                    ;   LPD index)
               50    2C B440   D0  0380   921          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R0       ; Get ADJ address
                             0385   922          $DISPATCH ADJ$B_PTYPE(R0),TYPE=B,-      ; CASE on ADJ's node type
```

```
                        0385        <-                                              ;
                        0385                <ADJ$C_PTY_AREA 10$>,-                  ; Phase IV level 2 router
                        0385                <ADJ$C_PTY_PH4  10$>,-                  ; Phase IV router
                        0385                <ADJ$C_PTY_PH4N 10$>,-                  ; Phase IV endnode
                        0385                <ADJ$C_PTY_PH3  15$>,-                  ; Phase III router
                        0385                <ADJ$C_PTY_PH3N 15$>,-                  ; Phase III endnode
                        0385        >
                        0396        ;
                        0396        ;   Build a Phase II NOP message
                        0396        ;
          87  08   90   0396        MOVB    #TR3$C_MSG_NOP2,(R7)+                   ; Enter Phase II msg header
            00DD   31   0399        BRW     50$                                     ; Continue in common
                        039C  10$:  ;
                        039C        ;       Build a Phase IV non-broadcast hello message
                        039C        ;
       50  0E A4   B0   039C        MOVW    RCB$W_ADDR(R4),R0                       ; Get local node address
              06   11   03A0        BRB     17$                                     ; Continue in common code
                        03A2        ;
                        03A2        ;   Build a Phase III hello message
                        03A2        ;
              00   EF   03A2  15$:  EXTZV   #TR4$V_ADDR_DEST,-                      ; Get the local node address
   50   0E A4  0A       03A4                #TR4$S_ADDR_DEST,RCB$W_ADDR(R4),R0 ;..without area number
          87  05   90   03A8  17$:  MOVB    #TR3$C_MSG_HELLO,(R7)+                  ; Enter msg type
          87  50   B0   03AB        MOVW    R0,(R7)+                                ; Enter local node address
          87  02   90   03AE        MOVB    #2,(R7)+                                ; Enter count of next field
     87   AAAA 8F  B0   03B1        MOVW    #^X<AAAA>,(R7)+                         ; Enter alternating 1's and 0's
            00C0   31   03B6        BRW     50$                                     ; Done
                        03B9  20$:  ;
                        03B9        ;   Build a Phase IV Broadcast hello message
                        03B9        ;
       1D A8  05   91   03B9        CMPB    #ADJ$C_PTY_PH4N,LPD$B_ETY(R8)           ; Are we an endnode?
              6A   13   03BD        BEQL    40$                                     ; Br if yes
                        03BF        ;
                        03BF        ;   Build a Phase IV Broadcast router hello message
                        03BF        ;
          87  0B   90   03BF        MOVB    #TR4$C_MSG_BCRHEL,(R7)+                 ; Enter msg type
          87  02   B0   03C2        MOVW    #TR4$C_VER_LOWW,(R7)+                   ; Enter XPORT version number
          87  00   90   03C5        MOVB    #TR4$C_VER_HIB,(R7)+                    ;       ...
  87   000400AA 8F D0   03C8        MOVL    #TR4$C_HIORD,(R7)+                      ; Enter HIORD portion of addres
       87  0E A4   B0   03CF        MOVW    RCB$W_ADDR(R4),(R7)+                    ; Enter local node address
          87  02   90   03D3        MOVB    #TR4$C_RTR_LVL1,(R7)+                   ; Assume level 1 router
     03   1D A8    91   03D6        CMPB    LPD$B_ETY(R8),#ADJ$C_PTY_AREA           ; Are we a level 2 router?
              04   12   03DA        BNEQ    30$                                     ; Br if not
       FF A7  01   90   03DC        MOVB    #TR4$C_RTR_LVL2,-1(R7)                  ; Enter level 2 router type
       87  50 A8   B0   03E0  30$:  MOVW    LPD$W_BUFSIZ(R8),(R7)+                  ; Enter datalink buffer size
       87  2A A8   90   03E4        MOVB    LPD$B_BCPRI(R8),(R7)+                   ; Enter router's priority
              87   94   03E8        CLRB    (R7)+                                   ; RESERVED (AREA)
       87  18 A8   B0   03EA        MOVW    LPD$W_INT_TLK(R8),(R7)+                 ; Enter hello timer
                        03EE
       87  18 A8   90   03EE        MOVB    LPD$W_INT_TLK(R8),(R7)+                 ; && Put hello in reserved
                        03F2                                                        ; && until all are updated
       50  2E A8   D0   03F2        MOVL    LPD$L_RTR_LIST(R8),R0                   ; Get R/S list
     87  60   08   81   03F6        ADDB3   #8,(R0),(R7)+                           ; Store length of NI-LIST
              87   7C   03FA        CLRQ    (R7)+                                   ; RESERVED logical NI name
       FF A7  60   90   03FC        MOVB    (R0),-1(R7)                             ; Store length of R/S list
         007E 8F   BB   0400        PUSHR   #^M<R1,R2,R3,R4,R5,R6>                  ; Save registers
          56  80   9A   0404        MOVZBL  (R0)+,R6                                ; Get length of R/S list
```

```
                67   60   56   28  0407   980          MOVC    R6,(R0),(R7)                    ; Move the R/S list
                     57   56   C0  040B   981          ADDL    R6,R7                           ; Account for bytes moved
                      007E 8F   BA  040E   982          POPR    #^M<R1,R2,R3,R4,R5,R6>          ; Restore registers
                 030000AB 8F   D0  0412   983          MOVL    #TR4$C_BCR_MID1,-               ; Set destination address
                       40 A3         0418   984                   IRP$Q_STATION(R3)           ;  assume we have to send
                                     041A   985                                               ;  to "All Routers"
                                     041A   986          ;
                                     041A   987          ;   If we are the designated router on a Broadcast Circuit,
                                     041A   988          ;   then we will send the "Hello" message to "All Endnodes"
                                     041A   989          ;   after we have sent it to "All Routers".
                                     041A   990          ;
              5A 22 A8    0B   E1  041A   991          BBC     #LPD$V_XEND,LPD$W_STS(R8),50$   ; Br if we have not already
                                     041F   992                                               ;  sent the "Hello" message
                                     041F   993                                               ;  to "All Routers".
                 040000AB 8F   D0  041F   994          MOVL    #TR4$C_BCE_MID1,-               ; Else, Set destination address
                       40 A3         0425   995                   IRP$Q_STATION(R3)           ;  to "All Endnodes"
                          50   11   0427   996          BRB     50$                             ; Done
                                     0429   997          ;
                                     0429   998          ;   Build a Broadcast end node hello message
                                     0429   999          ;
                      87   0D   90  0429  1000 40$:     MOVB    #TR4$C_MSG_BCEHEL,(R7)+         ; Enter msg type
                      87   02   B0  042C  1001          MOVW    #TR4$C_VER_LOWW,(R7)+           ; Enter XPORT version number
                      87   00   90  042F  1002          MOVB    #TR4$C_VER_HIB,(R7)+            ;  ...
              87   000400AA 8F   D0  0432  1003          MOVL    #TR4$C_HIORD,(R7)+              ; Enter HIORD portion of addres
                    87   0E   A4  B0  0439  1004          MOVW    RCB$W_ADDR(R4),(R7)+            ; Enter local node address
                      87   03   90  043D  1005          MOVB    #TR4$C_END_NODE,(R7)+           ; Enter endnode type
                    87   50   A8  B0  0440  1006          MOVW    LPD$W_BUFSIZ(R8),(R7)+          ; Enter datalink buffer size
                         87   94   0444  1007          CLRB    (R7)+                           ; RESERVED (AREA)
                         87   7C   0446  1008          CLRQ    (R7)+                           ; Verification seed
              87   000400AA 8F   D0  0448  1009          MOVL    #TR4$C_HIORD,(R7)+              ; Store designated router's
                                     044F  1010                                               ;  HIORD portion of address
                    50   2C   A8  3C  044F  1011          MOVZWL  LPD$W_DRT(R8),R0                ; Get inx of designated router
                         09   13  0453  1012          BEQL    45$                             ; Br if none
                 50   2C B440   D0  0455  1013          MOVL    @RCB$L_PTR_ADJ(R4)[R0],R0       ; Get address of ADJ
                    50   04   A0  3C  045A  1014          MOVZWL  ADJ$W_PNA(R0),R0                ; Get designated router address
                      87   50   B0  045E  1015 45$:     MOVW    R0,(R7)+                        ; Set designated router address
                    87   18   A8  B0  0461  1016          MOVW    LPD$W_INT_TLK(R8),(R7)+         ; Enter hello timer
                                     0465  1017
                    87   18   A8  90  0465  1018          MOVB    LPD$W_INT_TLK(R8),(R7)+         ; && Put hello in reserved
                                     0469  1019                                               ; && until all are updated
                      87   02   90  0469  1020          MOVB    #2,(R7)+                        ; Enter count of next field
                 87   AAAA 8F   B0  046C  1021          MOVW    #^X<AAAA>,(R7)+                 ; Enter bit pattern
                  030000AB 8F   D0  0471  1022          MOVL    #TR4$C_BCR_MID1,-               ; Set destination address
                       40 A3         0477  1023                   IRP$Q_STATION(R3)           ;   to "All Routers"
                                     0479  1024
                                     0479  1025
                      57   51   C2  0479  1026 50$:     SUBL    R1,R7                           ; Setup message size
          0C A3    0E9A'CF   9E  047C  1027          MOVAB   W^TR$RTRN_XMT_TLK,IRP$L_PID(R3) ; Setup end-action address
                52    8B'AF   9E  0482  1028          MOVAB   B^60$,R2                        ; Setup null end-action routine
                         54   D4  0486  1029          CLRL    R4                              ; No "quick solicit" wanted
                      50   01   9A  0488  1030          MOVZBL  #1,R0                           ; Return success
                                05  048B  1031 60$:     RSB                                     ; Return with status in R0
                                     048C  1032
```

```
                        048C  1034          .SBTTL  TR$SOLICIT      - Process ECL request to xmit into the network
                        048C  1035
                        048C  1036 ;+
                        048C  1037 ;  TR$SOLICIT   - Process ECL request to xmit into the network
                        048C  1038 ;
                        048C  1039 ;
                        048C  1040 ;  An ECL (e.g. NSP) is requesting to xmit into the network.  The appropriate
                        048C  1041 ;  logical path (LPD) is found, either because it was explicitly specified or
                        048C  1042 ;  because the specified destination node address maps to it.
                        048C  1043 ;
                        048C  1044 ;  If the resources for transmission (input packet limiter queue slot, square
                        048C  1045 ;  root packet limit queue slot, IRP) are not immediately available, the
                        048C  1046 ;  request block is entered onto a wait queue.
                        048C  1047 ;
                        048C  1048 ;
                        048C  1049 ;  INPUTS:       R5      Fork block address
                        048C  1050 ;                          The FPC,FR3,FR4 fields are all scratch and must not
                        048C  1051 ;                          be modified by the caller until it is reactivated by
                        048C  1052 ;                          either  TR$DENY  or  TR$GRANTED.
                        048C  1053 ;                  R4      Destination node address
                        048C  1054 ;                          Zero if Transport is to use the LPD index as ADJ index
                        048C  1055 ;                  R3      I.D. of LPD to xmit over
                        048C  1056 ;                          Zero if Transport is to choose the LPD
                        048C  1057 ;                  R2      RCB address
                        048C  1058 ;                  R1,R0   Scratch
                        048C  1059 ;
                        048C  1060 ;                   (SP)   Return address of caller
                        048C  1061 ;                 4(SP)    Return address of caller's caller
                        048C  1062 ;
                        048C  1063 ;
                        048C  1064 ;  OUTPUTS:      See parameters returned when reactivating process from
                        048C  1065 ;                routines TR$GRANT or TR$DENY
                        048C  1066 ;
                        048C  1067 ;-
                        048C  1068 TR$SOLICIT::                                 ; Process ECL request to xmit
                        048C  1069                  ;
                        048C  1070                  ;
                        048C  1071                  ;   Setup the fork block and pop the stack to simplify the code
                        048C  1072                  ;   in case the requestor needs to be suspended.
                        048C  1073                  ;
                        048C  1074                  ;
              0C A5 8ED0  048C  1075          POPL    FKB$L_FPC(R5)               ; Save return addr, pop stack
                        0490  1076                  ;
            07C0 8F   BB  0490  1077          PUSHR   #^M<R6,R7,R8,R9,R10>        ; Save reg used for LPD address
                  30  10  0494  1078          BSBB    SOL_WAIT                    ; Process request, okay to wait
            07C0 8F   BA  0496  1079          POPR    #^M<R6,R7,R8,R9,R10>        ; Restore reg
                        049A  1080                  ;
                  05    049A  1081          RSB                                 ; Done
                        049B  1082
                        049B  1083 SOL_NW:                                      ; Solicit - do not wait
                        049B  1084                  ;
                        049B  1085                  ;
                        049B  1086                  ;   Setup the IRP for eventual xmission.
                        049B  1087                  ;
                        049B  1088                  ;
              0C A5 8ED0  049B  1089          POPL    FKB$L_FPC(R5)               ; Setup return address
10 A5         20 A8   3C  049F  1090          MOVZWL  LPD$W_PTH(R8),FKB$L_FR3(R5) ; Save LPD i.d.
```

```
                1F A8   95  04A4  1091          TSTB    LPD$B_XMT_IPL(R8)       ; Does "input-packet-limiter" allow it
                  13    15  04A7  1092          BLEQ    30$                     ; If LEQ then no, DENY request
                1C A8   91  04A9  1093          CMPB    LPD$B_IRPCNT(R8),-      ; Does "square-root-limiter" allow it
                1E A8       04AC  1094                  LPD$B_XMT_SRL(R8)       ;
                  7A    14  04AE  1095          BGTR    TR$DENY                 ; If GTR then no, DENY request
            53  00 B2     0F  04B0  1096  20$:   REMQUE  @RCB$Q_IRP_FREE(R2),R3  ; Get a free IRP
                  08    1C  04B4  1097          BVC     40$                     ; If VC then got one
                 0C4D   30  04B6  1098          BSBW    TR$ADJUST_IRP           ; Adjust IRP count if possible
                F4 50   E8  04B9  1099          BLBS    R0,20$                  ; Br if any new IRPs were allocated
                  6C    11  04BC  1100  30$:   BRB     TR$DENY                 ; Else, deny permission to xmit
                            04BE  1101                                          ;
                1F A8   97  04BE  1102  40$:   DECB    LPD$B_XMT_IPL(R8)       ; Consume "input-packet-limit" slot
                1C A8   96  04C1  1103          INCB    LPD$B_IRPCNT(R8)        ; Account for IRP to be queued
                  6F    11  04C4  1104          BRB     TR$GRANT                ; Grant permission to xmit
                            04C6  1105
                            04C6  1106  SOL_WAIT:                               ; Process request, okay to wait
                 00D0   30  04C6  1107          BSBW    TR$GET_ADJ              ; Get AJD and LPD for output
                5E 50   E9  04C9  1108          BLBC    R0,TR$DENY              ; Br if no path to node
                  54    B5  04CC  1109          TSTW    R4                      ; Zero destination?
                  24    13  04CE  1110          BEQL    50$                     ; Br if yes, okay to send
                            04D0  1111          ;
                            04D0  1112          ;   If we are endnode, and we are connected to another endnode,
                            04D0  1113          ;   then make sure the endnode's address is the same as the
                            04D0  1114          ;   destination address.  If not, deny the request.  This ensures
                            04D0  1115          ;   that the remote endnode only receives packets destined for him.
                            04D0  1116          ;
                            04D0  1117          $DISPATCH ADJ$B_PTYPE(R9),TYPE=B,-
                            04D0  1118          <-
                            04D0  1119                  <ADJ$C_PTY_PH4N 20$>,-  ; Phase IV endnode
                            04D0  1120                  <ADJ$C_PTY_PH3N 10$>,-  ; Phase III endnode
                            04D0  1121          >
                      13    11  04DF  1122          BRB     50$                     ; Otherwise continue
                      00    EF  04E1  1123  10$:   EXTZV   #TR4$V_ADDR_DEST,-      ; For Phase III nodes,
                50    54    0A      04E3  1124                  #TR4$S_ADDR_DEST,R4,R0  ; compare only the node addr, not area
                  04 A9     50  B1  04E6  1125          CMPW    R0,ADJ$W_PNA(R9)        ; Is the destination node correct?
                            3E  12  04EA  1126          BNEQ    TR$DENY                 ; Br if no, deny request
                            06  11  04EC  1127          BRB     50$
                  04 A9     54  B1  04EE  1128  20$:   CMPW    R4,ADJ$W_PNA(R9)        ; Is the destination node correct?
                            36  12  04F2  1129          BNEQ    TR$DENY                 ; Br if no, deny request
              10 A5   20 A8     3C  04F4  1130  50$:   MOVZWL  LPD$W_PTH(R8),FKB$L_FR3(R5) ; Save LPD i.d.
                  14 A5     57  D0  04F9  1131          MOVL    R7,FKB$L_FR4(R5)        ; Save ADJ index if we have to FORK
                            04FD  1132                                              ;
                            04FD  1133
                            04FD  1134  QUICK_SOL:                                  ; Quick solicit entry
                            04FD  1135          ;
                            04FD  1136          ;   Make sure the NETACP is still active before actually granting
                            04FD  1137          ;   permission to transmit.
                            04FD  1138          ;
                      01    E1  04FD  1139          BBC     #RCB$V_ACT,-            ; If ACP is not active, then return
                  28 0B A2        04FF  1140                  RCB$B_STATUS(R2),TR$DENY;   failure to caller
                            0502  1141          ;
                            0502  1142          ;
                            0502  1143          ;   Need "request" slot, room on output queue, and IRP to proceed
                            0502  1144          ;
                            0502  1145          ;
                1F A8   95  0502  1146          TSTB    LPD$B_XMT_IPL(R8)       ; Does "input-packet-limiter" allow it?
                  1E    15  0505  1147          BLEQ    70$                     ; If LEQ then no
```

```
                1C A8   91  0507  1148          CMPB    LPD$B_IRPCNT(R8),-      ; Does "square-root-limiter" allow it?
                1E A8       050A  1149                  LPD$B_XMT_SRL(R8)       ;
                   17   14  050C  1150          BGTR    70$                     ; If GTR then no
                1F A8   97  050E  1151          DECB    LPD$B_XMT_IPL(R8)       ; Consume request slot
                1C A8   96  0511  1152          INCB    LPD$B_IRPCNT(R8)        ; Account for IRP to be queued
53              00 B2   0F  0514  1153 60$:     REMQUE  @RCB$Q_IRP_FREE(R2),R3  ; Get a free IRP
                   1B   1C  0518  1154          BVC     TR$GRANT                ; If VC then got one
                 0BE9   30  051A  1155          BSBW    TR$ADJUST_IRP           ; Adjust IRP count if possible
                F4 50   E8  051D  1156          BLBS    R0,60$                  ; If LBS, IRPs were allocated
                            0520  1157                                          ;
50 B2              65   0E  0520  1158          INSQUE  (R5),@RCB$Q_IRP_WAIT+4(R2)  ; Wait for IRP
                        05  0524  1159          RSB
                            0525  1160
04 B8              65   0E  0525  1161 70$:     INSQUE  (R5),@LPD$Q_REQ_WAIT+4(R8)  ; Wait for spot on datalink queue
                        05  0529  1162          RSB
                            052A  1163
```

```
                     052A  1165          .SBTTL  TR$DENY         - Deny solicitor permission to transmit
                     052A  1166          .SBTTL  TR$GRANT        - Grant solicitor permission to transmit
                     052A  1167
                     052A  1168 ;+
                     052A  1169 ;  TR$DENY   -  Reactivate solicitor, denying permission to transmit
                     052A  1170 ;  TR$GRANT  -  Reactivate solicitor, granting permission to transmit
                     052A  1171 ;
                     052A  1172 ;
                     052A  1173 ;  The R5 fork process cannot be suspended beyond this point.
                     052A  1174 ;
                     052A  1175 ;
                     052A  1176 ;  INPUTS:      R10     Scratch
                     052A  1177 ;                R9      ADJ address
                     052A  1178 ;                        Or ZERO if called by TALKER routine
                     052A  1179 ;                R8      LPD address
                     052A  1180 ;                R7,R6   Scratch
                     052A  1181 ;                R5      Fork block address
                     052A  1182 ;                R4      Scratch
                     052A  1183 ;                R3      If TR$GRANT  - IRP address
                     052A  1184 ;                        If TR$DENY   - Scratch
                     052A  1185 ;                R2      RCB address
                     052A  1186 ;                R1,R0   Scratch
                     052A  1187 ;
                     052A  1188 ;
                     052A  1189 ;  OUTPUTS:     R7-R0   Garbage
                     052A  1190 ;
                     052A  1191 ;                All other registers are preserved.
                     052A  1192 ;
                     052A  1193 ;-
                     052A  1194 TR$DENY:                                     ; Deny permisson to xmit
          50    94   052A  1195          CLRB    R0                       ; Indicate request denied
          52    DD   052C  1196          PUSHL   R2                       ; Save RCB address
       0C B5    16   052E  1197          JSB     @FKB$L_FPC(R5)           ; Tell requestor
          52 8ED0    0531  1198          POPL    R2                       ; Restore RCB address
             05      0534  1199          RSB                              ; Done
                     0535  1200
                     0535  1201 TR$GRANT:                                  ; Grant permission to xmit
                     0535  1202          ;
                     0535  1203          ;  Call requestor back with:
                     0535  1204          ;
                     0535  1205          ;       R10     Scratch
                     0535  1206          ;       R9      ADJ address or zero
                     0535  1207          ;       R8      LPD address
                     0535  1208          ;       R7,R6   Scratch
                     0535  1209          ;       R5      Fork block address
                     0535  1210          ;       R4      Scratch
                     0535  1211          ;       R3      IRP address only if R0 has low bit set, else scratch
                     0535  1212          ;       R2      RCB address
                     0535  1213          ;       R1      Scratch
                     0535  1214          ;       R0      Low bit set if permission granted
                     0535  1215          ;               Low bit clear if permission denied
                     0535  1216          ;
                     0535  1217          ;
                     0535  1218          ASSUME  IRP$L_AST       EQ  4+IRP$L_PID
                     0535  1219          ASSUME  IRP$L_ASTPRM    EQ  4+IRP$L_AST
                     0535  1220
     50    0C A3    9E   0535  1221          MOVAB   IRP$L_PID(R3),R0         ; Setup R4 for IRP builder
```

```
                80   0F06'CF   9E  0539  1222          MOVAB   W^TR$RTRN_XMT_ECL,(R0)+ ; Setup end-action address
                  80    20 A8   9A  053E  1223          MOVZBL  LPD$B_PTH_INX(R8),(R0)+ ; Enter LPD index
                      80    52  D0  0542  1224          MOVL    R2,(R0)+                ; Enter RCB address
                      50    01  90  0545  1225          MOVB    #1,R0                   ; Indicate "okay to xmit"
                        0C B5   16  0548  1226          JSB     @FKB$L_FPC(R5)          ; Reactivate solicitor
                                    054B  1227          ;
                                    054B  1228          ;
                                    054B  1229          ;   On return, the CXB and registers are setup as follows:
                                    054B  1230          ;
                                    054B  1231          ;
                                    054B  1232          ;   +---------------+
                                    054B  1233          ;   |   standard    |   11 bytes long.  CXB$L_FLINK and CXB$L_BLINK may
                                    054B  1234          ;   |     VMS       |   be used by the Transport layer.  CXB$W_SIZE
                                    054B  1235          ;   | buffer header |   must be correct.  CXB$B_TYPE must be DYN$C_CXB.
                                    054B  1236          ;   +---------------+
                                    054B  1237          ;   |               |   Starts with CXB$B_CODE (byte 11) and continues
                                    054B  1238          ;   |      ECL      |   to CXB$C_LENGTH.  This area is read-only to
                                    054B  1239          ;   |   pure area   |   Transport and below.  It cannot even be
                                    054B  1240          ;   |               |   saved/restored.
                                    054B  1241          ;   +---------------+
                                    054B  1242          ;   |    Datalink  |   Starts at CXB$C_LENGTH and is at least
                                    054B  1243          ;   |     Layer     |   CXB$C_DLL bytes long.  Used by the datalink for
                                    054B  1244          ;   |  impure area  |   protocol header or state information.
                                    054B  1245          ;   +---------------+
                                    054B  1246          ;   |               |
                                    054B  1247          ;   .    body of    .   Must be quadword aligned and starting no sooner
                                    054B  1248          ;   .    message    .   than CXB$C_LENGTH + CXB$C_DLL (= CXB$C_HEADER)
                                    054B  1249          ;   .               .   The first 6 bytes contain: RTFLG,DSTNOD,SRCNOD
                                    054B  1250          ;   .               .   FORWARD, in that order.
                                    054B  1251          ;   |               |
                                    054B  1252          ;   +---------------+
                                    054B  1253          ;   |    Datalink  |   Used by the datalink layer for protocol (e.g.,
                                    054B  1254          ;   |     Layer     |   checksum) or state information.  Must be at
                                    054B  1255          ;   |  impure area  |   least CXB$C_TRAILER in length.
                                    054B  1256          ;   +---------------+
                                    054B  1257          ;
                                    054B  1258          ;
                                    054B  1259          ;       R9      ADJ address or zero
                                    054B  1260          ;       R8      LPD address
                                    054B  1261          ;       R7      Size of message
                                    054B  1262          ;       R6      CXB address
                                    054B  1263          ;       R5      Garbage
                                    054B  1264          ;       R4      0 if "quick solicit" not requested
                                    054B  1265          ;               Else, pointer to request block (XWB fork block) with
                                    054B  1266          ;               FRK$L_FPC  pointing to the "quick solicit" routine
                                    054B  1267          ;       R3      IRP address -- unmodified from call
                                    054B  1268          ;       R2      Address of End-action routine to call on I/O competion
                                    054B  1269          ;       R1      Ptr to 1st byte in standard Phase III route-header
                                    054B  1270          ;       R0      Low bit set   - if message is to be xmitted
                                    054B  1271          ;               Low bit clear - if no message to xmit.  In this case
                                    054B  1272          ;                               R7-R4,R2,R1 contain garbage.
                                    054B  1273          ;
                                    054B  1274          ;
                      26 50     E9  054B  1275          BLBC    R0,60$                  ; If LBC then xmit aborted
                78 A3    52     D0  054E  1276          MOVL    R2,IRP$L_SAVD_RTN(R3)   ; Save ptr to End-action routine
                    3A A6       B4  0552  1277          CLRW    CXB$W_R_ADJ(R6)         ; No receive adjacency
                    32 A6       B4  0555  1278          CLRW    CXB$W_R_PATH(R6)        ; No receive LPD
```

```
                52   14 A3    D0  0558  1279          MOVL    IRP$L_ASTPRM(R3),R2      ; Recover RCB address
                                  055C  1280
                50   18 A3    9E  055C  1281          MOVAB   IRP$L_WIND(R3),R0        ; Setup R0 for building IRP
                     55   54   D0  0560  1282          MOVL    R4,R5                    ; "Quick solicit" requested ?
                          03   12  0563  1283          BNEQ    50$                      ; If NEQ then yes
                            07A9 31  0565  1284          BRW     FINISH_XMT_HDR           ; Finish building HDR & IRP, xmit it.
                                  0568  1285                                           ;
                               55 DD  0568  1286 50$:    PUSHL   R5                       ; Remember block's address
                               52 DD  056A  1287          PUSHL   R2                       ; Remember RCB address
                            07A2 30  056C  1288          BSBW    FINISH_XMT_HDR           ; Finish building HDR & IRP, xmit it.
                              24 BA  056F  1289          POPR    #^M<R2,R5>               ; Setup R2,R5 (R8 points to LPD)
                                  0571  1290                                           ;
                            FF89 31  0571  1291          BRW     QUICK_SOL                ; Perform "quick solicit"
                                  0574  1292 60$:       ;
                                  0574  1293            ;
                                  0574  1294            ;   User didn't want to xmit after all.  Return resources.
                                  0574  1295            ;
                                  0574  1296            ;
                     55   53   D0  0574  1297          MOVL    R3,R5                    ; Setup IRP address
                        24 A5 D4  0577  1298          CLRL    IRP$L_IOSB(R5)           ; No buffer to deallocate
                38 A5    01   D0  057A  1299          MOVL    #1,IRP$L_IOST1(R5)       ; Avoid false I/O failure detection
                52   14 A5    D0  057E  1300          MOVL    IRP$L_ASTPRM(R5),R2      ; Recover RCB address
                     1F A8    96  0582  1301          INCB    LPD$B_XMT_IPL(R8)        ; Return "request" slot
                            09D6 30  0585  1302          BSBW    TR_RTRN_IRP              ; Recycle unused the IRP
                                 05  0588  1303          RSB                              ; Done
```

```
                                                D 4
                     0589  1305          .SBTTL  TR$TEST_REACH   - Check if node is reachable
                     0589  1306 ;+
                     0589  1307 ;  TR$TEST_REACH  -  Check if node is reachable
                     0589  1308 ;
                     0589  1309 ;
                     0589  1310 ;  INPUTS:       R0      Remote/Local node address
                     0589  1311 ;                R2      RCB address
                     0589  1312 ;
                     0589  1313 ;  OUTPUTS:      R0      True if th      path to node available
                     0589  1314 ;                        False if       vailable to node
                     0589  1315 ;
                     0589  1316 ;                All registers are preserved.
                     0589  1317 ;-
                     0589  1318 TR$TEST_REACH::
03FA 8F    BB        0589  1319          PUSHR   #^M<R1,R3,R4,R5,R6,R7,R8,R9>     ; Save registers
54   50    D0        058D  1320          MOVL    R0,R4                           ; Pass node address
     53    D4        0590  1321          CLRL    R3                              ; No specific circuit
     05    10        0592  1322          BSBB    TR$GET_ADJ                      ; Get the output ADJ
03FA 8F    BA        0594  1323          POPR    #^M<R1,R3,R4,R5,R6,R7,R8,R9>    ; Restore registers
     05    05        0598  1324          RSB                                     ; Return to caller
```

```
                                                E 4
0599  1326          .SBTTL  TR$GET_ADJ      - Get output ADJ and LPD
0599  1327
0599  1328 ;+
0599  1329 ;  TR$GET_ADJ                    Get output ADJ and LPD
0599  1330 ;
0599  1331 ;
0599  1332 ;  INPUTS:       R4      Remote/Local node address or zero
0599  1333 ;               R3      LPD index or zero
0599  1334 ;               R2      RCB address
0599  1335 ;
0599  1336 ;
0599  1337 ;  OUTPUTS:      R9      ADJ address (zero if none)
0599  1338 ;               R8      LPD address (zero if none)
0599  1339 ;               R0      True if path available to node, false if unreachable
0599  1340 ;
0599  1341 ;               R7-R6,R1    are destroyed.
0599  1342 ;               R5-R2       are preserved.
0599  1343 ;-
0599  1344          .ENABL  LSB
0599  1345 TR$GET_ADJ::                                ; Get the output ADJ and LPD
0599  1346          ;
0599  1347          ;   Determine the LPD address from the path i.d. in the low byte of
0599  1348          ;   R3.  If the path i.d. is zero then determine output LPD from the
0599  1349          ;   destination node address.
0599  1350          ;
0599  1351          ;       CASES:          LPD     NODE
0599  1352          ;                       ---     ----
0599  1353          ;               NORMAL: R3 = 0, R4 = Destination node address or zero
0599  1354          ;               NORMAL: R3 <>0, R4 = Remote node address
0599  1355          ;               LOOP:   R3 <>0, R4 = Local node address
0599  1356          ;
     57   53   9A  0599  1357          MOVZBL  R3,R7                   ; Assume we need the LPD index
059C  1358                                                              ;  as the ADJ index
       03   12  059C  1359          BNEQ    10$                     ; Br if LOOP case - R3 is non-zero
     00C0   31  059E  1360          BRW     130$                    ; Else, NORMAL case
05A1  1361
       54   B5  05A1  1362 10$:     TSTW    R4                      ; Is the node address given?
       03   12  05A3  1363          BNEQ    20$                     ; Br if yes - LOOP NODE case
     0146   31  05A5  1364          BRW     240$                    ; Else, use LPD as ADJ
05A8  1365
 0E A2 54   B1  05A8  1366 20$:     CMPW    R4,RCB$W_ADDR(R2)       ; Is this intended for loopback?
       38   13  05AC  1367          BEQL    50$                     ; If so, then LOOP NODE request
05AE  1368          ;
05AE  1369          ;   Forced-LPD normal case - we are going to transmit a message
05AE  1370          ;   to a specific remote node, over a specific LPD.
05AE  1371          ;
05AE  1372          $DISPATCH RCB$B_ETY(R2),TYPE=B,- ; If we are an endnode, use DRT
05AE  1373          <-                              ;
05AE  1374                  <ADJ$C_PTY_PH3N 120$>,- ; Phase III endnode
05AE  1375                  <ADJ$C_PTY_PH4N 120$>,- ; Phase IV endnode
05AE  1376          >
05BE  1377
05BE  1378          ;
05BE  1379          ;   First we MUST find the output ADJ based on the node address given
05BE  1380          ;   the destination node address supplied in R4.
05BE  1381          ;
05BE  1382          ;   Determine the output LPD from the output adjacency.
```

```
                     05BE  1383          ;
          0A   EF    05BE  1384          EXTZV   #TR4$V_ADDR_AREA,-      ; Get the "Area" portion of the
          06         05C0  1385                  #TR4$S_ADDR_AREA,-      ;  node address
     51   54         05C1  1386                  R4,R1                   ;
          00   EF    05C3  1387          EXTZV   #TR4$V_ADDR_DEST,-      ; Get only the destination
          0A         05C5  1388                  #TR4$S_ADDR_DEST,-      ;  portion of the node address
     57   54         05C6  1389                  R4,R7                   ;
          51   95    05C8  1390          TSTB    R1                      ; Is this for area 0?
          07   13    05CA  1391          BEQL    30$                     ; Br if yes - our "logical" area
008B C2   51   91    05CC  1392          CMPB    R1,RCB$B_HOMEAREA(R2)   ; Is this request for our "Area"?
          10   12    05D1  1393          BNEQ    40$                     ; Br if no, deny request
  5A A2   57   B1    05D3  1394  30$:     CMPW    R7,RCB$W_MAX_ADDR(R2)   ; Is the node within bounds?
          0A   1A    05D7  1395          BGTRU   40$                     ; If GTRU then no
57   1C B247   3C    05D9  1396          MOVZWL  @RCB$L_PTR_OA(R2)[R7],R7 ; Get ADJ index
          03   13    05DE  1397          BEQL    40$                     ; Br if not there, deny request
        010B   31    05E0  1398          BRW     240$                    ; Else, continue processing
        0125   31    05E3  1399  40$:     BRW     NOT_REACH               ; Else, node unreachable
                     05E6  1400
                     05E6  1401  50$:     ;
                     05E6  1402          ;   LOOP NODE case - we are going to transmit a message to
                     05E6  1403          ;   a remote node over the LPD, but with the destination address
                     05E6  1404          ;   set to ourself so it will be looped back.
                     05E6  1405          ;
                     05E6  1406          $DISPATCH RCB$B_ETY(R2),TYPE=B,- ; Br if we are an endnode
                     05E6  1407          <-                      ;
                     05E6  1408                  <ADJ$C_PTY_PH3N 120$>,- ; Phase III endnode
                     05E6  1409                  <ADJ$C_PTY_PH4N 120$>,- ; Phase IV endnode
                     05E6  1410          >
                     05F6  1411          ;
                     05F6  1412          ;  For the LOOP case, we will first try the DRT for the LPD that
                     05F6  1413          ;  was passed down from the requesting process. If the LPD is a
                     05F6  1414          ;  BC and the DRT is not set then we must scan the entire BRA
                     05F6  1415          ;  ADJ list to find the first remote TRANSPORT which can do the loop
                     05F6  1416          ;  for us. Else, for non-BC circuits, we will use the DRT value as
                     05F6  1417          ;  given.
                     05F6  1418          ;
                     05F6  1419          ;  Also, if the DRT is set and we are the DRT for the LPD, then we
                     05F6  1420          ;  will have to scan the BRA list for a remote transport to talk to.
                     05F6  1421          ;
                     05F6  1422          ;  Inputs:
                     05F6  1423          ;               R7 = LPD index (zero extended)
                     05F6  1424          ;               R4 = Node address
                     05F6  1425          ;               R3 = LPD index
                     05F6  1426          ;
 58   28 B247   D0    05F6  1427          MOVL    @RCB$L_PTR_LPD(R2)[R7],R8 ; Get LPD address
           4C   18    05FB  1428          BGEQ    100$                    ; If GEQ then slot not in use
                     05FD  1429          ASSUME  LPD$V_ACTIVE  EQ  0
     48 22 A8   E9    05FD  1430          BLBC    LPD$W_STS(R8),100$      ; If LBC, circuit is inactive
                     0601  1431          ;
                     0601  1432          ;       We must now check to see if the DRT is ourself and if so,
                     0601  1433          ;       then we must try to find someone else to loop with. If we
                     0601  1434          ;       cannot find someone else to loop with, then we must try using
                     0601  1435          ;       the "main" LPD and hope we are in loopback.
                     0601  1436          ;
   50   2C A8   3C    0601  1437          MOVZWL  LPD$W_DRT(R8),R0        ; Get the designated router ADJ index
 50   2C B240   D0    0605  1438          MOVL    @RCB$L_PTR_ADJ(R2)[R0],R0 ; Get ADJ address
0E A2   04 A0   B1    060A  1439          CMPW    ADJ$W_PNA(R0),RCB$W_ADDR(R2) ; Are we the "Designated Router"?
```

```
                 12   13  060F  1440          BEQL    70$                             ; Br if yes, try to find someone else
          57  2C A8   B1  0611  1441          CMPW    LPD$W_DRT(R8),R7                ; Is the DRT set?
                       0615  1442                                                      ; (i.e. Not equal to "main" LPD)
                 05   12  0615  1443          BNEQ    60$                             ; Br if yes - use DRT
       07 22 A8  0A   E0  0617  1444          BBS     #LPD$V_BC,LPD$W_STS(R8),70$ ; Else, scan BRAs if NI
          57  2C A8   3C  061C  1445 60$:     MOVZWL  LPD$W_DRT(R8),R7                ; Get ADJ index for output ADJ
                00CB   31  0620  1446          BRW     240$                            ; Go get ADJ and LPD addresses
                       0623  1447
          56  6A A2   3C  0623  1448 70$:     MOVZWL  RCB$W_MAX_RTG(R2),R6    ; Get number of routing "destinations"
                 19   13  0627  1449          BEQL    90$                             ; Br if none - try "main" LPD
          57  5C A2   9A  0629  1450          MOVZBL  RCB$B_MAX_LPD(R2),R7    ; Initialize ADJ index
                 0F   11  062D  1451          BRB     80$                             ; Start at first ADJ
       59  2C B247   D0  062F  1452 75$:     MOVL    @RCB$L_PTR_ADJ(R2)[R7],R9 ; Get next ADJ address
                 01   E1  0634  1453          BBC     #ADJ$V_RUN,-                    ; Br if ADJ not in run state
              06 69       0636  1454                  ADJ$B_STS(R9),80$               ;
          02 A9  53   91  0638  1455          CMPB    R3,ADJ$B_LPD_INX(R9)    ; LPD match?
                 0E   13  063C  1456          BEQL    110$                            ; Br if YES
       ED 57     56   F3  063E  1457 80$:     AOBLEQ  R6,R7,75$                       ; Br if more
          57  20 A8   9A  0642  1458 90$:     MOVZBL  LPD$B_PTH_INX(R8),R7    ; If all else fails, use "main" ADJ
                00A5   31  0646  1459          BRW     240$                            ; Get ADJ and LPD addresses
                       0649  1460
                00BF   31  0649  1461 100$:    BRW     NOT_REACH                       ; DENY - if no remote transport
                       064C  1462          ;
                       064C  1463          ;   Found remote transport to loop with, get LPD address
                       064C  1464          ;
                00A4   31  064C  1465 110$:    BRW     260$                            ; Get LPD address and continue
                       064F  1466
                       064F  1467 120$:    ;
                       064F  1468          ;   For LOOP Endnodes we will ALWAYS use LPD$W_DRT for output
                       064F  1469          ;
    58  28 B247   D0  064F  1470          MOVL    @RCB$L_PTR_LPD(R2)[R7],R8 ; Get LPD address
                 43   18  0654  1471          BGEQ    160$                            ; If GEQ then slot not in use
                       0656  1472          ASSUME  LPD$V_ACTIVE  EQ  0
          3F 22 A8   E9  0656  1473          BLBC    LPD$W_STS(R8),160$      ; If LBC, circuit is inactive
                       065A  1474
          57  2C A8   3C  065A  1475          MOVZWL  LPD$W_DRT(R8),R7        ; Get ADJ index for output ADJ
                008D   31  065E  1476          BRW     240$                            ; Continue in common path
                       0661  1477
                       0661  1478          ;
                       0661  1479          ;
                       0661  1480          ;   NORMAL transmit request
                       0661  1481          ;
                       0661  1482          ;
                 0A    EF   0661  1483 130$:    EXTZV   #TR4$V_ADDR_AREA,-      ; Get the "Area" portion of the
                 06         0663  1484                  #TR4$S_ADDR_AREA,-      ;  node address
               51  54       0664  1485                  R4,R1                   ;
                       0666  1486          ASSUME  TR4$V_ADDR_DEST EQ 0
                 00    EF   0666  1487          EXTZV   #TR4$V_ADDR_DEST,-      ; Get only the destination
                 0A         0668  1488                  #TR4$S_ADDR_DEST,-      ;  portion of the node address
               57  54       0669  1489                  R4,R7                   ;
                       066B  1490          $DISPATCH RCB$B_ETY(R2),TYPE=B,- ; Dispatch on Our Node type
                       066B  1491          <-
                       066B  1492                  <ADJ$C_PTY_PH4N  SOL_PH4N>,-    ; Phase IV endnode
                       066B  1493                  <ADJ$C_PTY_AREA  SOL_AREA>,-    ; Phase IV Level 2 router
                       066B  1494          >
                       0677  1495
                       0677  1496                                          ; All other - including Level 1 Router
```

```
                     0677  1497 SOL_PH4:                                ; Phase IV Level 1 Router request
                     0677  1498 ;
                     0677  1499 ;   First we MUST find the output ADJ based on the node address given
                     0677  1500 ;   the destination node address supplied in R4.
                     0677  1501 ;
                     0677  1502 ;   Determine the output LPD from the output adjacency.
                     0677  1503 ;
           51   95  0677  1504         TSTB    R1                          ; Is this for area 0?
           11   13  0679  1505         BEQL    140$                        ; Br if yes - our "logical" area
008B C2    51   91  067B  1506         CMPB    R1,RCB$B_HOMEAREA(R2)       ; Is this request for our "Area"?
           0A   13  0680  1507         BEQL    140$                        ; Br if yes
57   00AC C2    3C  0682  1508         MOVZWL  RCB$W_LVL2(R2),R7           ; Else, get the nearest Level 2 router
           65   12  0687  1509         BNEQ    240$                        ; Br if okay - we have one
         007F   31  0689  1510         BRW     NOT_REACH                   ; Else, node unreachable
                     068C  1511 ;
  5A A2    57   B1  068C  1512 140$:   CMPW    R7,RCB$W_MAX_ADDR(R2)       ; Is the node within bounds?
           07   1A  0690  1513         BGTRU   160$                        ; If GTRU then no
57   1C B247    3C  0692  1514         MOVZWL  @RCB$L_PTR_0A(R2)[R7],R7    ; Get ADJ index
           55   12  0697  1515         BNEQ    240$                        ; Br if okay
         006F   31  0699  1516 160$:   BRW     NOT_REACH                   ; Else, node unreachable
                     069C  1517 ;
                     069C  1518 SOL_PH4N:                                ; Process Phase IV endnode request
                     069C  1519 ;
                     069C  1520 ;   For Endnodes, we will first scan the CACHE to see if the
                     069C  1521 ;   destination node is directly adjacent, and if so send it direct.
                     069C  1522 ;   Otherwise we will ALWAYS use RCB$W_DRT for output (ignoring R4).
                     069C  1523 ;
                     069C  1524 ;   Note that RCB$W_DRT is always guaranteed to be either the ADJ
                     069C  1525 ;   index of the "designated" router or the ADJ index of the LPD's
                     069C  1526 ;   "main" adjacency.
                     069C  1527 ;
                     069C  1528 ;       Try the CACHE first! The CACHE is pointed to by the LPD,
                     069C  1529 ;       we find the LPD to scan from RCB$W_DRT.
                     069C  1530 ;
     57    01   9A  069C  1531         MOVZBL  #LPD$C_LOC_INX,R7           ; Assume we use the "local" LPD
  0E A2    54   B1  069F  1532         CMPW    R4,RCB$W_ADDR(R2)           ; Is destination node address ourself?
           49   13  06A3  1533         BEQL    240$                        ; Br if yes - use "local" LPD
57   00AA C2    3C  06A5  1534         MOVZWL  RCB$W_DRT(R2),R7            ; Get ADJ index for output ADJ
           ED   13  06AA  1535         BEQL    160$                        ; Br if none available - DENY
59   2C B247    D0  06AC  1536         MOVL    @RCB$L_PTR_ADJ(R2)[R7],R9   ; Get ADJ address
  58     02 A9    9A  06B1  1537         MOVZBL  ADJ$B_LPD_INX(R9),R8        ; Get LPD index for this adjacency
58   28 B248    D0  06B5  1538         MOVL    @RCB$L_PTR_LPD(R2)[R8],R8   ; Get LPD address
           DD   18  06BA  1539         BGEQ    160$                        ; If GEQ then slot not in use - DENY
         0051   30  06BC  1540         BSBW    SCAN_CACHE                  ; Scan the cache for this LPD
                     06BF  1541 ;
                     06BF  1542 ;       If LBC, scan failed.
                     06BF  1543 ;       We already started with DRT, so we already have R9 -> ADJ.
                     06BF  1544 ;
       41 50     E9  06BF  1545         BLBC    R0,300$                     ; Continue in common path
                     06C2  1546 ;
                     06C2  1547 ;       If LBS, scan successful. We must use "main" ADJ, since the
                     06C2  1548 ;       "main" ADJ will always have the RUN bit turned off.
                     06C2  1549 ;
  57   20 A8    9A  06C2  1550 200$:   MOVZBL  LPD$B_PTH_INX(R8),R7        ; Pick up "main" ADJ index
59   2C B247    D0  06C6  1551         MOVL    @RCB$L_PTR_ADJ(R2)[R7],R9   ; Get ADJ address
           36   11  06CB  1552         BRB     300$                        ; Continue in common path
                     06CD  1553
```

```
                                                     I 4
                     06CD  1554 SOL_AREA:                                   ; Solicit request for Level 2 Router
          00   E0    06CD  1555          BBS     #RCB$V_LVL2,-             ; If we are not allowed to do
 03 0B A2            06CF  1556                  RCB$B_STATUS(R2),220$      ; Level 2 routing,
       FFA2   31     06D2  1557          BRW     SOL_PH4                   ; Then act like a Level 1 router
                     06D5  1558 220$:    ;
                     06D5  1559          ;   First we MUST find the output ADJ based on the node address given
                     06D5  1560          ;   the destination node address supplied in R4.
                     06D5  1561          ;
                     06D5  1562          ;   Determine the output LPD from the output adjacency.
                     06D5  1563          ;
          51   95    06D5  1564          TSTB    R1                        ; Is this for area 0?
          B3   13    06D7  1565          BEQL    140$                      ; Br if yes - our "logical" area
008B C2   51   91    06D9  1566          CMPB    R1,RCB$B_HOMEAREA(R2)     ; Are we in the same area?
          AC   13    06DE  1567          BEQL    140$                      ; Br if yes - same as Level 1 Router
008C C2   51   91    06E0  1568          CMPB    R1,RCB$B_MAX_AREA(R2)     ; Is the destination area in range?
          24   1A    06E5  1569          BGTRU   NOT_REACH                 ; Br if not - node unreachable
57   20 B241   3C    06E7  1570          MOVZWL  @RCB$L_PTR_AOA(R2)[R1],R7 ; Get the next area ADJ index
          1D   13    06EC  1571          BEQL    NOT_REACH                 ; Br if not known - node unreachable
                     06EE  1572
                     06EE  1573 240$:    ;
                     06EE  1574          ; At this point:
                     06EE  1575          ;
                     06EE  1576          ;               R7 = Adj index
                     06EE  1577          ;               R3 = LPD index or zero
                     06EE  1578          ;
59   2C B247   D0    06EE  1579          MOVL    @RCB$L_PTR_ADJ(R2)[R7],R9 ; Get ADJ address
                     06F3  1580 260$:    ;
                     06F3  1581          ; At this point:
                     06F3  1582          ;
                     06F3  1583          ;               R9 = Adj address
                     06F3  1584          ;               R7 = Adj index
                     06F3  1585          ;               R3 = LPD index or zero
                     06F3  1586          ;
     58   53   9A    06F3  1587          MOVZBL  R3,R8                     ; Get path index, 0 => select it via ADJ
          04   12    06F6  1588          BNEQ    280$                      ; If NEQ then use it
  58   02 A9   9A    06F8  1589          MOVZBL  ADJ$B_LPD_INX(R9),R8      ; Use LPD index for this adjacency
58   28 B248   D0    06FC  1590 280$:    MOVL    @RCB$L_PTR_LPD(R2)[R8],R8 ; Get LPD address
          08   18    0701  1591          BGEQ    NOT_REACH                 ; If GEQ then slot not in use
                     0703  1592          ASSUME  LPD$V_ACTIVE  EQ  0       ;
    04 22 A8   E9    0703  1593 300$:    BLBC    LPD$W_STS(R8),NOT_REACH   ; If LBC, circuit is inactive
     50   01   D0    0707  1594          MOVL    #1,R0                     ; Success
               05    070A  1595          RSB                               ; Return with success
                     070B  1596
                     070B  1597 NOT_REACH:
          58   7C    070B  1598          CLRQ    R8                        ; Clear ADJ and LPD address
          50   D4    070D  1599          CLRL    R0                        ; No path available to node
               05    070F  1600          RSB
                     0710  1601
                     0710  1602          .DSABL  LSB
                     0710  1603
                     0710  1604 ;
                     0710  1605 ;        Scan the on-NI cache for this LPD. Return success/failure in R0.
                     0710  1606 ;        Inputs: R4 = node address to look for, R8 = addr of LPD.
                     0710  1607 ;
                     0710  1608 SCAN_CACHE:
 50   66 A8   D0    0710  1609          MOVL    LPD$L_CACHE(R8),R0        ; Get the CACHE table for this LPD
          0E   13    0714  1610          BEQL    20$                       ; Br if none
```

```
                                                                J 4
51    FA A0    3C  0716  1611          MOVZWL  -6(R0),R1              ; Get number of entries in CACHE
                   071A  1612          ;
                   071A  1613          ;       Scan CACHE
                   071A  1614          ;
     54   80    B1  071A  1615  10$:    CMPW    (R0)+,R4              ; Node address in cache?
          08    13  071D  1616          BEQL    30$                   ; Br if found
          80    B5  071F  1617          TSTW    (R0)+                 ; Skip timer cell
        F6 51   F5  0721  1618          SOBGTR  R1,10$                ; Keep looking
          50    D4  0724  1619  20$:    CLRL    R0                    ; Failure: node not in cache.
                05  0726  1620          RSB
     50   01    D0  0727  1621  30$:    MOVL    #1,R0                 ; Success: found node in cache.
                05  072A  1622          RSB
                   072B  1623
```

```
                     072B  1625          .SBTTL  TR$RCV_DIO_DATA - Rcv Direct I/O from datalink layer
                     072B  1626
                     072B  1627 ;+
                     072B  1628 ;  TR$RCV_DIO_DATA - Receive Direct I/O from datalink layer
                     072B  1629 ;
                     072B  1630 ;
                     072B  1631 ;  The IRP is being returned by the data link driver after a receive operation.
                     072B  1632 ;  Statistics are taken and the packet is routed to its destination.
                     072B  1633 ;
                     072B  1634 ;  The action is to remove the buffer from the IRP and to requeue the IRP to
                     072B  1635 ;  the same device for another receive.  The route-header in the message is
                     072B  1636 ;  parsed to determine the circuit over which the message is to be forwarded.
                     072B  1637 ;  A transmit IRP is allocated in order to shuttle the buffer to the device.
                     072B  1638 ;
                     072B  1639 ;
                     072B  1640 ;  INPUTS:       R5      "Internal" IRP address
                     072B  1641 ;                 R4-R0   Scratch
                     072B  1642 ;
                     072B  1643 ;                 IPL     IPL$_IOPOST  or  NET$C_IPL
                     072B  1644 ;
                     072B  1645 ;  OUTPUTS:      R5-R0   Garbage
                     072B  1646 ;
                     072B  1647 ;                 IPL     Same as entry
                     072B  1648 ;
                     072B  1649 ;-
                     072B  1650 TR$RCV_DIO_DATA::                            ; Rcv Direct I/O data from datalink
                     072B  1651          DSBINT  #NET$C_IPL                  ; Raise IPL
          07C0 8F    BB  0731  1652          PUSHR   #^M<R6,R7,R8,R9,R10>        ; Save regs
                     0735  1653                                              ;
    52    14 A5      D0  0735  1654          MOVL    IRP$L_ASTPRM(R5),R2         ; Get RCB
    58    10 A5      9A  0739  1655          MOVZBL  IRP$L_AST(R5),R8            ; Get index of IRP's LPD
 58    28 B248       D0  073D  1656          MOVL    @RCB$[_PTR_LPD(R2)[R8],R8 ; Get LPD address
    51    7E A2      3C  0742  1657          MOVZWL  RCB$W_TOTBUFSIZ(R2),R1      ; Get total buffer size assuming
                     0746  1658                                              ; 6 byte route header
          51    16   C0  0746  1659          ADDL    #TR$C_MAXHDR-6,R1           ; Adjust to account for largest
                     0749  1660                                              ; possible route header (NI)
          51    02   A0  0749  1661          ADDW    #2,R1                       ; Add 2 bytes for CRC16 just in case
                     074C  1662                                              ; this is an X.25 DLM datalink
          004C 8F    A3  074C  1663          SUBW3   #CXB$C_OVERHEAD,-           ; Reset byte count
    32 A5    51          0750  1664                  R1,IRP$W_BCNT(R5)           ;
                     0753  1665 ;
                     0753  1666 ;
                     0753  1667 ;  Detach the CXB from the IRP.  Setup the BUFFAIL flag in CXB$B_R_FLG
                     0753  1668 ;  according to whether or not there is a spare CXB in the free queue.
                     0753  1669 ;
                     0753  1670 ;
    56    24 A5      D0  0753  1671          MOVL    IRP$L_IOSB(R5),R6           ; Get buffer (CXB) address
          24 A5      D4  0757  1672          CLRL    IRP$L_IOSB(R5)              ; Erase former CXB pointer
          38 A6      94  075A  1673          CLRB    CXB$B_R_FLG(R6)             ; Init CXB flags
       00A0 C2       D1  075D  1674 30$:     CMPL    RCB$Q_CXB_FREE(R2),-        ; Any CXB's on free queue ?
       00A0 D2           0761  1675                  @RCB$Q_CXB_FREE(R2)         ;
             17      12  0764  1676          BNEQ    10$                         ; If NEQ then yes
          0A0E       30  0766  1677          BSBW    TR$ALLOCATE                 ; Else allocate one
          51    52   D0  0769  1678          MOVL    R2,R1                       ; Copy buffer address
    52    14 A5      D0  076C  1679          MOVL    IRP$L_ASTPRM(R5),R2         ; Recover RCB address
          07 50      E9  0770  1680          BLBC    R0,40$                      ; If LBC then allocation failure
00A0 D2    61        0E  0773  1681          INSQUE  (R1),@RCB$Q_CXB_FREE(R2); Insert it on the queue
```

```
                03   11  0778  1682                BRB     100$                    ; Continue
             38 A6   96  077A  1683 40$:           INCB    CXB$B_R_FLG(R6)         ; Set BUFFAIL status in CXB
                         077D  1684 100$:          ;
                         077D  1685                ;
                         077D  1686                ;   Process the message and then requeue the Rcv IRP.  Upon return
                         077D  1687                ;   from RCV_DIO_BIO, only the following register contents are valid:
                         077D  1688                ;
                         077D  1689                ;
                         077D  1690                ;       R6  =  CXB pointer (0 if no CXB)
                         077D  1691                ;       R5  =  IRP pointer (0 if IRP has disappeared -- in which case
                         077D  1692                ;                           the CXB has been deallocated as well)
                         077D  1693                ;       R2  =  RCB address
                         077D  1694                ;
                         077D  1695                ;
              00A2   30  077D  1696                BSBW    RCV_DIO_BIO             ; Goto common code
             53   55   D0  0780  1697                MOVL    R5,R3                   ; Copy IRP address
                  42   13  0783  1698                BEQL    200$                    ; If EQL none
          24 A3   56   D0  0785  1699                MOVL    R6,IRP$L_IOSB(R3)       ; Store CXB address
                  13   12  0789  1700                BNEQ    150$                    ; If NEQ then CXB was still there
          52   14 A3   D0  078E  1701                MOVL    IRP$L_ASTPRM(R3),R2     ; Get RCB address
       56   00A0 D2   0F  078F  1702                REMQUE  @RCB$Q_CXB_FREE(R2),R6  ; Get the CXB stored there
                  04   1C  0794  1703                BVC     140$                    ; If VC then got one
                       0796  1704                BUG_CHECK  NETNOSTATE,FATAL     ; CXB should have been there
          24 A3   56   D0  079A  1705 140$:          MOVL    R6,IRP$L_IOSB(R3)       ; Store CXB address
          66   48 A6   9E  079E  1706 150$:          MOVAB   CXB$C_HEADER(R6),(R6)   ; Setup message address (used for
                       07A2  1707                                                  ; common processing with buffered I/O)
                       07A2  1708                ;
                       07A2  1709                ;
                       07A2  1710                ;   Finish setting up IRP and requeue it to the datalink
                       07A2  1711                ;
                       07A2  1712                ;
                54   66  D0  07A2  1713                MOVL    (R6),R4                 ; Get msg address
          56   00000000'GF  D0  07A5  1714                MOVL    G^MMG$GL_SPTBASE,R6     ; Get system page table base
                     09  EF  07AC  1715                EXTZV   S^#VA$V_VPN,-           ; Get Virtual page frame number
                51   54   15  07AE  1716                        S^#VA$S_VPN,R4,R1       ;
                2C A3   6641  DE  07B1  1717                MOVAL   (R6)[R1],-              ; Enter SVAPTE
                              07B6  1718                        IRP$L_SVAPTE(R3)        ;
          30 A3   54   FE00 8F  AB  07B6  1719                BICW3   #^C<VA$M_BYTE>,R4,-     ; Enter page offset of msg
                              07BD  1720                        IRP$W_BOFF(R3)          ;
                55   1C A3   D0  07BD  1721                MOVL    IRP$L_UCB(R3),R5        ; Get UCB address
                00000000'GF   16  07C1  1722                JSB     G^EXE$ALTQUEPKT         ; Requeue the receive
                              07C7  1723 200$:          ;
                              07C7  1724                ;
                              07C7  1725                ;   Done.  The IRP has been requeued.  Return empty-handed to the EXEC
                              07C7  1726                ;
                              07C7  1727                ;
                    07C0 8F   BA  07C7  1728                POPR    #^M<R6,R7,R8,R9,R10>    ; Restore regs
                              07CB  1729                ENBINT                          ; Restore IPL
                              05  07CE  1730                RSB                             ; Return to Exec
                              07CF  1731
```

```
                     07CF  1733          .SBTTL  TR$RCV_BIO_DATA - Rcv Buffered I/O from datalink layer
                     07CF  1734
                     07CF  1735 ;+
                     07CF  1736 ;  TR$RCV_BIO_DATA - Receive Buffered I/O from datalink layer
                     07CF  1737 ;
                     07CF  1738 ;
                     07CF  1739 ;  The IRP is being returned by the data link driver after a receive operation.
                     07CF  1740 ;  Statistics are taken and the packet is routed to its destination.
                     07CF  1741 ;
                     07CF  1742 ;  The action is to remove the buffer from the IRP and to requeue the IRP to
                     07CF  1743 ;  the same device for another receive.  The route-header in the message is
                     07CF  1744 ;  parsed to determine the circuit over which the message is to be forwarded.
                     07CF  1745 ;  A transmit IRP is allocated in order to shuttle the buffer to the device.
                     07CF  1746 ;
                     07CF  1747 ;
                     07CF  1748 ;  INPUTS:       R5      "Internal" IRP address
                     07CF  1749 ;                R4-R0   Scratch
                     07CF  1750 ;
                     07CF  1751 ;                IPL     IPL$_IOPOST  or  NET$C_IPL
                     07CF  1752 ;
                     07CF  1753 ;  OUTPUTS:      R5-R0   Garbage
                     07CF  1754 ;
                     07CF  1755 ;                IPL     Same as entry
                     07CF  1756 ;
                     07CF  1757 ;-
                     07CF  1758 TR$RCV_BIO_DATA::                        ; Rcv Buffered I/O data from datalink
                     07CF  1759          DSBINT  #NET$C_IPL              ; Raise IPL
                07C0 8F   BB 07D5  1760          PUSHR   #^M<R6,R7,R8,R9,R10>    ; Save regs
                     07D9  1761                                          ;
           58     10 A5   9A 07D9  1762          MOVZBL  IRP$L_AST(R5),R8        ; Get address of IRP's LPD
           52     14 A5   D0 07DD  1763          MOVL    IRP$L_ASTPRM(R5),R2     ; Get RCB address
        58   28 B248   D0 07E1  1764          MOVL    @RCB$[_PTR_LPD(R2)[R8],R8 ; Get LPD address
           56     2C A5   D0 07E6  1765          MOVL    IRP$L_SVAPTE(R5),R6     ; Get buffer (CXB) address
                  0B   13 07EA  1766          BEQL    20$                     ;
                38 A6   94 07EC  1767          CLRB    CXB$B_R_FLG(R6)         ; Assume CXB is available
                  0C   E1 07EF  1768          BBC     #XM$V_STS_BUFFAIL,-     ; If BS then DLL receive has
             03 3C A5      07F1  1769                  IRP$L_IOST2(R5),20$     ; run out of receive buffers
                38 A6   96 07F4  1770          INCB    CXB$B_R_FLG(R6)         ; Mark CXB as unavailable
                     07F7  1771 20$:        ;
                     07F7  1772             ;
                     07F7  1773             ;   Process the message and then requeue the Rcv IRP.  Upon return
                     07F7  1774             ;   from RCV_DIO_BIO, only the following register contents are valid:
                     07F7  1775             ;
                     07F7  1776             ;
                     07F7  1777             ;       R6  =  CXB pointer (0 if no CXB)
                     07F7  1778             ;       R5  =  IRP pointer (0 if IRP has disappeared -- in which case
                     07F7  1779             ;                           the CXB has been deallocated as well)
                     07F7  1780             ;       R2  =  RCB address
                     07F7  1781             ;
                     07F7  1782             ;
                0028   30 07F7  1783          BSBW    RCV_DIO_BIO             ; Goto common code
             53     55   D0 07FA  1784          MOVL    R5,R3                   ; Copy IRP address
                  1B   13 07FD  1785          BEQL    200$                    ; If EQL none
          2C A3     56   D0 07FF  1786          MOVL    R6,IRP$L_SVAPTE(R3)     ; Send CXB back with IRP (0 if no CXB)
       32 A3   3FFF 8F   B0 0803  1787          MOVW    #^X<3FFF>,IRP$W_BCNT(R3); Reset Byte count
             55     1C A3   D0 0809  1788          MOVL    IRP$L_UCB(R3),R5        ; Get UCB address
                  05   12 080D  1789          BNEQ    70$                     ; If NEQ then "real" datalink
```

```
                08A1  30  080F  1790          BSBW    TR$LOC_DLL_RCV          ; Else, "Local LPD"
                  06  11  0812  1791          BRB     200$                    ; Continue
00000000'GF       16  0814  1792 70$:         JSB     G^EXE$ALTQUEPKT         ; Requeue the receive
                      081A  1793 200$:        ;
                      081A  1794              ;
                      081A  1795              ;   Done.  The IRP has been requeued.  Return empty-handed to the EXEC
                      081A  1796              ;
                      081A  1797              ;
              07C0 8F   BA  081A  1798          POPR    #^M<R6,R7,R8,R9,R10>    ; Restore regs
                      081E  1799          ENBINT                              ; Restore IPL
                      05  0821  1800          RSB                             ; Return to Exec
                      0822  1801
```

```
                     0822  1803          .SBTTL  RCV_DIO_BIO     - Common Receive IRP processing
                     0822  1804
                     0822  1805 ;+
                     0822  1806 ;  RCV_DIO_BIO  - Common Receive IRP processing
                     0822  1807 ;
                     0822  1808 ;  Finish processing of the received buffer. Determine size of message
                     0822  1809 ;  and check for success of the read request.
                     0822  1810 ;
                     0822  1811 ;  INPUTS:       R10,R9  Scratch
                     0822  1812 ;                 R8      LPD ptr
                     0822  1813 ;                 R6      Message buffer pointer
                     0822  1814 ;                 R5      "Internal" IRP address
                     0822  1815 ;                 R3-R4   Scratch
                     0822  1816 ;                 R2      RCB ptr
                     0822  1817 ;                 R0-R1   Scratch
                     0822  1818 ;
                     0822  1819 ;  OUTPUTS:      R8,R7   Garbage
                     0822  1820 ;                 R6      Address of buffer to deallocate
                     0822  1821 ;                         0 if no buffer is to be deallocated
                     0822  1822 ;                 R5-R0   Garbage
                     0822  1823 ;
                     0822  1824 ;-
                     0822  1825 RCV_DIO_BIO:                                ; Common buffered/direct receive code
          00'  B0    0822  1826          MOVW    S^#IO$_READLBLK,-          ; Reset I/O function code
        20 A5        0824  1827          IRP$W_FUNC(R5)                     ;
        2C A5   D4   0826  1828          CLRL    IRP$L_SVAPTE(R5)           ; Indicate no buffer attached
     32 38 A5   E9   0829  1829          BLBC    IRP$L_IOST1(R5),50$        ; Br if I/O was unsuccessful
                     082D  1830          ;
                     082D  1831          ;
                     082D  1832          ;   Process the received message
                     082D  1833          ;
                     082D  1834          ;
  57    3A A5    3C   082D  1835          MOVZWL  IRP$L_IOST1+2(R5),R7       ; Get transfer size
           2B    13   0831  1836          BEQL    40$                        ; If EQL, no message
                     0833  1837
        19 38 A6    E9   0833  1838          BLBC    CXB$B_R_FLG(R6),20$        ; If BC then datalink doesn't need the
                     0837  1839                                             ; buffer back (i.e., no BUFFAIL)
           1C A5    D5   0837  1840          TSTL    IRP$L_UCB(R5)              ; Is there a UCB ?
              14    13   083A  1841          BEQL    20$                        ; If EQL no, the "Local LPD"
                     083C  1842          INCPMS  RCVBUFFL                   ; Update the PMS counter
              0C    10   0842  1843          BSBB    20$                        ; Dispatch on message type
              56    D5   0844  1844          TSTL    R6                         ; Was buffer consumed?
              16    12   0846  1845          BNEQ    40$                        ; If not, then return IRP/CXB to caller
  32 A8       55    D0   0848  1846          MOVL    R5,LPD$L_RCV_IRP(R8)       ; Save the IRP address  -- its
                     084C  1847                                             ; presence also serves as a flag
              55    D4   084C  1848          CLRL    R5                         ; Don't requeue this IRP to datalink
              0E    11   084E  1849          BRB     40$                        ; Exit
                     0850  1850          ;
                     0850  1851          ;   Normal case.  Datalink is not starved for receive buffers.
                     0850  1852          ;
                     0850  1853
           44 A5    B0   0850  1854 20$:     MOVW    IRP$Q_STATION+4(R5),-      ; Get source node address
           36 A6         0853  1855                  CXB$W_R_SRCNOD(R6)         ;  save it in the CXB
              55    DD   0855  1856          PUSHL   R5                         ; Save IRP address
              55    D4   0857  1857          CLRL    R5                         ; Make sure DISP doesn't use IRP
              0E    10   0859  1858          BSBB    DISP_RCV_MSG               ; Dispatch rcv'd message
              55  8ED0   085B  1859          POPL    R5                         ; Recover IRP address, fix stack
```

```
                     085E  1860
                05   085E  1861 40$:    RSB                                     ;
                     085F  1862
                     085F  1863
                     085F  1864 50$:    ;
                     085F  1865         ;
                     085F  1866         ;   The Datalink has gone inactive.  Requeue the IRP to the ACP to
                     085F  1867         ;   inform it of this event and dellocate the I/O buffer.
                     085F  1868         ;
                     085F  1869         ;
          24 A5  56  D0    085F  1870         MOVL    R6,IRP$L_IOSB(R5)       ; Setup CXB address for deallocation
               06F8  30    0863  1871         BSBW    TR_RTRN_IRP             ; LPD is shutting down, return IRP
                 56  D4    0866  1872         CLRL    R6                      ; Indicate the CXB was consumed
                     05    0868  1873         RSB                             ; Done
                           0869  1874
```

```
                                                D 5
                                0869  1876          .SBTTL  DISP_RCV_MSG    Dispatch rcv'd message
                                0869  1877
                                0869  1878 ;+
                                0869  1879 ;  DISP_RCV_MSG - Dispatch rcv'd message
                                0869  1880 ;
                                0869  1881 ;  Process the received message by dispatching to the appropriate action
                                0869  1882 ;  routine.  The most frequent case is a message with a Phase III route-header.
                                0869  1883 ;
                                0869  1884 ;  All ECL message type codes are currently constrained to have their low two
                                0869  1885 ;  bits clear so that they may be distinguished from Transport message headers.
                                0869  1886 ;  The first byte of the received message should be one of the following:
                                0869  1887 ;
                                0869  1888 ;
                                0869  1889 ;       <0000 1000>     Phase II NOP
                                0869  1890 ;       <0101 1000>     Phase II Start
                                0869  1891 ;
                                0869  1892 ;       <0100 xx10>     Phase II route header
                                0869  1893 ;       <000x x010>     Phase III route header
                                0869  1894 ;       <000x x010>     Phase IV non-broadcast circuit route header
                                0869  1895 ;       <00xx 0x10>     Phase IV broadcast circuit route header
                                0869  1896 ;
                                0869  1897 ;       <0000 0001>     Phase III init
                                0869  1898 ;       <0000 0011>     Phase III verification
                                0869  1899 ;       <0000 0101>     Phase III hello message
                                0869  1900 ;       <0000 0111>     Phase III routing message
                                0869  1901 ;       <0000 1001>     Phase IV Level 2 routing message
                                0869  1902 ;       <0000 1011>     Phase IV broadcast circuit Router Hello message
                                0869  1903 ;       <0000 1101>     Phase IV broadcast circuit Endnode Hello message
                                0869  1904 ;
                                0869  1905 ;  All ECL message type codes are currently contrained to have their low
                                0869  1906 ;  two bit clear so that they may be distinguished from Transport message
                                0869  1907 ;  headers.
                                0869  1908 ;
                                0869  1909 ;
                                0869  1910 ;  INPUTS:       R10,R9  Scratch
                                0869  1911 ;                R8      LPD ptr
                                0869  1912 ;                R7      Total bytes in message
                                0869  1913 ;                R6      Message buffer pointer
                                0869  1914 ;                R3-R5   Scratch
                                0869  1915 ;                R2      RCB ptr
                                0869  1916 ;                R0-R1   Scratch
                                0869  1917 ;
                                0869  1918 ;  OUTPUTS:      R8,R7   Garbage
                                0869  1919 ;                R6      Address of buffer to deallocate
                                0869  1920 ;                        0 if no buffer is to be deallocated
                                0869  1921 ;                R5-R0   Garbage
                                0869  1922 ;
                                0869  1923 ;-
                                0869  1924 DISP_RCV_MSG:                                ; Dispatch rcv'd message
0A A6    1B    90               0869  1925             MOVB    #DYN$C_CXB,CXB$B_TYPE(R6) ; Store standard buffer type
   51    66    D0               086D  1926             MOVL    (R6),R1                   ; Get msg address
      20 A8    B0               0870  1927             MOVW    LPD$W_PTH(R8),-           ; Setup receive path i.d.
      32 A6                     0873  1928                     CXB$W_R_PATH(R6)          ;
      0E A2    B0               0875  1929             MOVW    RCB$W_ADDR(R2),-          ; Setup default destination node
      34 A6                     0878  1930                     CXB$W_R_DSTNOD(R6)        ; (assume non-route-thru)
      20 A8    9B               087A  1931             MOVZBW  LPD$B_PTH_INX(R8),-       ; Store LPD index as ADJ index
      3A A6                     087D  1932                     CXB$W_R_ADJ(R6)           ; (in case we need to send to ACP)
```

```
                                 087F  1933          ;
                                 087F  1934          ; For the X.25 circuits we will calculate the CRC16 on the
                                 087F  1935          ; data portion of the message and check to make sure the data
                                 087F  1936          ; is valid.
                                 087F  1937          ;
          1E 22 A8    07    E1  087F  1938          BBC     #LPD$V_X25,LPD$W_STS(R8),9$ ; Br if not X.25 path
                57    02    C2  0884  1939          SUBL    #2,R7                       ; Remove CRC from size
                      13    15  0887  1940          BLEQ    3$                          ; If received size = 0-2, report error
                   7E   51    7D  0889  1941          MOVQ    R1,-(SP)                    ; Save registers
    02 A1   57    00    F770  CF  088C  1942          CRC     CRC16,#0,R7,2(R1)           ; Calculate CRC16 on data
                   51   8E    7D  0894  1943          MOVQ    (SP)+,R1                    ; Restore registers
                   81   50    B1  0897  1944          CMPW    R0,(R1)+                    ; Does the CRC match?
                        03    13  089A  1945          BEQL    5$                          ; Br if okay
                      0218    31  089C  1946 3$:      BRW     PFE                         ; Else, treat as Format Error
                                 089F  1947
                   66   51    D0  089F  1948 5$:      MOVL    R1,(R6)                     ; Reset message pointer
                                 08A2  1949
                                 08A2  1950 9$:      ;
                                 08A2  1951          ;   Strip off leading pad bytes
                                 08A2  1952          ;
                0E 61   07    E5  08A2  1953          BBCC    #7,(R1),10$                 ; Br if not padded
                   50   61    9A  08A6  1954          MOVZBL  (R1),R0                     ; Pick up pad length
                   51   50    C0  08A9  1955          ADDL    R0,R1                       ; Point to first byte of message
                   66   51    D0  08AC  1956          MOVL    R1,(R6)                     ; Reset message pointer
                   57   50    C2  08AF  1957          SUBL    R0,R7                       ; Adjust message length
                        E8    15  08B2  1958          BLEQ    3$                          ; Br if bad message
                                 08B4  1959 10$:     ;
                                 08B4  1960          ;   Find the adjacency using the source address of the message.
                                 08B4  1961          ;
                      0A    E1  08B4  1962          BBC     #LPD$V_BC,-                 ; Br if NOT a Broadcast circuit
                7A 22 A8          08B6  1963                   LPD$W_STS(R8),40$          ;
                                 08B9  1964          ;
                                 08B9  1965          ;       Get address of the "Designated OA", DRT.
                                 08B9  1966          ;
                      05    91  08B9  1967          CMPB    #ADJ$C_PTY_PH4N,-           ; Are we an Endnode?
                   1D A8          08BB  1968                  LPD$B_ETY(R8)               ; ..checked on LPD
                      0B    12  08BD  1969          BNEQ    15$                         ; Br if NOT
             54    2C A8    3C  08BF  1970          MOVZWL  LPD$W_DRT(R8),R4            ; Get designated output adjacency index
          59    2C B244    D0  08C3  1971          MOVL    @RCB$L_PTR_ADJ(R2)[R4],R9   ; Get ADJ address
                   7B    11  08C8  1972          BRB     60$                         ; Continue in common code
                                 08CA  1973 15$:     ;
                                 08CA  1974          ;       For Broadcast Circuit, we will first try the OA vector
                                 08CA  1975          ;       to look for a match in the ADJ database. If we find a
                                 08CA  1976          ;       match then we've got the ADJ, else we will assume this
                                 08CA  1977          ;       message came from a Broadcast Router and scan the BRA
                                 08CA  1978          ;       portion of the ADJ vector.
                                 08CA  1979          ;
                      0A    EF  08CA  1980          EXTZV   #TR4$V_ADDR_AREA,-          ; Get the area number
       53    36 A6    06        08CC  1981                  #TR4$S_ADDR_AREA,CXB$W_R_SRCNOD(R6),R3
                      07    13  08D0  1982          BEQL    18$                         ; If area = 0, assume our area
          008B C2     53    91  08D2  1983          CMPB    R3,RCB$B_HOMEAREA(R2)       ; Our area?
                      29    12  08D7  1984          BNEQ    23$                         ; If not, then skip OA optimization
                      00    EF  08D9  1985 18$:     EXTZV   #TR4$V_ADDR_DEST,-          ; Get node number within area
       53    36 A6    0A        08DB  1986                  #TR4$S_ADDR_DEST,CXB$W_R_SRCNOD(R6),R3
             5A A2     53    B1  08DF  1987          CMPW    R3,RCB$W_MAX_ADDR(R2)       ; Is address in range?
                      03    1B  08E3  1988          BLEQU   20$                         ; Br if yes
                   020F    31  08E5  1989          BRW     RANGE                       ; Else, address out of range
```

```
                                0902  2000 ;       Now try scan of entire ADJ database for the node
                                0902  2001 ;
   53   36 A6   3C   0902  2002 23$:    MOVZWL  CXB$W_R_SRCNOD(R6),R3   ; Get full source node address
   54   5C A2   9A   0906  2003         MOVZBL  RCB$B_MAX_LPD(R2),R4    ; Get number of LPD's in system
   55   68 A2   3C   090A  2004         MOVZWL  RCB$W_MAX_ADJ(R2),R5    ; Get number of routing "destinations"
          14    12   090E  2005         BNEQ    30$                     ; Start at BRA's, if any
          16    11   0910  2006         BRB     35$                     ; Else, skip it
59   2C B244   D0   0912  2007 25$:    MOVL    @RCB$L_PTR_ADJ(R2)[R4],R9 ; Get next ADJ
  04 A9   53   B1   0917  2008         CMPW    R3,ADJ$W_PHA(R9)        ; Does the node address match?
          07    12   091B  2009         BNEQ    30$                     ; Br if no - skip to next ADJ
       02 A9   91   091D  2010         CMPB    ADJ$B_LPD_INX(R9),-     ; Is this the right LPD?
       20 A8        0920  2011                 LPD$B_PTH_INX(R8)       ;
          21    13   0922  2012         BEQL    60$                     ; Br if yes
  EA 54   55   F3   0924  2013 30$:    AOBLEQ  R5,R4,25$               ; Loop if more BRA ADJ's
                                0928  2014 ;
                                0928  2015 ;     If scan fails, then use the "main" ADJ (where RUN flag is off)
                                0928  2016 ;
   54   20 A8   9A   0928  2017 35$:    MOVZBL  LPD$B_PTH_INX(R8),R4    ; Use the "main" ADJ
59   2C B244   D0   092C  2018         MOVL    @RCB$L_PTR_ADJ(R2)[R4],R9 ; Get ADJ address
          12    11   0931  2019         BRB     60$                     ; Skip reset of listener timer
                                0933  2020
                                0933  2021 ;
                                0933  2022 ;  For non-Broadcast Circuit, use ADJ index in the LPD
                                0933  2023 ;
 54   20 A8   9A   0933  2024 40$:    MOVZBL  LPD$B_PTH_INX(R8),R4    ; Get the ADJ index (same as LPD index)
59   2C B244   D0   0937  2025         MOVL    @RCB$L_PTR_ADJ(R2)[R4],R9 ; Get the ADJ address
       01    E1   093C  2026         BBC     #ADJ$V_RUN,-            ; If ADJ isn't up,
     05 69        093E  2027                 ADJ$B_STS(R9),60$       ; skip reset of listener timer
     08 A9   B0   0940  2028         MOVW    ADJ$W_INT_LSN(R9),-     ; Reset "listen" interval
     0A A9        0943  2029                 ADJ$W_TIM_LSN(R9)       ;
                                0945  2030
                                0945  2031 ;
                                0945  2032 ;   Save the source ADJ index in the CXB
                                0945  2033 ;
3A A6   54   B0   0945  2034 60$:    MOVW    R4,CXB$W_R_ADJ(R6)      ; Save the source adjacency index
                                0949  2035 ;
                                0949  2036 ;   Journal the received message.
                                0949  2037 ;
                                0949  2038         .IF     DF,JNX$$$
                                0949  2039
   50   01   8E   0949  2040         MNEGB   #1,R0                   ; Set journal type = Received msg
     083F   30   094C  2041         BSBW    TR_FILL_JNX             ; Store journal record
                                094F  2042
                                094F  2043         .ENDC
                                094F  2044
                                094F  2045 ;
                                094F  2046 ;   Parse the message and dispatch.
```

```
                        094F  2047          ;
                        094F  2048          ;   On input, R9 always points to an ADJ block.  If ADJ$V_RUN=0,
                        094F  2049          ;   then the message was received on an ADJ which hasn't yet been
                        094F  2050          ;   initialized.
                        094F  2051          ;
                        094F  2052
    55   81   9A       094F  2053          MOVZBL  (R1)+,R5                ; Get message type flag
                        0952  2054          ASSUME  TR3$V_MSG_CTL  EQ  0
      2B 55   E8        0952  2055          BLBS    R5,80$                  ; If LBS then control msg
0D 55    01   E1        0955  2056          BBC     #TR3$V_MSG_RTH,R5,75$   ; If BC then NOT a route header
     01 A9   91        0959  2057          CMPB    ADJ$B_PTYPE(R9),-       ; If Phase II connection,
         02             095C  2058                  #ADJ$C_PTY_PH2          ; then skip VER check (since VER is
         04   13        095D  2059          BEQL    74$                     ; the same bit as RTFLG_PH2)
                        095F  2060                                          ; Else, for non-PH2 circuits,
2F 55    06   E0        095F  2061          BBS     #TR4$V_RTFLG_VER,R5,90$ ; If version bit set, ignore msg
       0053   31        0963  2062 74$:     BRW     TR_RTHDR                ; Else, must be a route header
                        0966  2063          ;
                        0966  2064          ;   The message doesn't have a router header.  Assume Phase II
                        0966  2065          ;
    08   55   91        0966  2066 75$:     CMPB    R5,#TR3$C_MSG_NOP2      ; NOP message ?
         27   13        0969  2067          BEQL    90$                     ; If EQL yes, ignore it
58 8F    55   91        096B  2068          CMPB    R5,#TR3$C_MSG_STR2      ; Is it a Start message ?
         1E   13        096F  2069          BEQL    85$                     ; EQL => UNKNOWN MESSAGE
                        0971  2070          ;
                        0971  2071          ;   It's a Phase II data message.  Since the message didn't
                        0971  2072          ;   have any route header, we must store the source node from
                        0971  2073          ;   the adjacency for this circuit.
                        0971  2074          ;
         01   E1        0971  2075          BBC     #ADJ$V_RUN,-            ; If the ADJ is not known,
   05 69              0973  2076                  ADJ$B_STS(R9),77$       ; then leave node address  = 0
   04 A9   B0         0975  2077          MOVW    ADJ$W_PNA(R9),-         ; Save source node address
   36 A6              0978  2078                  CXB$W_R_SRCNOD(R6)
51    66   D0         097A  2079 77$:     MOVL    (R6),R1                 ; Point to first msg byte
     0109   31        097D  2080          BRW     TR_ECL                  ; Pass to ECL layer
                        0980  2081
                        0980  2082          ;
                        0980  2083          ;   NOTE - All offsets to the "Hello" message are off by 1 byte -
                        0980  2084          ;          from MOVZBL (R1)+... above.
                        0980  2085          ;
                        0980  2086 80$:                                     ; Transport layer control msg
55    05   91        0980  2087          CMPB    #TR3$C_MSG_HELLO,R5     ; "Hello" msg ?
         0D   13        0983  2088          BEQL    90$                     ; If EQL yes, ignore it
55    0D   91        0985  2089          CMPB    #TR4$C_MSG_BCEHEL,R5    ; Phase IV BC Endnode "Hello" msg?
         09   13        0988  2090          BEQL    100$                    ; Br if yes
55    0B   91        098A  2091          CMPB    #TR4$C_MSG_BCRHEL,R5    ; Phase IV BC router "Hello" msg?
         21   13        098D  2092          BEQL    ADJ_UP                  ; Br if yes
     0175   31        098F  2093 85$:     BRW     UNK                     ; Else message type unknown
            05        0992  2094 90$:     RSB                             ; Done
                        0993  2095
                        0993  2096          ;
                        0993  2097          ;   Process a broadcast endnode "Hello" msg, reset "listener" timer.
                        0993  2098          ;
         01   E1        0993  2099 100$:    BBC     #ADJ$V_RUN,-            ; If the ADJ is not known,
   19 69              0995  2100                  ADJ$B_STS(R9),ADJ_UP    ; report "new adjacency" to NETACP
1E    57   D1         0997  2101          CMPL    R7,#30                  ; Is message big enough?
         1A   15        099A  2102          BLEQ    PFE_BR                  ; Br if not, error
   01 A9   91         099C  2103          CMPB    ADJ$B_PTYPE(R9),-       ; Has the node type changed?
```

```
                    05        099F  2104                  #ADJ$C_PTY_PH4N
                    0E     12 09A0  2105          BNEQ    ADJ_UP                          ; Br if yes, adjacency up
          06 A9  0A A1     B1 09A2  2106          CMPW    10(R1),ADJ$W_BUFSIZ(R9)         ; Is BLKSIZ still okay?
                    07     12 09A7  2107          BNEQ    ADJ_UP                          ; Br if not, adjacency up
                 08 A9     B0 09A9  2108          MOVW    ADJ$W_INT_LSN(R9),-             ; Else, Reset "listen" timer
                 0A A9        09AC  2109                  ADJ$W_TIM_LSN(R9)
                    E2     11 09AE  2110          BRB     90$                             ; And ignore the msg
                              09B0  2111 ADJ_UP:                                         ; Adjacency UP event
                              09B0  2112          ;
                              09B0  2113          ;   Adjacency up processing, if we receive a Router's Broadcast
                              09B0  2114          ;   Hello message, then let the NETACP reset the "listener" timer.
                              09B0  2115          ;
                 50    0C     90 09B0  2116          MOVB    #NETMSG$C_ADJ,R0                ; Set up event code
                     0154     31 09B3  2117          BRW     TO_ACP                          ; Pass it to the ACP
                              09B6  2118
                     00FE     31 09B6  2119 PFE_BR:  BRW     PFE                             ; Packet format error
```

```
                                                     I  5
                09B9  2121          .SBTTL  TR_RTHDR        - Process rcv'd msg's route header
                09B9  2122
                09B9  2123 ;+
                09B9  2124 ;
                09B9  2125 ;  TR_RTHDR     - Process received message's route header
                09B9  2126 ;
                09B9  2127 ;
                09B9  2128 ;  INPUTS:      R10     Scratch
                09B9  2129 ;               R9      ADJ address (RUN flag may be "off")
                09B9  2130 ;               R8      LPD address
                09B9  2131 ;               R7      Message size
                09B9  2132 ;               R6      CXB address
                09B9  2133 ;               R5      Contents of first byte in message
                09B9  2134 ;               R4,R3   Scratch
                09B9  2135 ;               R2      RCB address
                09B9  2136 ;               R1      Ptr to second byte in message
                09B9  2137 ;               R0      Scratch
                09B9  2138 ;
                09B9  2139 ;  OUTPUTS:     R6      0 if CXB was consumed, else preserved
                09B9  2140 ;
                09B9  2141 ;
                09B9  2142 ;-
                09B9  2143 TR_RTHDR:                                    ; Process rcv'd msg's route-header
    23 55   06  E1  09B9  2144         BBC     #TR3$V_RTFLG_PH2,R5,20$ ; If BC then Phase III route-header
                09BD  2145         ;
                09BD  2146         ;
                09BD  2147         ;   Process Phase II header
                09BD  2148         ;
                09BD  2149         ;
    05 69   01  E1  09BD  2150         BBC     #ADJ$V_RUN,ADJ$B_STS(R9),10$ ; Br if "main" ADJ
        04 A9   B0  09C1  2151         MOVW    ADJ$W_PNA(R9),-         ; Save source node address
           36 A6    09C4  2152                 CXB$W_R_SRCNOD(R6)
        50  81  9A  09C6  2153 10$:    MOVZBL  (R1)+,R0                ; Get size of dest. node name
        51  50  C0  09C9  2154         ADDL    R0,R1                   ; Advance to src node name
        57  50  A2  09CC  2155         SUBW    R0,R7                   ; Subtract from total
        50  81  9A  09CF  2156         MOVZBL  (R1)+,R0                ; Get size of src node name
        57  50  A2  09D2  2157         SUBW    R0,R7                   ; Subtract from total
        51  50  C0  09D5  2158         ADDL    R0,R1                   ; Advance pointer
        57  03  A2  09D8  2159         SUBW    #3,R7                   ; Account for count field and
                09DB  2160                                             ; msg type bytes
            D9  15  09DB  2161         BLEQ    PFE_BR                  ; If LEQ, report Packet Format Error
          00A1  31  09DD  2162         BRW     100$                    ; Else, continue in common
                09E0  2163 20$:    ;
                09E0  2164         ;
                09E0  2165         ;   Process Phase III or Phase IV route header
                09E0  2166         ;
                09E0  2167         ;
            02  E0  09E0  2168         BBS     #TR4$V_RTFLG_LNG,-      ; Is this a Phase IV long packet?
        4A 55       09E2  2169                 R5,50$                  ; If so, parse as such
                09E4  2170         ;
                09E4  2171         ;   Process only Phase III and Phase IV non-broadcast route hdr
                09E4  2172         ;
        57  06  A2  09E4  2173         SUBW    #6,R7                   ; Account for message header
            CD  15  09E7  2174         BLEQ    PFE_BR                  ; Br if packet format error
        50  81  3C  09E9  2175         MOVZWL  (R1)+,R0                ; Get destination node address
        54  81  3C  09EC  2176         MOVZWL  (R1)+,R4                ; Get the source node address
                09EF  2177         ASSUME  ADJ$C_PTY_PH3 EQ 0
```

```
                                                J 5
                     09EF  2178          ASSUME  ADJ$C_PTY_PH3N EQ 1
          01 A9  91  09EF  2179          CMPB    ADJ$B_PTYPE(R9),-       ; Is this a Phase III node's msg?
             01      09F2  2180                  #ADJ$C_PTY_PH3N         ;
             1D  1A  09F3  2181          BGTRU   30$                     ; Br if not
       008B  C2  F0  09F5  2182          INSV    RCB$B_HOMEAREA(R2),-    ; Else, fill in the Area of the dst
             0A      09F9  2183                  #TR4$V_ADDR_AREA,-      ;  node address with our "homearea"
          50 06      09FA  2184                  #TR4$S_ADDR_AREA,R0     ;
    FC    A1 50  B0  09FC  2185          MOVW    R0,-4(R1)               ; Reset the dst node address in msg
             0A  ED  0A00  2186          CMPZV   #TR4$V_ADDR_AREA,-      ; Is the source "area"
 00       54 06      0A02  2187                  #TR4$S_ADDR_AREA,R4,#0  ;   zero?
             0B  12  0A05  2188          BNEQ    30$                     ; Br if no - leave it alone
       008B  C2  F0  0A07  2189          INSV    RCB$B_HOMEAREA(R2),-    ; Else, fill in the Area of the source
             0A      0A0B  2190                  #TR4$V_ADDR_AREA,-      ;  node address with our "homearea"
          54 06      0A0C  2191                  #TR4$S_ADDR_AREA,R4     ;
    FE    A1 54  B0  0A0E  2192          MOVW    R4,-2(R1)               ; Stuff it back into the message
    36    A6 54  B0  0A12  2193  30$:    MOVW    R4,CXB$W_R_SRCNOD(R6)   ; Save the source node address
 50       0E A2  B1  0A16  2194          CMPW    RCB$W_ADDR(R2),R0       ; Is this for the local node?
             63  13  0A1A  2195          BEQL    80$                     ; If EQL then its for ECL
 50    008D C2  B1  0A1C  2196          CMPW    RCB$W_ALIAS(R2),R0      ; Is this for our alias?
             5C  13  0A21  2197          BEQL    80$                     ; If EQL then its for ECL
             50  B5  0A23  2198          TSTW    R0                      ; We boot with address 0
                 0A25  2199                                              ;& is this extra check really needed?
             58  13  0A25  2200          BEQL    80$                     ; If EQL then its for ECL
    54    01 A1  9E  0A27  2201          MOVAB   1(R1),R4                ; Point to start of data
          00FA  31  0A2B  2202  40$:     BRW     TR_RTHRU                ; Else, its a route-thru packet
                 0A2E  2203
                 0A2E  2204          ;
                 0A2E  2205          ;   Process a Phase IV Broadcast Circuit header (long format)
                 0A2E  2206          ;
          57 15  A2  0A2E  2207  50$:    SUBW    #21,R7                  ; Account for message header
             83  15  0A31  2208          BLEQ    PFE_BR                  ; If LEQ, report Packet Format Error
          51 06  C0  0A33  2209          ADDL    #6,R1                   ; Skip S-AREA and S-SUBAREA and HIORD
          50 81  3C  0A36  2210          MOVZWL  (R1)+,R0                ; Get Destination address
          51 06  C0  0A39  2211          ADDL    #6,R1                   ; Skip S-AREA and S-SUBAREA and HIORD
                 0A3C  2212          ;
                 0A3C  2213          ;  If this is an Endnode circuit, then update the endnode cache
                 0A3C  2214          ;
             05  91  0A3C  2215          CMPB    #ADJ$C_PTY_PH4N,-       ; Are we an endnode?
          1D A8      0A3E  2216                  LPD$B_ETY(R8)           ; ..on this LPD (only PH4 can have BCs)
             0C  12  0A40  2217          BNEQ    55$                     ; Br if not
             0A  E1  0A42  2218          BBC     #LPD$V_BC,-             ; Br if NOT a Broadcast Circuit
       07 22 A8      0A44  2219                  LPD$W_STS(R8),55$       ;& ..this check may be redundant!
    03 55    04  E0  0A47  2220          BBS     #TR4$V_RTFLG_RTS,R5,55$ ; Br if this is an RTS packet,
                 0A4B  2221                                              ; then the source address is invalid
          03FF  30  0A4B  2222          BSBW    UPDATE_CACHE            ; Else, update the cache entry
    36 A6    81  B0  0A4E  2223  55$:    MOVW    (R1)+,CXB$W_R_SRCNOD(R6); Enter the source node address
             51  D6  0A52  2224          INCL    R1                      ; Skip over NEXT LEVEL 2 ROUTER
    50    0E A2  B1  0A54  2225          CMPW    RCB$W_ADDR(R2),R0       ; Is this for the local node?
             23  13  0A58  2226          BEQL    60$                     ; Br if yes - okay
 50    008D C2  B1  0A5A  2227          CMPW    RCB$W_ALIAS(R2),R0      ; Is this for the local alias?
             1C  13  0A5F  2228          BEQL    60$                     ; Br if yes - okay
    54    03 A1  9E  0A61  2229          MOVAB   3(R1),R4                ; Assume route thru message, preset
                 0A65  2230                                              ; R4 to point past the header
             50  B5  0A65  2231          TSTW    R0                      ; We boot with address 0
                 0A67  2232                                              ;& is this extra check really needed?
             C2  12  0A67  2233          BNEQ    40$                     ; Br if no - route the packet thru
 F9 A1  000400AA 8F  D1  0A69  2234          CMPL    #TR4$C_HIORD,-7(R1)     ; Does source HIORD match?
```

```
                    44   12  0A71  2235          BNEQ    PFE                             ; Br if not - format error
          F1 A1 000400AA 8F   D1  0A73  2236          CMPL    #TR4$C_HIORD,-15(R1)            ; Does destination HIORD match?
                    3A   12  0A7B  2237          BNEQ    PFE                             ; Br if not - format error
                    81   B5  0A7D  2238 60$:      TSTW    (R1)+                           ; Skip VISIT and S-CLASS
                    51   D6  0A7F  2239 80$:      INCL    R1                              ; Skip Protocol Type
                         0A81  2240 100$:     ASSUME  TR4$V_RTFLG_RTS EQ TR3$V_RTFLG_RTS
                 04 55   04   E1  0A81  2241          BBC     #TR3$V_RTFLG_RTS,R5,110$; Br if not return-to-sender packet
                 38 A6   02   88  0A85  2242          BISB    #2,CXB$B_R_FLG(R6)              ; Else, indicate a RTS packet
                         0A89  2243 110$:                                                 ; Fall thru to TR_ECL
                         0A89  2244
```

```
                                                L 5
                                .SBTTL  TR_ECL          - Pass Rcv'd Packet to ECL

                                ;+
                                ;  TR_ECL       - Pass Packet to End Communications Layer
                                ;
                                ;  INPUTS:      R10,R9  Scratch
                                ;               R8      LPD address associated with receiving datalink
                                ;               R7      Size of ECL message
                                ;               R6      Received CXB address
                                ;               R5-R3   Scratch
                                ;               R2      RCB address
                                ;               R1      Points to first byte in ECL message
                                ;               R0      Scratch
                                ;
                                ;               CXB$W_R_SRCNOD  Source node address
                                ;               CXB$W_R_DSTNOD  Destination node (the ECL) address
                                ;               CXB$B_R_FLG     Low bit clear if CXB can be consumed
                                ;                               Low bit set if CXB must be returned
                                ;
                                ;  OUTPUTS:     R8,R7   Garbage
                                ;               R6      0 if CXB was consumed
                                ;                       Else, CXB address
                                ;               R5-R0   Garbage
                                ;
                                ;-
                                TR_ECL:                                         ; Pass rcv'd packet to ECL
                                        BUMP    L,LPD$L_CNT_APR(R8)             ; Update 'arriving pkts rcvd'
                                        INCPMS  ARRLOCPR                        ; ... and the PMS database too
                                        MOVW    R7,CXB$W_R_BCNT(R6)             ; Setup ECL message size
                                        MOVL    R2,CXB$L_R_RCB(R6)              ; Setup RCB pointer
                                                                                ;& perhaps CXB...RCB is not needed
                                        MOVL    R1,CXB$L_R_MSG(R6)              ; Point to ECL message
                                        CMPZV   #TR4$V_ADDR_AREA,-              ; If the source area number = 0,
                                                #TR4$S_ADDR_AREA,-
                                                CXB$W_R_SRCNOD(R6),#0
                                        BNEQ    10$
                                        INSV    RCB$B_HOMEAREA(R2),-            ; then insert our home area
                                                #TR4$V_ADDR_AREA,-              ; to ensure that NSP matches node
                                                #TR4$S_ADDR_AREA,-              ; numbers correctly
                                                CXB$W_R_SRCNOD(R6)
                                10$:    ;
                                ;
                                ;  Call the ECL layer with the following:
                                ;
                                ;       R8      Scratch
                                ;       R7      Size of ECL message
                                ;       R6      Received CXB address
                                ;       R5-R3   Scratch
                                ;       R2      RCB address
                                ;       R1      Points to first byte in ECL message
                                ;       R0      Scratch
                                ;
                                ;       CXB$L_R_RCB     RCB address             (copy of R2)
                                ;       CXB$L_R_MSG     Points to ECL message   (copy of R1)
                                ;       CXB$W_R_BCNT    Size of ECL message     (copy of R7)
```

```
                                                M 5
                     0AB4  2303          ;       CXB$W_R_SRCNOD  Source node address
                     0AB4  2304          ;       CXB$W_R_DSTNOD  Destination node (the ECL) address
                     0AB4  2305          ;       CXB$B_R_FLG     Low bit clear if CXB can be consumed
                     0AB4  2306          ;                       Low bit set if CXB must be returned
                     0AB4  2307          ;                       Second bit clear if no return-to-sender packet
                     0AB4  2308          ;                       Second bit set if packet returned-to-sender
                     0AB4  2309          ;       CXB$W_R_PATH    I.D. of receiving LPD
                     0AB4  2310          ;
                     0AB4  2311          ;
                     0AB4  2312          ;   On return here:
                     0AB4  2313          ;
                     0AB4  2314          ;       R8,R7   Garbage
                     0AB4  2315          ;       R6      0 if CXB was consumed.  Else, CXB address with the
                     0AB4  2316          ;               CXB$W_SIZE and CXB$B_TYPE fields unmodified.
                     0AB4  2317          ;       R5-R0   Garbage
                     0AB4  2318          ;
                     0AB4  2319          ;
F549'  31            0AB4  2320          BRW     NET$UNSOL_INTR          ; Pass message to ECL layer
                     0AB7  2321
                     0AB7  2322
```

```
                        0AB7  2324          .SBTTL  Packet Errors   - Process miscellaneous packet errors
                        0AB7  2325
                        0AB7  2326 ;+
                        0AB7  2327 ;
                        0AB7  2328 ;  The packet (CXB) could not be routed thru.  Update the appropriate
                        0AB7  2329 ;  statistics.  Pass the packet to the ACP to report the event if necessary.
                        0AB7  2330 ;
                        0AB7  2331 ;
                        0AB7  2332 ;  INPUTS:       R10     Scratch
                        0AB7  2333 ;                R9      ADJ address or zero
                        0AB7  2334 ;                R8      Applicable LPD address
                        0AB7  2335 ;                R7      Message size
                        0AB7  2336 ;                R6      CXB address
                        0AB7  2337 ;                R5      Scratch
                        0AB7  2338 ;                R2      RCB address
                        0AB7  2339 ;                R0      Scratch
                        0AB7  2340 ;
                        0AB7  2341 ;  OUTPUTS:      R6      0 if CXB was consumed
                        0AB7  2342 ;                        Else unchanged
                        0AB7  2343 ;                R5      Garbage
                        0AB7  2344 ;                R0      Garbage
                        0AB7  2345 ;
                        0AB7  2346 ;                All other registers are preserved
                        0AB7  2347 ;
                        0AB7  2348 ;-
                        0AB7  2349 PFE:     BUMP    B,RCB$B_CNT_PFE(R2)     ; Update packet format errors
50   08   90            0AC2  2350          MOVB    #NETMSG$C_PFE,R0        ; Setup event code
     43   11            0AC5  2351          BRB     TO_ACP                  ; Give it to the ACP
                        0AC7  2352                                          ;
                        0AC7  2353 OPL:     BUMP    B,RCB$B_CNT_OPL(R2)     ; Update oversized packet loss
50   0A   90            0AD2  2354          MOVB    #NETMSG$C_OPL,R0        ; Setup event code
     33   11            0AD5  2355          BRB     TO_ACP                  ; Pass the buffer to the ACP
                        0AD7  2356                                          ;
                        0AD7  2357 AGED:    BUMP    B,RCB$B_CNT_APL(R2)     ; Update aged packet loss
50   05   90            0AE2  2358          MOVB    #NETMSG$C_APL,R0        ; Setup event code
     23   11            0AE5  2359          BRB     TO_ACP                  ; Pass it to the ACP
                        0AE7  2360                                          ;
                        0AE7  2361 REACH:   BUMP    W,RCB$W_CNT_NUL(R2)     ; Update node unreachable loss
50   06   90            0AF2  2362          MOVB    #NETMSG$C_NUL,R0        ; Setup event code
     13   11            0AF5  2363          BRB     TO_ACP                  ; Pass it to the ACP
                        0AF7  2364                                          ;
                        0AF7  2365 RANGE:   BUMP    B,RCB$B_CNT_NOL(R2)     ; Update node address out of range loss
50   07   90            0B02  2366          MOVB    #NETMSG$C_NOL,R0        ; Setup event code
     03   11            0B05  2367          BRB     TO_ACP                  ; Pass it to the ACP
                        0B07  2368                                          ;
                        0B07  2369 UNK:                                     ; Unknown message type
50   01   90            0B07  2370          MOVB    #NETMSG$C_UNK,R0        ; Set up event code
                        0B0A  2371
                        0B0A  2372 ;
                        0B0A  2373 ;  Send an indication to NETACP that there is a problem on this datalink.
                        0B0A  2374 ;  This is done by transforming the CXB into what looks like NETACP's WQE
                        0B0A  2375 ;  block (assuming that the fields don't overlap), and queueing it to the
                        0B0A  2376 ;  AQB.  It is important that the block type remains DYN$C_CXB since
                        0B0A  2377 ;  NETACP dispatches on block type.
                        0B0A  2378 ;
                        0B0A  2379 ;
                        0B0A  2380 ;  INPUTS:       R0 = NETMSG$C_xxx code
```

```
                        0B0A  2381 ;                       R7 = Message size
                        0B0A  2382 ;                       R6 = CXB address
                        0B0A  2383 ;
                        0B0A  2384 ;
      16 A6    57   B0  0B0A  2385 TO_ACP: MOVW    R7,WQE$L_PM2+2(R6)      ; Setup size of msg
      55    3A A6   3C  0B0E  2386         MOVZWL  CXB$W_R_ADJ(R6),R5      ; Get ADJ index for source node
14 A6    66    56   A3  0B12  2387         SUBW3   R6,(R6),WQE$L_PM2(R6)   ; Setup offset to msg
      10 A6    50   90  0B17  2388         MOVB    R0,WQE$B_EVT(R6)        ; Setup event code
      20 A6    55   B0  0B1B  2389         MOVW    R5,WQE$W_ADJ_INX(R6)    ; Store ADJ index in WQE
         55    56   D0  0B1F  2390         MOVL    R6,R5                   ; Get buffer address
               56   D4  0B22  2391         CLRL    R6                      ; Flag it as gone
            0507    30  0B24  2392         BSBW    TR$QUE_WQE_AQB          ; Queue it to the AQB
                    05  0B27  2393         RSB
```

```
                                                C 6
                                0B28  2395          .SBTTL  TR_RTHRU        - Process packet for route-thru
                                0B28  2396
                                0B28  2397 ;+
                                0B28  2398 ;  TR_RTHRU     - Process packet for route-thru
                                0B28  2399 ;
                                0B28  2400 ;  INPUTS:      R10     Scratch
                                0B28  2401 ;               R9      ADJ address of receiving adjacency
                                0B28  2402 ;               R8      LPD address of receiving datalink
                                0B28  2403 ;               R7      message size (excluding header)
                                0B28  2404 ;               R6      CXB address
                                0B28  2405 ;               R5      Contents of first byte in message
                                0B28  2406 ;               R4      Ptr to message past the header
                                0B28  2407 ;               R3      Scratch
                                0B28  2408 ;               R2      RCB address
                                0B28  2409 ;               R1      Ptr to messages's VISIT field in route-header
                                0B28  2410 ;               R0      Destination node address
                                0B28  2411 ;
                                0B28  2412 ;  IMPLICIT INPUTS:
                                0B28  2413 ;
                                0B28  2414 ;               CXB$W_R_SRCNOD = Node address of source of message
                                0B28  2415 ;
                                0B28  2416 ;  OUTPUTS:     R8,R7   Garbage
                                0B28  2417 ;               R6      0 if CXB was consumed.
                                0B28  2418 ;                       Else CXB address
                                0B28  2419 ;               R5-R0   Garbage
                                0B28  2420 ;
                                0B28  2421 ;
                                0B28  2422 ;-
                                0B28  2423 TR_RTHRU:                                ; Process packet for route-thru
                                0B28  2424 ;
                                0B28  2425 ;
                                0B28  2426 ;  Route-thru packet
                                0B28  2427 ;
                                0B28  2428 ;
                          01    0B28  2429          BBC     #RCB$V_ACT,-            ; If ACP is not active, then return
                      2A 0B A2  0B2A  2430                  RCB$B_STATUS(R2),2$      ;   packet to sender
                                0B2D  2431          $DISPATCH LPD$B_ETY(R8),TYPE=B,- ; Return packet if we are an Endnode
                                0B2D  2432          <-
                                0B2D  2433                  <ADJ$C_PTY_PH3N 2$>,-   ; Phase III endnode
                                0B2D  2434                  <ADJ$C_PTY_PH4N 2$>,-   ; Phase IV endnode
                                0B2D  2435          >
                                0B3C  2436          BUMP    L,LPD$L_CNT_TPR(R8)     ; Bump 'transit packets rcvd'
                                0B45  2437          INCPMS  ARRTRAPR                ; ... and the PMS database too
                             58   DD  0B4B  2438          PUSHL   R8                      ; Save LPD that we received packet on
                           006F   30  0B4D  2439          BSBW    ROUTE                   ; Re-route the packet
                             58 8ED0  0B50  2440          POPL    R8                      ; Restore receiving LPD address
                             56   D5  0B53  2441          TSTL    R6                      ; Was packet consumed?
                             29   13  0B55  2442          BEQL    5$                      ; If EQL then yes
                                0B57  2443 2$:         ;
                                0B57  2444          ;  Return Packet to Sender
                                0B57  2445          ;
                                0B57  2446          ;  If the packet was not sent we must return the packet to
                                0B57  2447          ;  the sender, but only if the sender has requested it.
                                0B57  2448          ;
                                0B57  2449          ;  Swap the source and destination node addresses, repair the
                                0B57  2450          ;  VISITs field, reset R0 and R8, and route the packet to its source.
                                0B57  2451          ;
```

```
                                                D 6
                     0B57  2452          ;   The request return to sender is different for Phase IV Broadcast
                     0B57  2453          ;   packet headers - so we will parse that separately.
                     0B57  2454          ;
  26 55    02    E0  0B57  2455          BBS     #TR4$V_RTFLG_LNG,R5,10$ ; Br if Phase IV long format
  21 55    03    E5  0B5B  2456          BBCC    #TR3$V_RTFLG_RQR,R5,5$ ; Br if return not requested
  1D 55    04    E2  0B5F  2457          BBSS    #TR3$V_RTFLG_RTS,R5,5$ ; If BS then already being returned
     51    66    D0  0B63  2458          MOVL    (R6),R1                 ; Get message address
     81    55    90  0B66  2459          MOVB    R5,(R1)+                ; Reset control flags
     53    50    B0  0B69  2460          MOVW    R0,R3                   ; Save output node address
  50    36 A6    3C  0B6C  2461          MOVZWL  CXB$W_R_SRCNOD(R6),R0   ; Get node address of orignal src node
  36 A6    53    B0  0B70  2462          MOVW    R3,CXB$W_R_SRCNOD(R6)   ; Set new src node address
81    61    10    9C  0B74  2463          ROTL    #16,(R1),(R1)+          ; Swap src, dst node addresses
     61    97    DE  0B78  2464          DECB    (R1)                    ; Repair VISITs field
54    51    01    C1  0B7A  2465          ADDL3   #1,R1,R4                ; R4 points to start of data
           3F    10  0B7E  2466          BSBB    ROUTE                   ; Route the packet to its sender
           05    0B80  2467 5$:          RSB                             ; Done
                 0B81  2468          ;
                 0B81  2469          ;   Phase IV long packet format - return to sender
                 0B81  2470          ;
  2D 55    03    E5  0B81  2471 10$:     BBCC    #TR4$V_RTFLG_RQR,R5,20$ ; Br if return not requested
  29 55    04    E2  0B85  2472          BBSS    #TR4$V_RTFLG_RTS,R5,20$ ; If BS, then already being returned
     51    66    D0  0B89  2473          MOVL    (R6),R1                 ; Get message address
     61    55    90  0B8C  2474          MOVB    R5,(R1)                 ; Reset control flags
     53    50    B0  0B8F  2475          MOVW    R0,R3                   ; Save output node address
  50    36 A6    3C  0B92  2476          MOVZWL  CXB$W_R_SRCNOD(R6),R0   ; Get node address of orignal src node
  36 A6    53    B0  0B96  2477          MOVW    R3,CXB$W_R_SRCNOD(R6)   ; Set new src node address
  7E    07 A1    B0  0B9A  2478          MOVW    7(R1),-(SP)             ; Save old destination node address
07 A1    0F A1    B0  0B9E  2479          MOVW    15(R1),7(R1)            ; Set new destination node address
  0F A1    8E    B0  0BA3  2480          MOVW    (SP)+,15(R1)            ; Set new source node address
     51    12    C0  0BA7  2481          ADDL    #18,R1                  ; Point to VISITs field of message
     61    97    DE  0BAA  2482          DECB    (R1)                    ; Repair VISITs field
54    51    03    C1  0BAC  2483          ADDL3   #3,R1,R4                ; R4 points to start of data
           0D    10  0BB0  2484          BSBB    ROUTE                   ; Route the packet to its sender
           05    0BB2  2485 20$:         RSB                             ; Done
                 0BB3  2486
                 0BB3  2487
                 0BB3  2488 CHECK_RQR:                                   ; Check if return requested
                 0BB3  2489          ASSUME  TR3$V_RTFLG_RQR EQ TR4$V_RTFLG_RQR
                 0BB3  2490          ASSUME  TR3$V_RTFLG_RTS EQ TR4$V_RTFLG_RTS
07 55    03    E1  0BB3  2491          BBC     #TR3$V_RTFLG_RQR,R5,20$ ; Br if return not requested
03 55    04    E0  0BB7  2492          BBS     #TR3$V_RTFLG_RTS,R5,20$ ; If BS then already being returned
  5E    04    C0  0BBB  2493          ADDL    #4,SP                   ; Return to callers caller
           05    0BBE  2494 20$:         RSB
                 0BBF  2495
                 0BBF  2496
03 38 A6    E9  0BBF  2497 ROUTE:   BLBC    CXB$B_R_FLG(R6),10$     ; If LBC, okay to forward packet
     009D    31  0BC3  2498          BRW     100$                    ; Else, can't take packet - error
                 0BC6  2499          ;
                 0BC6  2500          ;
                 0BC6  2501          ;   Process the VISIT field to prevent infinite packet looping
                 0BC6  2502          ;
                 0BC6  2503          ;
     61    96    0BC6  2504 10$:     INCB    (R1)                    ; Bump the VISIT field
61    5E A2    91  0BC8  2505          CMPB    RCB$B_MAX_VISIT(R2),(R1); Within VISIT range ?
     11    1A    0BCC  2506          BGTRU   20$                     ; If GTRU then no violation
                 0BCE  2507          ASSUME  TR3$V_RTFLG_RTS EQ TR4$V_RTFLG_RTS
0A 55    04    E1  0BCE  2508          BBC     #TR3$V_RTFLG_RTS,R5,15$ ; If BC then packet is not being
```

```
                        0BD2  2509                                    ;  returned to its original sender
7E   5E A2    02   85   0BD2  2510          MULB3   #2,RCB$B_MAX_VISIT(R2),-(SP) ; Else allow twice MAX_VISITS
          8E  61   91   0BD7  2511          CMPB    (R1),(SP)+              ;
              03   19   0BDA  2512          BLSS    20$                     ; If LSS then let it return to sender
                 0094  31   0BDC  2513 15$:   BRW     110$                    ; Else, report AGED Packet Loss
                       0BDF  2514
                       0BDF  2515 20$:   ;
                       0BDF  2516        ;
                       0BDF  2517        ;    Determine the output adjacency for the packet
                       0BDF  2518        ;
                       0BDF  2519        ;
                   0A  EF   0BDF  2520          EXTZV   #TR4$V_ADDR_AREA,-      ; Get the destination node "AREA"
      5A   50      06       0BE1  2521                  #TR4$S_ADDR_AREA,R0,R10 ;
                   00  EF   0BE4  2522          EXTZV   #TR4$V_ADDR_DEST,-      ; Get only the destination
                   0A       0BE6  2523                  #TR4$S_ADDR_DEST,-      ;  portion of the node address
               53  50       0BE7  2524                  R0,R3                   ;
                       0BE9  2525        ;
                       0BE9  2526        ;       We must find the output adjacency based on the type of node
                       0BE9  2527        ;       we are and what the destination node "area" is.
                       0BE9  2528        ;
                       0BE9  2529          $DISPATCH RCB$B_ETY(R2),TYPE=B,- ; Dispatch on our node type
                       0BE9  2530          <-
                       0BE9  2531                  <ADJ$C_PTY_AREA 30$>,-  ; Phase IV level 2 router
                       0BE9  2532                  <ADJ$C_PTY_PH4  40$>,-  ; Phase IV level 1 router
                       0BE9  2533          >
                   2C  11   0BF3  2534          BRB     50$                     ; All others
                       0BF5  2535        ;
                       0BF5  2536        ;   Phase IV Level 2 router.
                       0BF5  2537        ;
                   00  E1   0BF5  2538 30$:   BBC     #RCB$V_LVL2,-           ; If we are not allowed to do Level 2
        19 0B      A2       0BF7  2539                  RCB$B_STATUS(R2),40$    ; routing, then do Level 1 routing
008C    C2         5A   91  0BFA  2540          CMPB    R10,RCB$B_MAX_AREA(R2)  ; Area within range?
                   78   1A  0BFF  2541          BGTRU   120$                    ; Br if no
                   5A   95  0C01  2542          TSTB    R10                     ; Is this for our "logical" area 0?
                   1C   13  0C03  2543          BEQL    50$                     ; Br if yes
008B    C2         5A   91  0C05  2544          CMPB    R10,RCB$B_HOMEAREA(R2)  ; Is this in our area?
                   15   13  0C0A  2545          BEQL    50$                     ; Br if yes - just like Level 1 message
53      20 B24A    3C       0C0C  2546          MOVZWL  @RCB$L_PTR_AOA(R2)[R10],R3 ; Else, get "area" output adjacency
                   19   11  0C11  2547          BRB     60$                     ; Finish in common code
                       0C13  2548        ;
                       0C13  2549        ;   Phase IV level 1 router
                       0C13  2550        ;
008B    C2         5A   91  0C13  2551 40$:   CMPB    R10,RCB$B_HOMEAREA(R2)  ; Is this in our area?
                   07   13  0C18  2552          BEQL    50$                     ; Br if yes
53      00AC       C2   3C  0C1A  2553          MOVZWL  RCB$W_LVL2(R2),R3       ; Else, get our nearest level 2 router
                   0B   11  0C1F  2554          BRB     60$                     ; Finish in common code
                       0C21  2555        ;
                       0C21  2556        ;   All destinations for our area
                       0C21  2557        ;
     5A    A2      53   B1  0C21  2558 50$:   CMPW    R3,RCB$W_MAX_ADDR(R2)   ; Within range?
                   52   1A  0C25  2559          BGTRU   120$                    ; If GTRU then out of range
53      1C B243    3C       0C27  2560          MOVZWL  @RCB$L_PTR_OA(R2)[R3],R3; Get the output adjacency index
                       0C2C  2561                                          ; Don't clobber R9 yet in case EQL
                       0C2C  2562        ;
                       0C2C  2563        ;   Common proccessing
                       0C2C  2564        ;
                       0C2C  2565        ;       Inputs:
```

```
                        0C2C  2566          ;                       R3 = ADJ index
                        0C2C  2567          ;
          51    13      0C2C  2568 60$:     BEQL    130$                        ; If EQL then unreachable
59   2C B243    D0      0C2E  2569          MOVL    @RCB$L_PTR_ADJ(R2)[R3],R9   ; Get ADJ address
          01    E1      0C33  2570          BBC     #ADJ$V_RUN,-                ; Br if adjacency is not up
        48 69           0C35  2571           ADJ$B_STS(R9),130$                 ;
  53    02 A9    9A     0C37  2572          MOVZBL  ADJ$B_LPD_INX(R9),R3        ; Get LPD index for this adjacency
          42    13      0C3B  2573          BEQL    130$                        ; Br if no output path
     5A    58    D0      0C3D  2574          MOVL    R8,R10                      ; Save receiving LPD address
58   28 B243    D0      0C40  2575          MOVL    @RCB$L_PTR_LPD(R2)[R3],R8   ; Get LPD address
        1C A8    91      0C45  2576          CMPB     LPD$B_IRPCNT(R8),-         ; Does "square-root-limiter" allow it?
        1E A8            0C48  2577           LPD$B_XMT_SRL(R8)                  ;
          17    14      0C4A  2578          BGTR    100$                        ; If GTR then queue is full
  06 A9    57    B1      0C4C  2579          CMPW    R7,ADJ$W_BUFSIZ(R9)         ; Is the message too big for partner?
          33    1A      0C50  2580          BGTRU   140$                        ; If GTRU then oversized
          50    DD      0C52  2581          PUSHL   R0                          ; Save destination node address
  53    00 B2    0F      0C54  2582 80$:     REMQUE  @RCB$Q_IRP_FREE(R2),R3      ; Get an IRP
          42    1C      0C58  2583          BVC     200$                        ; If VC then got one
        04A9    30      0C5A  2584          BSBW    TR$ADJUST_IRP               ; Replenish the IRP queue
        F4 50   E8      0C5D  2585          BLBS    R0,80$                      ; If LBS then there's an IRP
          50 8ED0       0C60  2586          POPL    R0                          ; Restore destination node address
                        0C63  2587 100$:    BUMP    W,LPD$W_CNT_TCL(R8)         ; Update resource error packet loss
                        0C6C  2588          INCPMS  TRCNGLOS                    ; ... and the PMS database too
                05      0C72  2589          RSB
                        0C73  2590
        FF3D    30      0C73  2591 110$:    BSBW    CHECK_RQR                   ; Check if return requested
        FE5E    31      0C76  2592          BRW     AGED                        ; Packet VISITs field violation
        FF37    30      0C79  2593 120$:    BSBW    CHECK_RQR                   ; Check if return requested
        FE78    31      0C7C  2594          BRW     RANGE                       ; Destination address to large
        FF31    30      0C7F  2595 130$:    BSBW    CHECK_RQR                   ; Check if return requested
        FE62    31      0C82  2596          BRW     REACH                       ; Destination address unreachable
     58   5A    D0      0C85  2597 140$:    MOVL    R10,R8                      ; Reset address of receiving LPD
        FF28    30      0C88  2598          BSBW    CHECK_RQR                   ; Check if return requested
        FE39    31      0C8B  2599          BRW     OPL                         ; Packet too large to be forwarded
                        0C8E  2600
  55    00 B6    9A     0C8E  2601 150$:    MOVZBL  @(R6),R5                    ; Get the flags byte again
     62   63    0E      0C92  2602 160$:    INSQUE  (R3),RCB$Q_IRP_FREE(R2)     ; Put back the IRP
     58   5A    D0      0C95  2603          MOVL    R10,R8                      ; Reset address of receive LPD
        FF18    30      0C98  2604          BSBW    CHECK_RQR                   ; Return packet if requested
                05      0C9B  2605          RSB                                 ; Else, just drop it
                        0C9C  2606
          50 8ED0       0C9C  2607 200$:    POPL    R0                          ; Restore destination node address
                        0C9F  2608          ;
                        0C9F  2609          ;   We will prevent any route-thru traffic to Phase III nodes,
                        0C9F  2610          ;   if the source node is outside of our area. This is to prevent
                        0C9F  2611          ;   some implementions of DECnet from re-cycling the line on suspected
                        0C9F  2612          ;   packet format errors. There are no known implementations of Phase
                        0C9F  2613          ;   III DECnet that can handle the area field anyway.
                        0C9F  2614          ;
                        0C9F  2615          $DISPATCH ADJ$B_PTYPE(R9),TYPE=B,-
                        0C9F  2616          <-
                        0C9F  2617                  <ADJ$C_PTY_PH3N 210$>,- ; Phase III endnode
                        0C9F  2618                  <ADJ$C_PTY_PH3  210$>,- ; Phase III router
                        0C9F  2619          >
          11    11      0CA8  2620          BRB     220$                        ; Else, okay
          0A    EF      0CAA  2621 210$:    EXTZV   #TR4$V_ADDR_AREA,-          ; Get the source id "area" address
          06            0CAC  2622           #TR4$S_ADDR_AREA,-                 ;
```

```
                55    36 A6        0CAD  2623                    CXB$W_R_SRCNOD(R6),R5   ;
              008B C2     55   91   0CB0  2624          CMPB      R5,RCB$B_HOMEAREA(R2)   ; Is this our "homearea"?
                          D7   12   0CB5  2625          BNEQ      150$                    ; Br if not, not reachable to endnode
                55    00 B6   9A   0CB7  2626          MOVZBL    @(R6),R5                ; Pick up flags byte again
                                   0CBB  2627
                39 55     02   E1   0CBB  2628 220$:    BBC       #TR4$V_RTFLG_LNG,R5,270$ ; Br if NOT Phase IV long format header
              19 22 A8    0A   E0   0CBF  2629          BBS       #LPD$V_BC,LPD$W_STS(R8),230$ ; Br if output is a broadcast circuit
                                   0CC4  2630          ;
                                   0CC4  2631          ; We are converting the long format to short format, clear INI and
                                   0CC4  2632          ; long format flags, and check HIORD.
                                   0CC4  2633          ;
                   55   24   8A   0CC4  2634          BICB      #TR4$M_RTFLG_INI!-      ; Make sure the Intra-NI and
                                   0CC7  2635                      TR4$M_RTFLG_LNG,R5    ;  long format flags are clear
          F9 A1    000400AA 8F   D1   0CC7  2636          CMPL      #TR4$C_HIORD,-7(R1)     ; Does source HIORD match?
                            C1   12   0CCF  2637          BNEQ      160$                    ; Br if not, packet format error
          F1 A1    000400AA 8F   D1   0CD1  2638          CMPL      #TR4$C_HIORD,-15(R1)    ; Does destination HIORD match?
                            B7   12   0CD9  2639          BNEQ      160$                    ; Br if not, packet format error
                            0B   11   0CDB  2640          BRB       240$                    ; Continue
                                   0CDD  2641
                                   0CDD  2642 230$:    ;
                                   0CDD  2643          ;   Check to make sure the OUTPUT LPD = the INPUT LPD, if not
                                   0CDD  2644          ;   the same, then clear the Intra-Ethernet bit.  The Intra-NI
                                   0CDD  2645          ;   flag has already been set by the originating node if it
                                   0CDD  2646          ;   sent the packet over an Ethernet circuit, so all we have
                                   0CDD  2647          ;   to do in the route-thru case is make sure we clear the flag
                                   0CDD  2648          ;   when it leaves the Ethernet.
                                   0CDD  2649          ;
                      32 A6   B1   0CDD  2650          CMPW      CXB$W_R_PATH(R6),-      ; Is the output LPD = input LPD?
                      20 A8        0CE0  2651                    LPD$W_PTH(R8)
                            04   13   0CE2  2652          BEQL      240$                    ; Br if yes, okay
                                   0CE4  2653          CLRBIT    #TR4$V_RTFLG_INI,R5     ; Else, clear the Intra-Ethernet bit
                                   0CE8  2654 240$:    ;
                                   0CE8  2655          ;   Build a standard Phase III type route header from the
                                   0CE8  2656          ;   Phase IV long format header.
                                   0CE8  2657          ;
                   74    61   90   0CE8  2658          MOVB      (R1),-(R4)              ; Backbuild the header - visits field
                74    36 A6   B0   0CEB  2659          MOVW      CXB$W_R_SRCNOD(R6),-(R4); Store source node address
                   74    50   B0   0CEF  2660          MOVW      R0,-(R4)                ;   Store destination node address
                   74    55   90   0CF2  2661          MOVB      R5,-(R4)                ;   Store route msg flag byte
                   66    54   D0   0CF5  2662          MOVL      R4,(R6)                 ; Reset start of message ptr
                                   0CF8  2663          ;
                                   0CF8  2664          ;   Done building header, adjust message size and ship it.
                                   0CF8  2665          ;
                   51    66   D0   0CF8  2666 270$:    MOVL      (R6),R1                 ; Point to start of message
                   57    06   A0   0CFB  2667          ADDW      #6,R7                   ; Account for header
                     1C A8   96   0CFE  2668          INCB      LPD$B_IRPCNT(R8)        ; Account for IRP to be queued
                                   0D01  2669          ;
                                   0D01  2670          ;
                                   0D01  2671          ;   Build the IRP, attach the buffer, queue to communications driver
                                   0D01  2672          ;
                                   0D01  2673          ;
                                   0D01  2674          ASSUME    IRP$L_AST     EQ  4+IRP$L_PID
                                   0D01  2675          ASSUME    IRP$L_ASTPRM  EQ  4+IRP$L_AST
                                   0D01  2676
                50    0C A3   9E   0D01  2677          MOVAB     IRP$L_PID(R3),R0        ; Point to PID field
              80    0EE0'CF   9E   0D05  2678          MOVAB     W^TR$RTRN_XMT_RTH,(R0)+ ; Setup return address
                80    20 A8   3C   0D0A  2679          MOVZWL    LPD$W_PTH(R8),(R0)+     ; LPD i.d. into AST field
```

```
                80   52   D0  0D0E  2680          MOVL    R2,(R0)+                ; RCB address into ASTPRM
                              0D11  2681
                              0D11  2682          ASSUME  IRP$L_WIND EQ 4+IRP$L_ASTPRM
                              0D11  2683                                          ; Fall thru
                              0D11  2684
```

```
                        .SBTTL  FINISH_XMT_HDR  - Finish building HDR and transmit it

;+
; FINISH_XMT_HDR   - Finish building HDR and transmit it
;
; This routine will build a new Route Header based upon the output path.
;
;    The CXB is setup as follows:
;
;
;       +---------------+
;       |   standard    |       11 bytes long.  CXB$L_FLINK and CXB$L_BLINK may
;       |     VMS       |       be used by the Transport layer.  CXB$W_SIZE
;       | buffer header |       must be correct.  CXB$B_TYPE must be DYN$C_CXB.
;       +---------------+
;       |               |       Starts with CXB$B_CODE (byte 11) and continues
;       |      ECL      |       to CXB$C_LENGTH.  This area is read-only to
;       |   pure area   |       Transport and below.  It cannot even be
;       |               |       saved/restored.
;       +---------------+
;       |   Datalink    |       Starts at CXB$C_LENGTH and is at least
;       |     Layer     |       CXB$C_DLL bytes long.  Used by the datalink for
;       |  impure area  |       protocol header or state information.
;       +---------------+
;       |               |
;       .   body of     .       Must be quadword aligned and starting no sooner
;       .   message     .       than CXB$C_LENGTH + CXB$C_DLL (= CXB$C_HEADER)
;       .               .       The first 6 bytes contain: RTFLG,DSTNOD,SRCNOD
;       |               |         FORWARD, in that order.
;       +---------------+
;       |   Datalink    |       Used by the datalink layer for protocol (e.g.,
;       |     Layer     |       checksum) or state information.  Must be at
;       |  impure area  |       least CXB$C_TRAILER in length.
;       +---------------+
;
;
; INPUTS:       R10     Scratch
;               R9      ADJ address
;                       Zero if called by TALKER
;               R8      LPD address
;               R7      Total number of bytes in message
;               R6      Pointer to buffer containing message (CXB)
;               R5,R4   Scratch
;               R3      IRP address
;               R2      RCB address
;               R1      Pointer to start of message
;               R0      Address of IRP$L_WIND(R3)
;
; OUTPUTS:      R8      Preserved
;               R7      Garbage
;               R6      0
;               R5-R0   Garbage
;
;-
FINISH_XMT_HDR:                                 ; Finish building HDR and xmt it.
59   D5         TSTL    R9                      ; Did we have an ADJ?
```

```
                     24   13   0D13  2743          BEQL    5$                      ; If EQL then no - no header
                                0D15  2744          ;
                                0D15  2745          ;   We will make a special check here, to see if we are an
                                0D15  2746          ;   Endnode. This is because on a BC circuit the "main" ADJ has a
                                0D15  2747          ;   FTYPE of "unknown" which prevents the building of a route
                                0D15  2748          ;   header.
                                0D15  2749          ;
           08 22 A8   0A   E1   0D15  2750          BBC     #LPD$V_BC,LPD$W_STS(R8),3$ ; Br if NOT a broadcast-circuit
                                0D1A  2751          $DISPATCH RCB$B_ETY(R2),TYPE=B,-
                                0D1A  2752          <-
                                0D1A  2753                  <ADJ$C_PTY_PH4N, 10$>,- ; Phase IV endnode
                                0D1A  2754          >
                                0D22  2755  3$:     ;
                                0D22  2756          ; Build the appropriate header type - based on output adjacency
                                0D22  2757          ; node type.
                                0D22  2758          ;
                                0D22  2759          $DISPATCH ADJ$B_PTYPE(R9),TYPE=B,-
                                0D22  2760          <-
                                0D22  2761                  <ADJ$C_PTY_AREA 10$>,-  ; Phase IV level 2 router
                                0D22  2762                  <ADJ$C_PTY_PH4  10$>,-  ; Phase IV router
                                0D22  2763                  <ADJ$C_PTY_PH4N 10$>,-  ; Phase IV endnode
                                0D22  2764                  <ADJ$C_PTY_PH3  20$>,-  ; Phase III router
                                0D22  2765                  <ADJ$C_PTY_PH3N 20$>,-  ; Phase III endnode
                                0D22  2766          >
                                0D33  2767          ;
                                0D33  2768          ; All others including Phase II
                                0D33  2769          ;
                  57   06   C2  0D33  2770  4$:     SUBL    #TR3$C_HSZ_DATA,R7      ; Adjust msg size
                  51   06   C0  0D36  2771          ADDL    #TR3$C_HSZ_DATA,R1      ; Skip over Transport header
                       0075 31  0D39  2772  5$:     BRW     40$                     ; Join common code
                                0D3C  2773          ;
                                0D3C  2774          ;   Phase IV Router/Endnode
                                0D3C  2775          ;
                                0D3C  2776          ;       Build a new header if the output LPD is a broadcast-circuit
                                0D3C  2777          ;
           63 22 A8   0A   E1   0D3C  2778  10$:    BBC     #LPD$V_BC,LPD$W_STS(R8),30$ ; Br if NOT a broadcast circuit
                                0D41  2779          ;
                                0D41  2780          ;    Build a Phase IV broadcast circuit header
                                0D41  2781          ;
                                0D41  2782          ASSUME  TR4$V_RTFLG_RTS  EQ  TR3$V_RTFLG_RTS
                                0D41  2783          ASSUME  TR4$V_RTFLG_RQR  EQ  TR3$V_RTFLG_RQR
                        5A   61 90  0D41  2784          MOVB    (R1),R10                ; Get the flags byte
                                0D44  2785          ;
                                0D44  2786          ;   If the output LPD is a Broadcast Circuit Endnode, then
                                0D44  2787          ;   set the Intra-NI flag in the RTFLG byte of the message.
                                0D44  2788          ;   It will be cleared by routers if they route this packet
                                0D44  2789          ;   off the Ethernet.
                                0D44  2790          ;
                           05 A1 95  0D44  2791          TSTB    5(R1)                   ; Is this packet originating from here?
                              04 12  0D47  2792          BNEQ    12$                     ; If so,
                                0D49  2793          SETBIT  #TR4$V_RTFLG_INI,R10    ; Set the Intra-NI flag
                                0D4D  2794  12$:    SETBIT  #TR4$V_RTFLG_LNG,R10    ; Set the long format flag
                        7E   05 A1 90  0D51  2795          MOVB    5(R1),-(SP)             ; Get visits byte
                           52   61 90  0D55  2796          MOVB    (R1),R2                 ; Get route header flags byte
                        54   03 A1 B0  0D58  2797          MOVW    3(R1),R4                ; Get source node address
                        7E   01 A1 B0  0D5C  2798          MOVW    1(R1),-(SP)             ; Get destination address
                     55   51   0F C3  0D60  2799          SUBL3   #<TR4$C_HSZ_DATA-TR3$C_HSZ_DATA>,R1,R5 ; Point to header area
```

```
                51    55    D0   0D64  2800          MOVL    R5,R1                           ; Set new start of data
                57    0F    C0   0D67  2801          ADDL    #<TR4$C_HSZ_DATA-TR3$C_HSZ_DATA>,R7 ; Adjust msg size
                85    5A    90   0D6A  2802          MOVB    R10,(R5)+                       ; Enter transports message type
                      85    B4   0D6D  2803          CLRW    (R5)+                           ; RESERVED D-AREA, D-SUBAREA
          85    000400AA    8F   D0   0D6F  2804          MOVL    #TR4$C_HIORD,(R5)+              ; Store destination HIORD
                5A    8E    B0   0D76  2805          MOVW    (SP)+,R10                       ; Get destination node address
                85    5A    B0   0D79  2806          MOVW    R10,(R5)+                       ; Store destination address
                      85    B4   0D7C  2807          CLRW    (R5)+                           ; RESERVED S-AREA, D-SUBAREA
          85    000400AA    8F   D0   0D7E  2808          MOVL    #TR4$C_HIORD,(R5)+              ; Store source HIORD
                85    54    B0   0D85  2809          MOVW    R4,(R5)+                        ; Store source node address
                      85    D4   0D88  2810          CLRL    (R5)+                           ; Clear NL2, VISIT-CT, SERVICE CLASS
                                 0D8A  2811                                                  ;  and PROTOCOL TYPE
             FD A5    8E    90   0D8A  2812          MOVB    (SP)+,-3(R5)                    ; Store VISITs count
             44 A3    5A    B0   0D8E  2813          MOVW    R10,IRP$Q_STATION+4(R3)         ; Store destination node address in IRP
                01    E1   0D92  2814          BBC     #ADJ$V_RUN,-                    ; Br if adjacency is not up (ie this is
                   13 69          0D94  2815                   ADJ$B_STS(R9),35$           ;  the "main" ADJ)
                0C    11   0D96  2816          BRB     30$                             ; Join common code
                                 0D98  2817          ;
                                 0D98  2818          ;   Phase III header
                                 0D98  2819          ;
                                 0D98  2820          ;       For Phase III node, we must reset the "homearea" field of
                                 0D98  2821          ;       the destination id, and also for the source id.
                                 0D98  2822          ;
                                 0D98  2823 20$:     ;   Phase III endnodes
                                 0D98  2824          ;   Phase III routers
                                 0D98  2825          ;
                                 0D98  2826          ;       ***** The following is a requirement of the architecture *****
                                 0D98  2827          ;
                                 0D98  2828          ;       There are no known DECnet implementations which can handle
                                 0D98  2829          ;       node addresses from other areas. Therefore, for Phase III
                                 0D98  2830          ;       nodes we will always reset the area field of the source
                                 0D98  2831          ;       node address. There are checks in the route-thru code to
                                 0D98  2832          ;       prevent route through nodes from sending to Phase III nodes
                                 0D98  2833          ;       from other areas.
                                 0D98  2834          ;
          0A    00    F0   0D98  2835          INSV    #0,#TR4$V_ADDR_AREA,-           ; Reset "area" of source id
       03 A1    06         0D9B  2836                  #TR4$S_ADDR_AREA,3(R1)          ;
          0A    00    F0   0D9E  2837          INSV    #0,#TR4$V_ADDR_AREA,-           ; Reset "area" of destination id
       01 A1    06         0DA1  2838                  #TR4$S_ADDR_AREA,1(R1)          ;
             04 A9    B0   0DA4  2839 30$:     MOVW    ADJ$W_PNA(R9),-                 ; Set destination address
                44 A3         0DA7  2840                  IRP$Q_STATION+4(R3)         ;    in IRP
          000400AA 8F    D0   0DA9  2841 35$:     MOVL    #TR4$C_HIORD,-                  ;       ...
                40 A3         0DAF  2842                  IRP$Q_STATION(R3)           ;       ...
                              0DB1  2843          ;
                              0DB1  2844          ; Pad the message if required
                              0DB1  2845          ;
             55    51    D0   0DB1  2846 40$:     MOVL    R1,R5                           ; Copy start of message pointer
                   0D    E1   0DB4  2847          BBC     #LPD$V_ALIGNW,-                 ; Br if no word alignment needed
                03 22 A8         0DB6  2848                  LPD$W_STS(R8),47$           ;
              51    01    CA   0DB9  2849          BICL    #1,R1                           ; Else, backup message to word boundary
                   0E    E1   0DBC  2850 47$:     BBC     #LPD$V_ALIGNQ,-                 ; Br if no quadword alignment needed
                03 22 A8         0DBE  2851                  LPD$W_STS(R8),49$           ;
              51    07    CA   0DC1  2852          BICL    #7,R1                           ; Else, backup message to quadword
                              0DC4  2853                                                  ;  boundary
             55    51    C2   0DC4  2854 49$:     SUBL    R1,R5                           ; Calculate size of rounding
                   0A    13   0DC7  2855          BEQL    50$                             ; Branch if no pad required
             57    55    C0   0DC9  2856          ADDL    R5,R7                           ; Increase size of transfer
```

```
                ODCC  2857          SETBIT  #7,R5                   ; Set high bit to indicate pad count
61   55   90    ODD0  2858          MOVB    R5,(R1)                 ; Store pad "indicator"
                ODD3  2859 50$:
```

```
                                ;
                                ; Finish building the IRP and transmit it
                                ;
                                ;  INPUTS:      R10     Scratch
                                ;               R9      ADJ address or zero
                                ;               R8      LPD address
                                ;               R7      Total number of bytes in message
                                ;               R6      Pointer to buffer containing message (CXB)
                                ;               R5,R4   Scratch
                                ;               R3      IRP address
                                ;               R2      Scratch
                                ;               R1      Pointer to start of data
                                ;               R0      Address of IRP$L_WIND(R3)
                                ;
                                ;
                                ;   Journal the message to be transmitted
                                ;

                                .IF     DF,JNX$$$

                        50  DD  PUSHL   R0                          ; Save registers
                    52  14 A3  MOVL    IRP$L_ASTPRM(R3),R2       ; Get RCB address
                        50  D4  CLRL    R0                        ; Set journal type = Start transmit
                     03B0   30  BSBW    TR_FILL_JNX               ; Store journal record
                        50 8ED0 POPL    R0                        ; Restore registers

                                .ENDC

                                ;
                                ; For X.25 circuits we will have to calculate a CRC16 on the data
                                ; portion of the message.
                                ;
                 14 22 A8   07  E1  BBC     #LPD$V_X25,LPD$W_STS(R8),100$ ; Skip if not X.25 datalink
                            0B  BB  PUSHR   #^M<R0,R1,R3>             ; Save regs
            61  57  00    F214 CF  0B  CRC     CRC16,#0,R7,(R1)          ; Calculate CRC16 on data
                    52      50  D0  MOVL    R0,R2                     ; Save CRC
                            0B  BA  POPR    #^M<R0,R1,R3>             ; Restore regs
                    71      52  B0  MOVW    R2,-(R1)                  ; Save CRC in datagram
                    57      02  A0  ADDW    #2,R7                     ; Account

                    66      51  D0  100$:   MOVL    R1,(R6)                   ; Save address of start of data

                                ASSUME  IRP$L_WIND  EQ  4+IRP$L_ASTPRM
                                ASSUME  IRP$L_UCB   EQ  4+IRP$L_WIND
                                ASSUME  LPD$L_UCB   EQ  4+LPD$L_WIND

                80      0C A8   7D  MOVQ    LPD$L_WIND(R8),(R0)+      ; Fill WIND,UCB fields

                                ASSUME  IRP$W_FUNC  EQ  4+IRP$L_UCB
                                ASSUME  IRP$B_EFN   EQ  2+IRP$W_FUNC
                                ASSUME  IRP$B_PRI   EQ  1+IRP$B_EFN
                                ASSUME  IRP$L_IOSB  EQ  1+IRP$B_PRI

                    80  00'  D0  MOVL    S^#IO$_WRITELBLK,(R0)+    ; Fill FUNC, clear EFN and PRI
                FF  A0  1F   90  MOVB    #31,-1(R0)                ; Use lowest priority
                    80  56   D0  MOVL    R6,(R0)+                  ; Buffer address into IOSB
```

```
                        0E0B  2918          ASSUME  IRP$W_CHAN   EQ  4+IRP$L_IOSB
                        0E0B  2919          ASSUME  IRP$W_STS    EQ  2+IRP$W_CHAN
                        0E0B  2920          ASSUME  IRP$L_SVAPTE EQ  2+IRP$W_STS
                        0E0B  2921          ASSUME  IRP$W_BOFF   EQ  4+IRP$L_SVAPTE
                        0E0B  2922
          80  14 A8 AE  0E0B  2923          MNEGW   LPD$W_CHAN(R8),(R0)+    ; Enter CHAN
              05 E1     0E0F  2924          BBC     #LPD$V_XBF,-            ; If BC the xmitter I/O is direct
            0A 22 A8    0E11  2925                  LPD$W_STS(R8),120$      ;
                        0E14  2926          ;
                        0E14  2927          ;
                        0E14  2928          ;   Xmitter I/O is buffered
                        0E14  2929          ;
                        0E14  2930          ;
            80  01 B0   0E14  2931          MOVW    #IRP$M_BUFIO,(R0)+      ; Enter STS field
            80  56 D0   0E17  2932          MOVL    R6,(R0)+                ; Setup buffer ptr in SVAPTE
                80 B4   0E1A  2933          CLRW    (R0)+                   ; Clear BOFF
                1B 11   0E1C  2934          BRB     140$                    ; Continue
                        0E1E  2935 120$:    ;
                        0E1E  2936          ;
                        0E1E  2937          ;   Xmitter I/O is direct
                        0E1E  2938          ;
                        0E1E  2939          ;
                80 B4   0E1E  2940          CLRW    (R0)+                   ; Clear STS
              54  66 D0   0E20  2941          MOVL    (R6),R4                 ; Get msg address
      56  00000000'GF D0   0E23  2942          MOVL    G^MMG$GL_SPTBASE,R6     ; Get system page table base
                09 EF     0E2A  2943          EXTZV   S^#VA$V_VPN,-           ; Get Virtual page frame number
           51  54  15     0E2C  2944                  S^#VA$S_VPN,R4,R1       ;
              80 6641 DE  0E2F  2945          MOVAL   (R6)[R1],(R0)+          ; Enter SVAPTE
    80  54  FE00 8F AB     0E33  2946          BICW3   #^C<VA$M_BYTE>,R4,(R0)+ ; Enter page offset of msg in BOFF
                        0E39  2947 140$:    ;
                        0E39  2948          ;
                        0E39  2949          ;   Complete the IRP and queue it to the device
                        0E39  2950          ;
                        0E39  2951          ;
                        0E39  2952          ASSUME  IRP$W_BCNT   EQ  2+IRP$W_BOFF
                        0E39  2953          ASSUME  IRP$L_BCNT   EQ  0+IRP$W_BCNT
                        0E39  2954
        60  57  3C      0E39  2955          MOVZWL  R7,(R0)                 ; Enter BCNT
            56  D4      0E3C  2956          CLRL    R6                      ; Prevent buffer deallocation
    55  1C A3  D0      0E3E  2957          MOVL    IRP$L_UCB(R3),R5        ; Get comm driver UCB
            06  13      0E42  2958          BEQL    150$                    ; If EQL then this is Local LPD
00000000'GF     17      0E44  2959          JMP     G^EXE$ALTQUEPKT         ; Queue the packet to "real" datalink
          0254  31      0E4A  2960 150$:    BRW     TR$LOC_DLL_XMT          ; Queue the packet to "local" datalink
                        0E4D  2961
```

```
                                0E4D  2963          .SBTTL  UPDATE_CACHE    - Update the BC cache table
                                0E4D  2964
                                0E4D  2965 ;+
                                0E4D  2966 ;  UPDATE_CACHE - Update the BC cache table
                                0E4D  2967 ;
                                0E4D  2968 ;  INPUTS:       R10     Scratch
                                0E4D  2969 ;                R9      ADJ address
                                0E4D  2970 ;                R8      LPD address associated with receiving datalink
                                0E4D  2971 ;                R7      Size of ECL message
                                0E4D  2972 ;                R6      Received CXB address
                                0E4D  2973 ;                R5      Contents of first byte in message
                                0E4D  2974 ;                R4,R3   Scratch
                                0E4D  2975 ;                R2      RCB address
                                0E4D  2976 ;                R1      Ptr to source node address in message
                                0E4D  2977 ;                R0      Destination node address
                                0E4D  2978 ;
                                0E4D  2979 ;                CXB$W_R_SRCNOD  "Last Hop" node address
                                0E4D  2980 ;
                                0E4D  2981 ;
                                0E4D  2982 ;  OUTPUTS:
                                0E4D  2983 ;                R3,R4,R10       Garbage
                                0E4D  2984 ;                All other registers are preserved.
                                0E4D  2985 ;
                                0E4D  2986 ;
                                0E4D  2987 ;-
                                0E4D  2988 UPDATE_CACHE:                                ; Update the LPD's cache table
                                0E4D  2989                 ;
                                0E4D  2990                 ;
                                0E4D  2991                 ;       First we will check the source node address
                                0E4D  2992                 ;       against the PNA for the DRT. If they match, then
                                0E4D  2993                 ;       it must be the "Designated Router" (DRT) who sent the
                                0E4D  2994                 ;       message, since the "Main Adjacency" would have a node
                                0E4D  2995                 ;       address of -1. We will then set the ADJ to point to the
                                0E4D  2996                 ;       DRT, else we will scan the CACHE table for the received
                                0E4D  2997                 ;       LPD, treating this like a Non-BC circuit and use the ADJ
                                0E4D  2998                 ;       index of the LPD.
                                0E4D  2999                 ;
                                0E4D  3000                 ;   CACHE TABLE HANDLING:
                                0E4D  3001                 ;
                                0E4D  3002                 ;       If the DRT is not a real BRA, then we will scan the LPD
                                0E4D  3003                 ;       CACHE table to try and find the entry. If the entry was not
                                0E4D  3004                 ;       found then it will be inserted at the first available slot,
                                0E4D  3005                 ;       as long as the Intra-NI bit is set or the source of the packet
                                0E4D  3006                 ;       was the same as the last hop.
                                0E4D  3007                 ;
    53    61    3C              0E4D  3008                 MOVZWL  (R1),R3                 ; Get the source node address
04  A9    53    B1              0E50  3009                 CMPW    R3,ADJ$W_PNA(R9)        ; Do the node addresses match?
            43    13              0E54  3010                 BEQL    100$                    ; Br if YES - must have come from
                                0E56  3011                                                 ;  the "Designated Router", skip it
5A    66    A8    D0              0E56  3012                 MOVL    LPD$L_CACHE(R8),R10     ; Else, get the CACHE table for LPD
            3D    13              0E5A  3013                 BEQL    100$                    ; Br if none available - leave now
54    FA    AA    3C              0E5C  3014                 MOVZWL  -6(R10),R4              ; Get number of entries in CACHE
                                0E60  3015                 ;
                                0E60  3016                 ;       Scan CACHE
                                0E60  3017                 ;
    53    8A    B1              0E60  3018 10$:            CMPW    (R10)+,R3               ; Node address in cache?
            2D    13              0E63  3019                 BEQL    60$                     ; Br if yes
```

```
                    8A   B5   0E65  3020          TSTW    (R10)+                          ; Skip timer cell
                 F6 54   F5   0E67  3021          SOBGTR  R4,10$                          ; Loop if more
                         0E6A  3022                ;
                         0E6A  3023                ;       CACHE scan failed, find empty cell and enter new Node
                         0E6A  3024                ;       address into the CACHE. If an empty cell is not found,
                         0E6A  3025                ;       then throw the oldest entry away!
                         0E6A  3026                ;
                         0E6A  3027                ;       Make sure the Intra-NI bit is set before entering in CACHE.
                         0E6A  3028                ;
              2B 55   05 E1   0E6A  3029          BBC     #TR4$V_RTFLG_INI,R5,100$ ; Br if Intra-NI packet, insert entry
              5A   66 A8 D0   0E6E  3030          MOVL    LPD$L_CACHE(R8),R10             ; Get the CACHE table for LPD, again
              54   FA AA 3C   0E72  3031          MOVZWL  -6(R10),R4                      ; Get size of CACHE table
                 53   5A D0   0E76  3032          MOVL    R10,R3                          ; Make a copy of the oldest entry
                         0E79  3033                                                      ; ..assume first is oldest
                    6A   D5   0E79  3034  30$:    TSTL    (R10)                           ; Empty entry?
                    12   13   0E7B  3035          BEQL    50$                             ; Br if yes
                 02 A3   02 AA B1   0E7D  3036          CMPW    2(R10),2(R3)                    ; Is this the new oldest?
                    03   14   0E82  3037          BGTR    40$                             ; Br if not
                 53   5A D0   0E84  3038          MOVL    R10,R3                          ; Else, set new oldest
                    8A   D5   0E87  3039  40$:    TSTL    (R10)+                          ; Skip to next
                   ED 54   F5   0E89  3040          SOBGTR  R4,30$                          ; Loop if more
                 5A   53   D0   0E8C  3041          MOVL    R3,R10                          ; Else, purge the oldest entry
                         0E8F  3042  50$:    ;
                         0E8F  3043                ;       Enter new Node Address into CACHE table.
                         0E8F  3044                ;
                    8A   61   B0   0E8F  3045          MOVW    (R1),(R10)+             ; Enter new node address
              8A   00000000'GF   B0   0E92  3046  60$:    MOVW    G^EXE$GL_ABSTIM,(R10)+  ; Enter current time
                         05   0E99  3047  100$:   RSB                                     ; Return to caller
                         0E9A  3048
```

```
                     0E9A  3050          .SBTTL  TR$RTRN_XMT_RTH - End-action routine for route-thru IRP's
                     0E9A  3051          .SBTTL  TR$RTRN_XMT_ECL - End-action routine for "ECL" IRP's
                     0E9A  3052          .SBTTL  TR$RTRN_XMT_TLK - End-action routine for "TALKER" IRP's
                     0E9A  3053
                     0E9A  3054 ;+
                     0E9A  3055 ;  TR$RTRN_XMT_RTH - Transmit I/O end-action routine for route-thru IRP's
                     0E9A  3056 ;  TR$RTRN_XMT_ECL - Transmit I/O end-action routine for "ECL" IRP's
                     0E9A  3057 ;  TR$RTRN_XMT_TLK - Transmit I/O end-action routine for "TALKER" IRP's
                     0E9A  3058 ;
                     0E9A  3059 ;
                     0E9A  3060 ;  End-action after Xmt IRP is returned due to I/O completion.  In general,
                     0E9A  3061 ;  each routine returns the "input packet limiter" resource, and the hello
                     0E9A  3062 ;  timer is reset if the transmit was successful.
                     0E9A  3063 ;
                     0E9A  3064 ;
                     0E9A  3065 ;  INPUTS:       R5      IRP ptr
                     0E9A  3066 ;                R4-R0   Scratch
                     0E9A  3067 ;
                     0E9A  3068 ;                IPL     4
                     0E9A  3069 ;
                     0E9A  3070 ;  OUTPUTS:      R5-R0   Garbage
                     0E9A  3071 ;
                     0E9A  3072 ;                IPL     4
                     0E9A  3073 ;
                     0E9A  3074 ;-
                     0E9A  3075                          .ENABL  LSB
                     0E9A  3076 TR$RTRN_XMT_TLK::                        ; HELLO message I/O completion
                     0E9A  3077          DSBINT  #NET$C_IPL              ; Raise to driver IPL
           07C0 8F    BB   0EA0  3078          PUSHR   #^M<R6,R7,R8,R9,R10>    ; Save regs
                     0EA4  3079
         52   14 A5   D0   0EA4  3080          MOVL    IRP$L_ASTPRM(R5),R2     ; Get RCB address
         58   10 A5   9A   0EA8  3081          MOVZBL  IRP$L_AST(R5),R8        ; Get LPD index
      58   28 B248    D0   0EAC  3082          MOVL    @RCB$L_PTR_LPD(R2)[R8],R8 ; Get LPD address
            1F A8    96   0EB1  3083          INCB    LPD$B_XMT_IPL(R8)       ; Return "input-packet-limiter" slot
                     0EB4  3084          ;
                     0EB4  3085          ;   For Broadcast Circuits, we will check to see if we are the
                     0EB4  3086          ;   "Designated Router" and if so, setup to send the "Broadcast
                     0EB4  3087          ;   Endnode Hello" message (in addition to the "Router Hello" we
                     0EB4  3088          ;   just sent).
                     0EB4  3089          ;
         23 38 A5    E9   0EB4  3090          BLBC    IRP$L_IOST1(R5),10$     ; If error, exit, but don't reset timer
   20 22 A8   0A     E1   0EB8  3091          BBC     #LPD$V_BC,LPD$W_STS(R8),259$ ; Br if not a BC
   1B 22 A8   0B     E4   0EBD  3092          BBSC    #LPD$V_XEND,LPD$W_STS(R8),259$ ; Br if we have already sent
                     0EC2  3093                                              ;  the "Broadcast Endnode Hello" msg
         51   2C A8   3C   0EC2  3094          MOVZWL  LPD$W_DRT(R8),R1        ; Get designated router ADJ index
      51   2C B241    D0   0EC6  3095          MOVL    @RCB$L_PTR_ADJ(R2)[R1],R1 ; Get ADJ address
            04 A1    B1   0ECB  3096          CMPW    ADJ$W_PNA(R1),-         ; Are we the Designated Router?
            0E A2         0ECE  3097                  RCB$W_ADDR(R2)          ;
                     0B   12   0ED0  3098          BNEQ    259$            ; Br if not - reset timer
                     16 A8   B4   0ED2  3099          CLRW    LPD$W_TIM_TLK(R8)       ; Else, force hello msg next time
   22 A8   0800 8F   A8   0ED5  3100          BISW    #LPD$M_XEND,LPD$W_STS(R8) ; Send the "Broadcast Endnode" hello
                        77   11   0EDB  3101 10$:     BRB     30$             ; Don't reset timer
                      006F   31   0EDD  3102 259$:    BRW     25$             ; Reset the "hello" timer
                     0EE0  3103
                     0EE0  3104 TR$RTRN_XMT_RTH::                        ; Route-thru I/O completion
                     0EE0  3105          DSBINT  #NET$C_IPL              ; Raise to driver IPL
           07C0 8F    BB   0EE6  3106          PUSHR   #^M<R6,R7,R8,R9,R10>    ; Save regs
```

```
                     0EEA  3107                                ;
   52   14 A5   D0  0EEA  3108          MOVL    IRP$L_ASTPRM(R5),R2      ; Get RCB address
   58   10 A5   9A  0EEE  3109          MOVZBL  IRP$L_AST(R5),R8         ; Get LPD index
58  28 B248   D0  0EF2  3110          MOVL    @RCB$L_PTR_LPD(R2)[R8],R8 ; Get LPD address
   59 38 A5   E9  0EF7  3111          BLBC    IRP$L_IOST1(R5),30$      ; If LBC then I/O error
                     0EFB  3112          BUMP    L,LPD$L_CNT_TPS(R8)      ; Update 'transit packets sent'
         44   11  0F04  3113          BRB     20$                      ; Continue in common
                     0F06  3114
                     0F06  3115 TR$RTRN_XMT_ECL::                     ; ECL xmt I/O completion
                     0F06  3116          DSBINT  #NET$C_IPL               ; Raise IPL
    07C0 8F   BB  0F0C  3117          PUSHR   #^M<R6,R7,R8,R9,R10>     ; Save regs
                     0F10  3118                                ;
   52   14 A5   D0  0F10  3119          MOVL    IRP$L_ASTPRM(R5),R2      ; Get RCB pointer
   58   10 A5   9A  0F14  3120          MOVZBL  IRP$L_AST(R5),R8         ; Get LPD index
58  28 B248   D0  0F18  3121          MOVL    @RCB$L_PTR_LPD(R2)[R8],R8 ; Get LPD address
                     0F1D  3122
                     0F1D  3123          .IF     DF,JNX$$$
                     0F1D  3124
    57   08   D0  0F1D  3125          MOVL    #8,R7                    ; Set length of IOSB
 51   38 A5   9E  0F20  3126          MOVAB   IRP$L_IOST1(R5),R1       ; Journal the IOSB quadword
    50   01   90  0F24  3127          MOVB    #1,R0                    ; Set journal type = Transmit complete
      0264   30  0F27  3128          BSBW    TR_FILL_JNX              ; Store journal record
                     0F2A  3129
                     0F2A  3130          .ENDC
                     0F2A  3131
    1F A8   96  0F2A  3132          INCB    LPD$B_XMT_IPL(R8)        ; Return "input-packet-limiter" slot
 50   24 A5   D0  0F2D  3133          MOVL    IRP$L_IOSB(R5),R0        ; Get buffer
    24 A5   D4  0F31  3134          CLRL    IRP$L_IOSB(R5)           ; Detach it from the IRP
                     0F34  3135          ;
                     0F34  3136          ;
                     0F34  3137          ;  Deliver end-action status to the ECL issuing the transmit.  It
                     0F34  3138          ;  is the responsibility of the ECL routine to consume the R0
                     0F34  3139          ;  buffer -- deallocate it, requeue it, etc.  Attaching R0 to
                     0F34  3140          ;  IRP$L_IOSB will cause it to be deallocate on return (see the code
                     0F34  3141          ;  in TR_RTRN_IRP).
                     0F34  3142          ;
                     0F34  3143          ;
                     0F34  3144          ;  Call with:  R5       IRP address
                     0F34  3145          ;              R4,R3    Scratch
                     0F34  3146          ;              R2       RCB address
                     0F34  3147          ;              R1       Scratch
                     0F34  3148          ;              R0       CXB address
                     0F34  3149          ;
                     0F34  3150          ;              CXB$L_ENDACTION(R0) has been repaired
                     0F34  3151          ;
                     0F34  3152          ;  On return from ECL:
                     0F34  3153          ;
                     0F34  3154          ;              R4,R3,R1,R0     may be garbage.
                     0F34  3155          ;
                     0F34  3156          ;              All other registers must be unchanged.
                     0F34  3157          ;
                     0F34  3158          ;
    78 B5   16  0F34  3159          JSB     @IRP$L_SAVD_RTN(R5)      ; Deliver status to ECL layer
19 38 A5   E9  0F37  3160          BLBC    IRP$L_IOST1(R5),30$      ; If LBC then I/O was not successful
                     0F3B  3161          BUMP    L,LPD$L_CNT_DPS(R8)      ; Bump 'departing pkts sent'
                     0F44  3162          INCPMS  DEPLOCPR                 ; ... and the PMS database too
      0A   E0  0F4A  3163 20$:    BBS     #LPD$V_BC,-              ; If this is a broadcast circuit,
```

```
05 22 A8          0F4C  3164                    LPD$W_STS(R8),30$       ; then never reset talker (so that
                  0F4F  3165                                            ; Router Hellos are sent regularly)
   18 A8    B0    0F4F  3166 25$:     MOVW       LPD$W_INT_TLK(R8),-     ; Reset talker interval
   16 A8          0F52  3167                    LPD$W_TIM_TLK(R8)       ;
      08    10    0F54  3168 30$:     BSBB       TR_RTRN_IRP             ; Return IRP to the Xmit pool
                  0F56  3169                                            ;
07C0 8F    BA    0F56  3170          POPR       #^M<R6,R7,R8,R9,R10>    ; Restore reg
                  0F5A  3171          ENBINT                             ; Restore IPL
            05    0F5D  3172          RSB                                ; Return to Exec
                  0F5E  3173
                  0F5E  3174                    .DSABL  LSB
                  0F5E  3175
                  0F5E  3176
```

```
                                0F5E  3178          .SBTTL  TR_RTRN_IRP     - Recycle IRP Xmit IRP pool
                                0F5E  3179
                                0F5E  3180 ;+
                                0F5E  3181 ;  TR_RTRN_IRP  - Recycle (Rcv or Xmt) IRP to transmit IRP pool
                                0F5E  3182 ;
                                0F5E  3183 ;
                                0F5E  3184 ;  If the low bit is clear in IRP$L_IOST1 and the LPD is still marked "active"
                                0F5E  3185 ;  then an "LPD-down" event is generated.
                                0F5E  3186 ;
                                0F5E  3187 ;  Otherwise, the used resources are returned.  If a fork process is awaiting
                                0F5E  3188 ;  any of these resources,  its wait state is advanced and may be reactivated.
                                0F5E  3189 ;
                                0F5E  3190 ;
                                0F5E  3191 ;  INPUTS:       R8      LPD pointer
                                0F5E  3192 ;                R5      IRP pointer
                                0F5E  3193 ;                R2      RCB pointer
                                0F5E  3194 ;
                                0F5E  3195 ;                IRP$L_IOST1(R5) low bit set if I/O was succesful
                                0F5E  3196 ;                IRP$L_IOSB(R5)  points to CXB to be deallocated, zero if none
                                0F5E  3197 ;
                                0F5E  3198 ;  OUTPUTS:      R8-R6   Garbage
                                0F5E  3199 ;                R5      Zero
                                0F5E  3200 ;                R4-R0   Garbage
                                0F5E  3201 ;
                                0F5E  3202 ;-
                                0F5E  3203 TR_RTRN_IRP:                             ; Return IRP to Xmit pool
          53    55    D0        0F5E  3204          MOVL    R5,R3                   ; Copy the IRP address
                                0F61  3205          ;
                                0F61  3206          ;
                                0F61  3207          ;  Deallocate the attached CXB, if any
                                0F61  3208          ;
                                0F61  3209          ;
          50    24 A5   D0      0F61  3210          MOVL    IRP$L_IOSB(R5),R0       ; Get the CXB
                   09   13      0F65  3211          BEQL    10$                     ; If EQL then none
                24 A5   D4      0F67  3212          CLRL    IRP$L_IOSB(R5)          ; Nullify CXB pointer
00000000'GF     16              0F6A  3213          JSB     G^COM$DRVDEALMEM        ; Deallocate the CXB
                                0F70  3214 10$:     ;
                                0F70  3215          ;
                                0F70  3216          ;  If LBS in IOST1 then I/O was successful, branch to recycle the IRP.
                                0F70  3217          ;  Otherwise, the I/O failed -- assume the datalink is down, clear
                                0F70  3218          ;  the LPD$V_ACTIVE bit:
                                0F70  3219          ;
                                0F70  3220          ;      If it was set the generate an "LPD-down" event.  This event
                                0F70  3221          ;      consumes the IRP since it is queue to the ACP to signal the
                                0F70  3222          ;      event.
                                0F70  3223          ;
                                0F70  3224          ;      If it was already clear, then the "LPD-down" event was already
                                0F70  3225          ;      generated.  Recycle this IRP to the Xmit pool, even if it is
                                0F70  3226          ;      a Rcv IRP.
                                0F70  3227          ;
                                0F70  3228          ;
                                0F70  3229          ;      NOTE:   There is only one RCV IRP ever queued to a datalink.
                                0F70  3230          ;              Because of this, because the "LPD-down" event is
                                0F70  3231          ;              generated only once, and because all failed I/O packets
                                0F70  3232          ;              -- Rcv and Xmt -- sent here, it makes no difference
                                0F70  3233          ;              which type of IRP is used to signal the "LPD-down"
                                0F70  3234          ;              event and which are used to return the xmitter
```

```
                     0F70  3235          ;                resources.
                     0F70  3236          ;
                     0F70  3237          ;
                     0F70  3238          ;
 05 38 A5    E8      0F70  3239          BLBS    IRP$L_IOST1(R5),15$     ; If LBC then I/O was successful
       00    E4      0F74  3240          BBSC    #LPD$V_ACTIVE,-         ; If BS then "LPD-down" event has
          22 A8      0F76  3241                   LPD$W_STS(R8),-         ; not yet been processed.
             5E      0F78  3242                   TR_LPD_DOWN             ; Signal "LPD-down"
          1C A8 97   0F79  3243 15$:     DECB    LPD$B_IRPCNT(R8)        ; Return the queue space
             4B 13   0F7C  3244          BEQL    100$                    ; If EQL then report "circuit rundown"
                     0F7E  3245          ;
                     0F7E  3246          ;
                     0F7E  3247          ;   Account for IRP being returned and advance a single waiting
                     0F7E  3248          ;   solicitor if possible.  Both the "input-packet-limiter" and
                     0F7E  3249          ;   the "square-root-limit" restrictions must be checked before
                     0F7E  3250          ;   reactivating a solicitor since these limits may have been
                     0F7E  3251          ;   changed asynchronously be NETACP while this IRP was being
                     0F7E  3252          ;   processed by the datalink.
                     0F7E  3253          ;
                     0F7E  3254          ;
          1C A8 91   0F7E  3255          CMPB    LPD$B_IRPCNT(R8),-      ; Does "square-root-limiter" allow
          1E A8      0F81  3256                   LPD$B_XMT_SRL(R8)       ; another I/O ?
             15 14   0F83  3257          BGTR    30$                     ; If GTR then no
          1F A8 95   0F85  3258          TSTB    LPD$B_XMT_IPL(R8)       ; Does "input-packet-limiter" allow it?
             10 15   0F88  3259          BLEQ    30$                     ; If LEQ then no
 50    00 B8 0F      0F8A  3260          REMQUE  @LPD$Q_REQ_WAIT(R8),R0  ; Get the waiting process
             0A 1D   0F8E  3261          BVS     30$                     ; If VS then none
          1F A8 97   0F90  3262          DECB    LPD$B_XMT_IPL(R8)       ; Consume "input-packet-limiter" slot
          1C A8 96   0F93  3263          INCB    LPD$B_IRPCNT(R8)        ; Account for IRP to be queued
 50 B2       60 0E   0F96  3264          INSQUE  (R0),@RCB$Q_IRP_WAIT+4(R2)      ; Move to IRP wait state
                     0F9A  3265 30$:     ;
                     0F9A  3266          ;
                     0F9A  3267          ;   Reactivate a solicitor waiting for an IRP if possible.  If none
                     0F9A  3268          ;   are waiting, return the IRP -- shrinking the IRP free queue if
                     0F9A  3269          ;   necessary.
                     0F9A  3270          ;
                     0F9A  3271          ;
 55    4C B2    0F    0F9A  3272          REMQUE  @RCB$Q_IRP_WAIT(R2),R5  ; Get oldest waiting process
             11 1C   0F9E  3273          BVC     50$                     ; If VC then got one
       62    63 0E   0FA0  3274          INSQUE  (R3),RCB$Q_IRP_FREE(R2) ; Queue the IRP
       0080  C2 B1   0FA3  3275          CMPW    RCB$W_CUR_PKT(R2),-     ; Does IRP pool needs reducing?
       0082  C2      0FA7  3276                  RCB$W_MAX_PKT(R2)       ;
             1A 1B   0FAA  3277          BLEQU   60$                     ; If EQL, no
          0157 30    0FAC  3278          BSBW    TR$ADJUST_IRP           ; Adjust Xmit IRP pool
             15 11   0FAF  3279          BRB     60$                     ; Continue
                     0FB1  3280          ;
   58    10 A5 9A    0FB1  3281 50$:     MOVZBL  FKB$L_FR3(R5),R8        ; Get LPD index
58    28 B248 D0     0FB5  3282          MOVL    @RCB$[_PTR_LPD(R2)[R8],R8 ; Get LPD address
   59    14 A5 3C    0FBA  3283          MOVZWL  FKB$L_FR4(R5),R9        ; Get ADJ index
59    2C B249 D0     0FBE  3284          MOVL    @RCB$[_PTR_ADJ(R2)[R9],R9 ; Get ADJ address
                     0FC3  3285          ;
          F56F 30    0FC3  3286          BSBW    TR$GRANT                ; Restart solicitor
                     0FC6  3287 60$:     ;
                     0FC6  3288          ;
                     0FC6  3289          ;   Done
                     0FC6  3290          ;
                     0FC6  3291          ;
```

```
                                                     I  7
                     55   D4   0FC6  3292          CLRL    R5                      ; Nullify IRP pointer
                          05   0FC8  3293          RSB                             ; Done
                               0FC9  3294
                               0FC9  3295
                               0FC9  3296 100$:     ;
                               0FC9  3297           ;
                               0FC9  3298           ;   Report that the LPD has been "run-down"
                               0FC9  3299           ;
                               0FC9  3300           ;
                0080 C2   B7   0FC9  3301          DECW    RCB$W_CUR_PKT(R2)       ; Account for xmit IRP going away
                  1C A8   96   0FCD  3302          INCB    LPD$B_IRPCNT(R8)        ; Prevent LPD activity until CRD
                               0FD0  3303                                          ; successfully makes it to NETACP
                50    0B   90   0FD0  3304          MOVB    #NETMSG$C_CRD,R0        ; Setup event code for NETACP
                     0072   30   0FD3  3305          BSBW    TR$QUE_IRP_AQB          ; Queue IRP to NETACP
                          05   0FD6  3306          RSB                             ;
                               0FD7  3307
                               0FD7  3308
```

```
                                                                J 7
                     0FD7  3310          .SBTTL  TR_LPD_DOWN     - Process "LPD down" event
                     0FD7  3311
                     0FD7  3312 ;+
                     0FD7  3313 ;  TR_LPD_DOWN     - Process "LPD down" event
                     0FD7  3314 ;
                     0FD7  3315 ;
                     0FD7  3316 ;  The LPD is marked inactive.  All suspended fork processes waiting to
                     0FD7  3317 ;  transmit over the datalink are reactivate with their request to xmit
                     0FD7  3318 ;  denied.
                     0FD7  3319 ;
                     0FD7  3320 ;
                     0FD7  3321 ;  INPUTS:       R8      LPD address
                     0FD7  3322 ;                R5      IRP address
                     0FD7  3323 ;                R2      RCB address
                     0FD7  3324 ;
                     0FD7  3325 ;  OUTPUTS:      R5      Zero
                     0FD7  3326 ;                R0      Destroyed
                     0FD7  3327 ;
                     0FD7  3328 ;                All other registers are unchanged.
                     0FD7  3329 ;
                     0FD7  3330 ;-
                     0FD7  3331 TR_LPD_DOWN:                                 ; Process "LPD down" event
           02FE 8F   BB 0FD7  3332          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R9> ; Save regs
                     0FDB  3333                                             ;
            1C A8    97 0FDB  3334          DECB    LPD$B_IRPCNT(R8)        ; Account for IRP being returned
                     0FDE  3335
                     0FDE  3336          ;
                     0FDE  3337          ;   Deallocate the LPD CACHE, if present.
                     0FDE  3338          ;
               52    DD 0FDE  3339          PUSHL   R2                      ; Save RCB address
         50    66 A8    D0 0FE0  3340          MOVL    LPD$L_CACHE(R8),R0      ; Get CACHE address
                  0C    13 0FE4  3341          BEQL    10$                     ; Br if none
            50    0C    C2 0FE6  3342          SUBL    #12,R0                  ; Get start address of CACHE
               66 A8    D4 0FE9  3343          CLRL    LPD$L_CACHE(R8)         ; Zero CACHE pointer
00000000'GF      16    FB 0FEC  3344          JSB     G^EXE$DEANONPAGED       ; Deallocate the pool
               52 8ED0    0FF2  3345 10$:     POPL    R2                      ; Restore RCB address
                          0FF5  3346
                          0FF5  3347          ;
                          0FF5  3348          ;
                          0FF5  3349          ;   Reactivate all solicitors associated with this LPD and which are
                          0FF5  3350          ;   waiting for an IRP, denying each of them permission to transmit.
                          0FF5  3351          ;
                          0FF5  3352          ;
         57    4C A2    9E 0FF5  3353          MOVAB   RCB$Q_IRP_WAIT(R2),R7   ; Get listhead
            55    57    D0 0FF9  3354          MOVL    R7,R5                   ; Make copy
            56    55    D0 0FFC  3355 30$:     MOVL    R5,R6                   ; Advance last fork block ptr
            55    66    D0 0FFF  3356 40$:     MOVL    (R6),R5                 ; Get next fork block
            57    55    D1 1002  3357          CMPL    R5,R7                   ; Listhead?
                  12    13 1005  3358          BEQL    50$                     ; If EQL then done
               10 A5    91 1007  3359          CMPB    FKB$L_FR3(R5),-         ; Associated with this LPD?
               20 A8           100A  3360                  LPD$B_PTH_INX(R8)   ;
                  EE    12 100C  3361          BNEQ    30$                     ; If NEQ then no
            55    65    0F 100E  3362          REMQUE  (R5),R5                 ; Deque it
               1F A8    96 1011  3363          INCB    LPD$B_XMT_IPL(R8)       ; Return request slot
                F513    30 1014  3364          BSBW    TR$DENY                 ; Reactivate with failure
                  E6    11 1017  3365          BRB     40$                     ; Loop
                        1019  3366 50$:     ;
```

```
                                                K 7
                     1019  3367          ;
                     1019  3368          ;  Reactivate all solicitors waiting for room on the datalink queues,
                     1019  3369          ;  denying each permission to transmit.
                     1019  3370          ;
                     1019  3371          ;
55   00 B8   0F      1019  3372          REMQUE  @LPD$Q_REQ_WAIT(R8),R5  ; Get next fork block
        05   1D      101D  3373          BVS     60$                     ; If VS then none
      F508   30      101F  3374          BSBW    TR$DENY                 ; Reactivate with failure
        F5   11      1022  3375          BRB     50$                     ; Loop
                     1024  3376          ;
   02FE 8F   BA      1024  3377 60$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R9> ; Restore regs
50    04   90      1028  3378          MOVB    #NETMSG$C_IRP,R0        ; Setup the event
        1B   10      102B  3379          BSBB    TR$QUE_IRP_AQB          ; Signal the ACP, clear R5
             05      102D  3380          RSB                             ; Done
                     102E  3381
```

```
                                        .SBTTL  TR$GIVE_TO_ACP  - ECL entry to queue a buffer to the ACP
                                        .SBTTL  TR$QUE_WQE_AQB  - Queue WQE to AQB
                                        .SBTTL  TR$QUE_IRP_AQB  - Queue "LPD down" IRP to AQB

                                ;+
                                ;  TR$GIVE_TO_ACP       - ECL entry to queue a buffer to the ACP
                                ;  TR$QUE_WQE_AQB       - Queue WQE to AQB
                                ;  TR$QUE_IRP_AQB       - Queue IRP to AQB - RCB$W_TRANS was already inc'd
                                ;
                                ;  Setup the common fields in the WQE and queue it to the AQB.
                                ;
                                ;  The action here is to fork before queueing the IRP since SCH$WAKE may
                                ;  have to be called.  SCH$WAKE assumes it is called at IPL$_SYNC
                                ;
                                ;  In the case of TR$QUE_IRP_AQB, the IRP has already been accounted for,
                                ;  neither the TRANSaction count nor the AQB_CNT will have to be incremented.
                                ;
                                ;  INPUTS:      R9      ADJ address or zero (only if TR$QUE_AWQE_AQB)
                                ;               R8      LPD address
                                ;               R5      WQE address - block to be queued to NETACP
                                ;               R2      RCB address
                                ;               R1      Not used
                                ;               R0      If TR$QUE_IRP_AQB then the NETMSG$... event code
                                ;                       Else, not looked at
                                ;
                                ;  OUTPUTS:     R5      0
                                ;
                                ;               All other registers are preserved.
                                ;-
                                                .ENABL  LSB
                                TR$QUE_WQE_AQB:                                 ; Queue WQE (i.e., CXB) to AQB
14 91                                   CMPB    #NET$C_MAX_WQE,-                ; Can we insert more entries
00A9 C2                                  RCB$B_AQB_CNT(R2)                      ;   on AQB?
62 19                                   BLSS    50$                             ; Br if no, deallocate WQE
00A9 C2 96                              INCB    RCB$B_AQB_CNT(R2)               ; Increment count of new entries
                                                                                ;   inserted on AQB
12 A5 20 A8 B0                          MOVW    LPD$W_PTH(R8),WQE$W_REQIDT(R5)  ; Remember datalink i.d.
0C A2 B6                                INCW    RCB$W_TRANS(R2)                 ; Count new transaction
1A 11                                   BRB     15$                             ; Continue, don't convert block
                                                                                ; structure type
                                ;
                                TR$GIVE_TO_ACP::                                ; ECL entry pass block to ACP
0C A2 B6                                INCW    RCB$W_TRANS(R2)                 ; Else, count new transaction
11 11                                   BRB     10$                             ; Continue
                                ;
                                TR$QUE_IRP_AQB:                                 ; Queue IRP (as WQE) to AQB
20 A5 B4                                CLRW    WQE$W_ADJ_INX(R5)               ; No ADJ index available
14 A5 38 A5 9E                          MOVAB   IRP$L_IOST1(R5),WQE$L_PM2(R5)   ; Store ptr to IOSB image
10 A5 50 90                             MOVB    R0,WQE$B_EVT(R5)                ; Setup the event
12 A5 20 A8 B0                          MOVW    LPD$W_PTR(R8),WQE$W_REQIDT(R5)  ; Remember datalink i.d.
0A A5 17 90                     10$:    MOVB    S^#DYN$C_NET,WQE$B_TYPE(R5)     ; Convert the IRP to a WQE
                                ;
```

```
                1F   BB  105D  3440 15$:    PUSHR   #^M<R0,R1,R2,R3,R4>                     ; Save regs
                     1F  105F  3441                                                         ;
      18 A5    10 A2   D0  105F  3442         MOVL    RCB$L_AQB(R2),WQE$L_PM2+4(R5)       ; Save AQB address
         0B A5    06   90  1064  3443         MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R5)       ; Setup fork IPL
         53    10 A5   7D  1068  3444         MOVQ    FKB$L_FR3(R5),R3                    ; Prevent EXE$FORK from
                     6C  1068  3445                                                         ; destroying these fields
                05   10  106C  3446         BSBB    30$                                 ; Create fork process
                55   D4  106E  3447         CLRL    R5                                  ; Prevent access to this block
                     70  1070  3448                                                         ;
                1F   BA  1070  3449         POPR    #^M<R0,R1,R2,R3,R4>                 ; Restore regs
                05   72  1072  3450         RSB                                         ; Done
                     73  1073  3451
                     73  1073  3452
                     73  1073  3453
  00000000'GF   16   73  1073  3454 30$:    JSB     G^EXE$FORK                          ; Create fork process
                     79  1079  3455                                                         ;
                     79  1079  3456         DSBINT  #IPL$_SYNCH                         ; We're back. Sync with SCH$WAKE
    54    18 A5   D0  107F  3457         MOVL    WQE$L_PM2+4(R5),R4                  ; Get AQB address
    04 B4    65   0E  1083  3458         INSQUE  (R5),@AQB$L_ACPQBL(R4)              ; Inert IRP at end of queue
             0A   12  1087  3459         BNEQ    40$                                 ; Br unless first
    51    0C A4   D0  1089  3460         MOVL    AQB$L_ACPPID(R4),R1                 ; Get PID
  00000000'GF   16  108D  3461         JSB     G^SCH$WAKE                          ; Wake the ACP
                     93  1093  3462 40$:    ENBINT                                      ; Restore IPL
                     96  1096  3463                                                         ;
                05   96  1096  3464         RSB                                         ; Done
                     97  1097  3465
                     97  1097  3466         ;
                     97  1097  3467         ;   Too many entries on AQB queue
                     97  1097  3468         ;
    50    55   D0  1097  3469 50$:    MOVL    R5,R0                               ; Copy WQE address
  00000000'GF   16  109A  3470         JSB     G^COM$DRVDEALMEM                    ; Deallocate it
                05  10A0  3471         RSB                                         ; Return to caller
                    10A1  3472
                    10A1  3473                 .DSABL  LSB
```

```
                     10A1  3475          .SBTTL  TR$LOC_DLL_XMT  - "Local" datalink driver transmit
                     10A1  3476          .SBTTL  TR$LOC_DLL_RCV  - "Local" datalink driver receive
                     10A1  3477
                     10A1  3478 ;+
                     10A1  3479 ;  TR$LOC_DLL_XMT  - "Local" datalink driver transmit
                     10A1  3480 ;  TR$LOC_DLL_RCV  - "Local" datalink driver receive
                     10A1  3481 ;
                     10A1  3482 ;
                     10A1  3483 ;  This routine simulates a datalink driver.  It is used to allow the Transport
                     10A1  3484 ;  layer to handle IRPs for ECL-ECL communication in a manner consistent with
                     10A1  3485 ;  the remainder of the Datalink layer.  Both the transmitter and the receiver
                     10A1  3486 ;  appear to "buffered" (as opposed to "direct") I/O.
                     10A1  3487 ;
                     10A1  3488 ;  It appears as a line constantly in "loopback".  The receive IRP is made to
                     10A1  3489 ;  point to the buffer carried by the transmit IRP.  In order to get away with
                     10A1  3490 ;  this, the XM$V_STS_BUFFAIL bit must be set in the receive's IRP$L_IOST2
                     10A1  3491 ;  field -- this prevents it the buffer from being consumed and is still
                     10A1  3492 ;  attached to the IRP when it is requeued for another receive operation.
                     10A1  3493 ;
                     10A1  3494 ;
                     10A1  3495 ;       NOTE:   Sharing the buffer this way only works if the receive is
                     10A1  3496 ;               completed before its corresponding transmit.  Also, it can
                     10A1  3497 ;               only work if the buffer is never sent to NETACP -- this
                     10A1  3498 ;               restriction is enforced by the fact that the "local"
                     10A1  3499 ;               datalink is only used to carry ECL messages.
                     10A1  3500 ;
                     10A1  3501 ;
                     10A1  3502 ;  The pertinent IRP fields are as follows:
                     10A1  3503 ;
                     10A1  3504 ;  On input to this routine:
                     10A1  3505 ;
                     10A1  3506 ;                          Rcv's                       Xmt's
                     10A1  3507 ;                          -----                       -----
                     10A1  3508 ;       IRP$L_SVAPTE       Garbage                     Buffer pointer
                     10A1  3509 ;       IRP$W_BCNT         Garbage                     Message size
                     10A1  3510 ;       IRP$L_IOST1        Garbage                     Garbage
                     10A1  3511 ;       IRP$L_IOST2        Garbage                     Garbage
                     10A1  3512 ;
                     10A1  3513 ;  When sent to I/O completion:
                     10A1  3514 ;
                     10A1  3515 ;       IRP$L_SVAPTE       Buffer pointer              Buffer pointer
                     10A1  3516 ;       IRP$W_BCNT         Message size                Message size
                     10A1  3517 ;       IRP$L_IOST1        SS$_NORMAL in low word      SS$_NORMAL in low word
                     10A1  3518 ;                          IRP$W_BCNT in high word     IRP$W_BCNT in high word
                     10A1  3519 ;       IRP$L_IOST2        XM$M_STS_ACTIVE!-           XM$M_STS_ACTIVE
                     10A1  3520 ;                          XM$M_STS_BUFFAIL
                     10A1  3521 ;
                     10A1  3522 ;
                     10A1  3523 ;  INPUTS:      R3              IRP address
                     10A1  3524 ;
                     10A1  3525 ;  OUTPUTS:     R5-R0           Garbage
                     10A1  3526 ;
                     10A1  3527 ;-
                     10A1  3528 TR$LOC_DLL_XMT:                                     ; "Local" datalink driver xmt'r
          51  53 D0  10A1  3529          MOVL    R3,R1                             ; Copy IRP address
       52  14 A1 D0  10A4  3530          MOVL    IRP$L_ASTPRM(R1),R2               ; Get RCB
       53  3C B2 0F  10A8  3531          REMQUE  @RCB$Q_LOC_RCV(R2),R3             ; Get a waiting receive
```

```
                17   1C  10AC  3532          BVC     XFER                                    ; If VC then got one
          48 B2 61   0E  10AE  3533          INSQUE  (R1),@RCB$Q_LOC_XMT+4(R2)               ; Else queue the IRP
                05   10B2  3534          RSB
                     10B3  3535
                     10B3  3536 TR$LOC_DLL_RCV:                                             ; "Local datalink driver rcv'r
             2C A3   D4  10B3  3537          CLRL    IRP$L_SVAPTE(R3)                        ; Erase former buffer ptr
          52 14 A3   D0  10B6  3538          MOVL    IRP$L_ASTPRM(R3),R2                     ; Get RCB
          51 44 B2   0F  10BA  3539          REMQUE  @RCB$Q_LOC_XMT(R2),R1                   ; Get a waiting transmit
                05   1C  10BE  3540          BVC     XFER                                    ; If VC then got one
          40 B2 63   0E  10C0  3541          INSQUE  (R3),@RCB$Q_LOC_RCV+4(R2)               ; Else queue the IRP
                05   10C4  3542          RSB
                     10C5  3543
                     10C5  3544 XFER:    ;
                     10C5  3545          ;
                     10C5  3546          ;   Setup IRP's for I/O completion.  Let the RCV IRP temporarily
                     10C5  3547          ;   share the XMT IRP's CXB in order to deliver the received message
                     10C5  3548          ;   to the ECL layer.
                     10C5  3549          ;
                     10C5  3550          ;
             2C A1   D0  10C5  3551          MOVL    IRP$L_SVAPTE(R1),-                      ; Setup Rcvr buffer ptr
             2C A3       10C8  3552                  IRP$L_SVAPTE(R3)                        ;
       32 A3 32 A1   B0  10CA  3553          MOVW    IRP$W_BCNT(R1),IRP$W_BCNT(R3)           ; Setup size of message
          38 A1 00'  B0  10CF  3554          MOVW    S^#SS$_NORMAL,IRP$L_IOST1(R1)           ; Setup I/O status
       3A A1 32 A1   B0  10D3  3555          MOVW    IRP$W_BCNT(R1),IRP$L_IOST1+2(R1)        ; Setup xfer size
       38 A3 38 A1   D0  10D8  3556          MOVL    IRP$L_IOST1(R1),IRP$L_IOST1(R3)         ; Copy XMT status to RCV IRP
    3C A1  0800 8F   3C  10DD  3557          MOVZWL  #XM$M_STS_ACTIVE,IRP$L_IOST2(R1)        ; Setup Xmtter device status
    3C A3  1800 8F   3C  10E3  3558          MOVZWL  #XM$M_STS_BUFFAIL!-                     ; Set Rcvr device status
                     10E9  3559                   XM$M_STS_ACTIVE,IRP$L_IOST2(R3)        ; - BUFFAIL is crucial since
                     10E9  3560                                                           ;   rcvr doesn't own the CXB
                     10E9  3561          ;
                     10E9  3562          ;
                     10E9  3563          ;   The order here is crucial.  The RCV must be posed before the XMT
                     10E9  3564          ;   so that the XMT end-action routine doesn't deallocate the buffer
                     10E9  3565          ;   before the RCV'r has had a chance to look at it.  Note that the
                     10E9  3566          ;   BUFFAIL flag is set so that the RCV'r cannot take the buffer.
                     10E9  3567          ;
                     10E9  3568          ;
                03   10  10E9  3569          BSBB    POST                                    ; Post the RCV IRP
             53 51   D0  10EB  3570          MOVL    R1,R3                                   ; Get the XMT IRP for posting
          0A A3 0A   91  10EE  3571 POST:    CMPB    S^#DYN$C_IRP,IRP$B_TYPE(R3)             ; IRP ?
                0E   12  10F2  3572          BNEQ    70$                                     ; If not, bug
    50 00000000'GF   9E  10F4  3573          MOVAB   G^IOC$GL_PSBL,R0                        ; Get I/O post back ptr address
             90 63   0E  10FB  3574          INSQUE  (R3),@(R0)+                             ; Queue the IRP
                     10FE  3575          SOFTINT #IPL$_IOPOST                            ; ALWAYS Post an interrupt due
                     1101  3576                                                           ; to an obscure system bug
                05   1101  3577 60$:     RSB                                             ; Done
                     1102  3578
                     1102  3579 70$:     BUG_CHECK NETNOSTATE,FATAL                      ; Bugcheck
                     1106  3580
```

```
                                1106  3582          .SBTTL  TR$ADJUST_IRP   - Adjust the number of IRPs in the pool
                                1106  3583
                                1106  3584 ;+
                                1106  3585 ;  TR$ADJUST_IRP        - Adjust the number of IRPs in the pool
                                1106  3586 ;
                                1106  3587 ;
                                1106  3588 ;  The number of free IRPs are adjusted in whatever direction necessary to
                                1106  3589 ;  bring  RCB$W_CUR_PKT  closer to  RCB$W_MAX_PKT.
                                1106  3590 ;
                                1106  3591 ;
                                1106  3592 ;  INPUTS:       R2      RCB pointer
                                1106  3593 ;
                                1106  3594 ;  OUTPUTS:      R0      Low bit clear if free queue is empty.
                                1106  3595 ;                        Low bit set otherwise.
                                1106  3596 ;
                                1106  3597 ;
                                1106  3598 ;                 All other registers are preserved.
                                1106  3599 ;
                                1106  3600 ;-
                                1106  3601 TR$ADJUST_IRP:                              ; Adjust IRP pool
                     53   DD    1106  3602          PUSHL   R3                        ; Save reg
                                1108  3603                                           ;
                0080 C2   B1    1108  3604 10$:     CMPW    RCB$W_CUR_PKT(R2),-      ; See what adjustments are needed
                0082 C2         110C  3605                  RCB$W_MAX_PKT(R2)        ;
                     29   13    110F  3606          BEQL    50$                      ; Br if none
                     12   1A    1111  3607          BGTRU   30$                      ; Br if decrease is needed
                     33   10    1113  3608          BSBB    TR$ALLOC_IRP             ; Get an IRP if possible
                  22 50   E9    1115  3609          BLBC    R0,50$                   ; Br on failure
             00 B2    63   0E    1118  3610          INSQUE  (R3),@RCB$Q_IRP_FREE(R2) ; Insert the IRP onto the free list
                0080 C2   B6    111C  3611          INCW    RCB$W_CUR_PKT(R2)        ; Account for IRP
                  0C A2   B6    1120  3612          INCW    RCB$W_TRANS(R2)          ; Here, too
                     15   11    1123  3613          BRB     50$                      ; Only allocate 1 at a time
                                1125  3614
          50    00 B2    0F    1125  3615 30$:     REMQUE  @RCB$Q_IRP_FREE(R2),R0   ; Get IRP if any
                     0F   1D    1129  3616          BVS     50$                      ; If VS then none
                   0080 C2   B7    112B  3617          DECW    RCB$W_CUR_PKT(R2)        ; Account for the IRP
                     0C A2   B7    112F  3618          DECW    RCB$W_TRANS(R2)          ; Account for it here, too
          00000000'GF   16    1132  3619          JSB     G^COM$DRVDEALMEM         ; Deallocate it
                     CE   11    1138  3620          BRB     10$                      ; Try again
                                113A  3621 50$:     ;
                                113A  3622 ;   Return a flag to the caller indicating if the queue is empty
                                113A  3623 ;
                     50   94    113A  3624          CLRB    R0                       ; Indicate empty
                     62   D1    113C  3625          CMPL    RCB$Q_IRP_FREE(R2),-     ; Is queue empty?
                00 B2         113E  3626                  @RCB$Q_IRP_FREE(R2)      ;
                     02   13    1140  3627          BEQL    60$                      ; If EQL, its empty
                     50   96    1142  3628          INCB    R0                       ; Indicate non-empty
                                1144  3629                                           ;
                     53 8ED0    1144  3630 60$:     POPL    R3                       ; Restore reg
                        05      1147  3631          RSB
                                1148  3632
```

```
                                                D 8
                                1148  3634          .SBTTL  TR$ALLOC_IRP    - Allocate IRP
                                1148  3635
                                1148  3636 ;+
                                1148  3637 ;  TR$ALLOC_IRP - Allocate IRP
                                1148  3638 ;
                                1148  3639 ;
                                1148  3640 ;  An IRP is allocated and its header is initialized.
                                1148  3641 ;
                                1148  3642 ;
                                1148  3643 ;  INPUTS:       None
                                1148  3644 ;
                                1148  3645 ;  OUTPUTS:      R3      IRP pointer if successful
                                1148  3646 ;                 R0      Status code
                                1148  3647 ;
                                1148  3648 ;                 All other registers are preserved
                                1148  3649 ;-
                                1148  3650 TR$ALLOC_IRP:                              ; Allocate Transport IRP
                         7E  51 7D      1148  3651          MOVQ    R1,-(SP)                ; Save regs
                       51  C4 8F 9A     114B  3652          MOVZBL  #IRP$C_LENGTH,R1        ; Setup IRP size
                  00000000'GF 16        114F  3653          JSB     G^EXE$ALONONPAGED       ; Get the block
                        1B 50 E9        1155  3654          BLBC    R0,10$                  ; Br on error
                        53 52 D0        1158  3655          MOVL    R2,R3                   ; Copy block address
                                        115B  3656          ;
                                        115B  3657          ;
                                        115B  3658          ;&  zero the entire IRP for now to catch access violations
                                        115B  3659          ;&  eventually, only the IRP$L_IOSB field (buffer ptr) will
                                        115B  3660          ;&  need to be zeroed
                                        115B  3661          ;
                                        115B  3662          ;
                         3F BB          115B  3663          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
63  00C4 8F 00 6E 00 2C                 115D  3664          MOVC5   #0,(SP),#0,#IRP$C_LENGTH,(R3)
                         3F BA          1165  3665          POPR    #^M<R0,R1,R2,R3,R4,R5>
                                        1167  3666
                         52 08 C0       1167  3667          ADDL    #IRP$W_SIZE,R2          ; Advance to size field
                                        116A  3668
                                        116A  3669          ASSUME  IRP$B_TYPE  EQ  2+IRP$W_SIZE
                                        116A  3670          ASSUME  IRP$B_RMOD  EQ  1+IRP$B_TYPE
                                        116A  3671
82  51 B0                               116A  3672          MOVW    R1,(R2)+                ; Enter size for deallocation
82  0A 90                               116D  3673          MOVB    S^#DYN$C_IRP,(R2)+      ; Enter buffer type
62  08 90                               1170  3674          MOVB    #NET$C_IPL,(R2)         ; Enter driver IPL
51  8E 7D                               1173  3675 10$:     MOVQ    (SP)+,R1                ; Restore regs
                         05             1176  3676          RSB                             ; Return
                                        1177  3677
```

```
                        1177  3679          .SBTTL  TR$ALLOCATE     - Allocate and initialize buffer
                        1177  3680 ;+
                        1177  3681 ;  TR$ALLOCATE  - Allocate and initialize buffer
                        1177  3682 ;
                        1177  3683 ;
                        1177  3684 ;  A buffer is allocated and initialized
                        1177  3685 ;
                        1177  3686 ;
                        1177  3687 ;  INPUTS:      R1      Size of buffer
                        1177  3688 ;
                        1177  3689 ;  OUTPUTS:     R2      Ptr to buffer if successful
                        1177  3690 ;                R1      Garbage
                        1177  3691 ;                R0      Status
                        1177  3692 ;-
                        1177  3693 TR$ALLOCATE:                                ; Allocate memory block
             53    DD   1177  3694          PUSHL   R3                        ; Save reg
                        1179  3695                                            ;
00000000'GF    16   E9   1179  3696          JSB     G^EXE$ALONONPAGED         ; Get buffer
          08   50   E9   117F  3697          BLBC    R0,50$                    ; Br on error
   08 A2       51   B0   1182  3698          MOVW    R1,FKB$W_SIZE(R2)         ; Setup size
               1B   90   1186  3699          MOVB    S^#DYN$C_CXB,-            ; Setup type
         0A A2           1188  3700                  FKB$B_TYPE(R2)            ;
                        118A  3701                                            ;
             53 8ED0   118A  3702 50$:     POPL    R3                        ; Restore reg
                  05   118D  3703          RSB
                        118E  3704
```

```
                                118E  3706          .SBTTL  TR_FILL_JNX     - Conditionally fill journal record.
                                118E  3707
                                118E  3708          .IF DF JNX$$$
                                118E  3709 ;+
                                118E  3710 ;  TR_FILL_JNX - If journalling is enabled, fill journal record.
                                118E  3711 ;
                                118E  3712 ; Inputs:
                                118E  3713 ;
                                118E  3714 ;       R0 = Journal record type
                                118E  3715 ;       R1 = Address of message
                                118E  3716 ;       R2 = RCB address
                                118E  3717 ;       R7 = Size of message
                                118E  3718 ;       R8 = LPD address
                                118E  3719 ;
                                118E  3720 ; Outputs:
                                118E  3721 ;
                                118E  3722 ;       No registers are destroyed.
                                118E  3723 ;-
                                118E  3724
                  00000040      118E  3725          JNL_REC_SIZ = 64
                                118E  3726
                                118E  3727 TR_FILL_JNX:
                          55 DD 118E  3728          PUSHL   R5                              ; Save reg
                    55 18 A2 D0 1190  3729          MOVL    RCB$L_PTR_JNX(R2),R5            ; Get the journal buffer
                          0A 13 1194  3730          BEQL    100$                            ; If EQL then no buffer
                06 A5 0040 8F B1 1196  3731          CMPW    #JNL_REC_SIZ,6(R5)              ; Enough space left?
                          02 1E 119C  3732          BGEQU   100$                            ; If GEQU then yes
                          04 10 119E  3733          BSBB    200$                            ; Record data
                       55 8ED0 11A0  3734 100$:     POPL    R5                              ; Restore reg
                          05 11A3  3735          RSB
                             11A4  3736
                       1F BB 11A4  3737 200$:     PUSHR   #^M<R0,R1,R2,R3,R4>             ; Save regs
                06 A5 0040 8F A2 11A6  3738          SUBW    #JNL_REC_SIZ,6(R5)              ; Acquire space to be used
                       54 65 D0 11AC  3739          MOVL    (R5),R4                         ; Get output pointer
             65 00000040 8F C0 11AF  3740          ADDL    #JNL_REC_SIZ,(R5)               ; Bump output pointer
             84 00000000'GF 7D 11B6  3741          MOVQ    G^EXE$GQ_SYSTIME,(R4)+          ; Enter timestamp
                       84 50 90 11BD  3742          MOVB    R0,(R4)+                        ; Enter record type
                    84 20 A8 90 11C0  3743          MOVB    LPD$B_PTH_INX(R8),(R4)+         ; Enter line i.d.
                       84 57 B0 11C4  3744          MOVW    R7,(R4)+                        ; Enter total message size
                       61 57 2C 11C7  3745          MOVC5   R7,(R1),-                       ;
                 64 34 00       11CA  3746                  #0,#JNL_REC_SIZ-12,(R4)         ; Enter begining of message
                       1F BA 11CD  3747          POPR    #^M<R0,R1,R2,R3,R4>             ; Restore regs
                          05 11CF  3748          RSB                                     ; Return to caller
                             11D0  3749
                             11D0  3750          .ENDC
                             11D0  3751
                             11D0  3752
                             11D0  3753
                  00000000   0000  3754          .PSECT  $$$116_DRIVER,LONG,EXE,RD,WRT   ; Make sure we're at the end
                             0000  3755                                                  ; of the driver
                             0000  3756
                             0000  3757
                             0000  3758 NET$END::
                          00 0000  3759          HALT
                             0001  3760
                             0001  3761
                             0001  3762 .END
```

```
$$_NSPMSG                       =  00000000
$$_TR3MSG                       =  00000000
$$_TR4MSG                       =  00000000
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
ADJ$B_LPD_INX                   =  00000002
ADJ$B_PTYPE                     =  00000001
ADJ$B_STS                       =  00000000
ADJ$C_PTY_AREA                  =  00000003
ADJ$C_PTY_PH2                   =  00000002
ADJ$C_PTY_PH3                   =  00000000
ADJ$C_PTY_PH3N                  =  00000001
ADJ$C_PTY_PH4                   =  00000004
ADJ$C_PTY_PH4N                  =  00000005
ADJ$V_LSN                       =  00000003
ADJ$V_RUN                       =  00000001
ADJ$W_BUFSIZ                    =  00000006
ADJ$W_INT_LSN                   =  00000008
ADJ$W_LPD                       =  00000002
ADJ$W_PNA                       =  00000004
ADJ$W_TIM_LSN                   =  0000000A
ADJ_UP                             000009B0 R      02
AGED                               00000AD7 R      02
AQB$L_ACPPID                    =  0000000C
AQB$L_ACPQBL                    =  00000004
BUG$_NETNOSTATE                    ********   X   02
CA$_MEASURE                     =  00000002
CHECK_RQR                          00000BB3 R      02
CNF$_ADVANCE                    =  00000000
CNF$_QUIT                       =  00000002
CNF$_TAKE_CURR                  =  00000003
CNF$_TAKE_PREV                  =  00000001
COM$DRVDEALMEM                     ********   X   02
CRC16                              00000000 R      02
CXB$B_R_AREA                       00000039
CXB$B_R_FLG                        00000038
CXB$B_R_NSPTYP                     00000039
CXB$B_TYPE                      =  0000000A
CXB$B_X_NSPTYP                     0000004E
CXB$C_DLL                       =  00000020
CXB$C_HEADER                    =  00000048
CXB$C_OVERHEAD                  =  0000004C
CXB$C_R_LENGTH                  =  0000003C
CXB$L_R_MSG                        0000002C
CXB$L_R_RCB                        00000028
CXB$T_DLL                       =  00000028
CXB$T_X_DATA                       00000057
CXB$T_X_XPORT                      00000048
CXB$W_R_ADJ                        0000003A
CXB$W_R_BCNT                       00000030
CXB$W_R_DSTNOD                     00000034
CXB$W_R_NSPSEQ                     0000003A
CXB$W_R_PATH                       00000032
CXB$W_R_SRCNOD                     00000036
CXB$W_X_NSPACK                     00000053
CXB$W_X_NSPLOC                     00000051
CXB$W_X_NSPREM                     0000004F
CXB$W_X_NSPSEQ                     00000055
DISP_RCV_MSG                       00000869 R      02
DYN$C_CXB                       =  0000001B
DYN$C_IRP                       =  0000000A
DYN$C_NET                       =  00000017
EXE$ALONONPAGED                    ********     X   02
EXE$ALTQUEPKT                      ********     X   02
EXE$DEANONPAGED                    ********     X   02
EXE$FORK                           ********     X   02
EXE$GL_ABSTIM                      ********     X   02
EXE$GQ_SYSTIME                     ********     X   02
EXIT                               00000348 R      02
FINISH_XMT_HDR                     00000D11 R      02
FKB$B_FIPL                      =  0000000B
FKB$B_TYPE                      =  0000000A
FKB$C_LENGTH                    =  00000018
FKB$L_FPC                       =  0000000C
FKB$L_FR3                       =  00000010
FKB$L_FR4                       =  00000014
FKB$W_SIZE                      =  00000008
GET_OUT_ADJ                        00000094 R      02
HELLO_MSG_SIZE                  =  00000024
INIT_CXB_FREE                      000001AA R      02
INIT_RCV                           000000DD R      02
IO$_READLBLK                       ********     X   02
IO$_WRITELBLK                      ********     X   02
IOC$GL_PSBL                        ********     X   02
IPL$_IOPOST                     =  00000004
IPL$_QUEUEAST                   =  00000006
IPL$_SYNCH                      =  00000008
IRP$B_EFN                       =  00000022
IRP$B_PRI                       =  00000023
IRP$B_RMOD                      =  0000000B
IRP$B_TYPE                      =  0000000A
IRP$C_LENGTH                    =  000000C4
IRP$L_AST                       =  00000010
IRP$L_ASTPRM                    =  00000014
IRP$L_BCNT                      =  00000032
IRP$L_IOSB                      =  00000024
IRP$L_IOST1                     =  00000038
IRP$L_IOST2                     =  0000003C
IRP$L_PID                       =  0000000C
IRP$L_SAVD_RTN                  =  00000078
IRP$L_SVAPTE                    =  0000002C
IRP$L_UCB                       =  0000001C
IRP$L_WIND                      =  00000018
IRP$M_BUFIO                     =  00000001
IRP$M_FUNC                      =  00000002
IRP$Q_NT_PRVMSK                 =  00000040
IRP$Q_STATION                   =  00000040
IRP$W_BCNT                      =  00000032
IRP$W_BOFF                      =  00000030
IRP$W_CHAN                      =  00000028
```

```
IRP$W_FUNC                      =  00000020
IRP$W_SIZE                      =  00000008
IRP$W_STS                       =  0000002A
JNL_REC_SIZ                     =  00000040
JNX$$$                          =  00000001
LISTENER                           00000322 R      02
LPD$B_BCPRI                     =  0000002A
LPD$B_ETY                       =  0000001D
LPD$B_IRPCNT                    =  0000001C
LPD$B_PTH_INX                   =  00000020
LPD$B_XMT_IPL                   =  0000001F
LPD$B_XMT_SRL                   =  0000001E
LPD$C_LOC_INX                   =  00000001
LPD$L_CACHE                     =  00000066
LPD$L_CNT_APR                   =  0000003A
LPD$L_CNT_DPS                   =  00000042
LPD$L_CNT_TPR                   =  0000003E
LPD$L_CNT_TPS                   =  00000046
LPD$L_RCV_IRP                   =  00000032
LPD$L_RTR_LIST                  =  0000002E
LPD$L_UCB                       =  00000010
LPD$L_WIND                      =  0000000C
LPD$M_ACTIVE                    =  00000001
LPD$M_XEND                      =  00000800
LPD$Q_REQ_WAIT                  =  00000000
LPD$V_ACTIVE                    =  00000000
LPD$V_ALIGNQ                    =  0000000E
LPD$V_ALIGNW                    =  0000000D
LPD$V_BC                        =  0000000A
LPD$V_RBF                       =  00000006
LPD$V_RUN                       =  00000004
LPD$V_X25                       =  00000007
LPD$V_XBF                       =  00000005
LPD$V_XEND                      =  0000000B
LPD$W_BUFSIZ                    =  00000050
LPD$W_CHAN                      =  00000014
LPD$W_CNT_TCL                   =  0000004C
LPD$W_DRT                       =  0000002C
LPD$W_INT_TLK                   =  00000018
LPD$W_PTH                       =  00000020
LPD$W_STS                       =  00000022
LPD$W_TIM_TLK                   =  00000016
MAX_NODES                       =  00000400
MMG$GL_SPTBASE                     ********    X   02
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$END                            00000000 RG     03
NET$M_MAXLNKMSK                 =  000003FF
NET$UNSOL_INTR                     ********    X   02
NETMSG$C_ADJ                    =  0000000C
NETMSG$C_APL                    =  00000005
NETMSG$C_CRD                    =  0000000B
NETMSG$C_IRP                    =  00000004
NETMSG$C_LSN                    =  00000009
NETMSG$C_NOL                    =  00000007
NETMSG$C_NUL                    =  00000006
NETMSG$C_OPL                    =  0000000A
NETMSG$C_PFE                    =  00000008
NETMSG$C_UNK                    =  00000001
NETUPD$_DLL_ON                  =  00000005
NETUPD$_GET_ADJ                 =  0000000E
NETUPD$_REACT_RCV               =  0000000C
NETUPD$_SEND_HELLO              =  0000000D
NETUPD$_TEST_ADJ                =  0000000F
NODES_PER_PASS                  =  00000100
NODE_SHIFT                      =  00000008
NOT_REACH                          0000070B 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$C_MAX_DELAY                 =  00000014
NSP$C_MAX_R_CXB                 =  00000007
NSP$C_MAX_XPW                   =  00000007
NSP$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
OPL                               00000AC7 R      02
PFE                               00000AB7 R      02
PFE_BR                            000009B6 R      02
PMS$GL_ARRLOCPK                   ********     X  02
PMS$GL_ARRTRAPK                   ********     X  02
PMS$GL_DEPLOCPK                   ********     X  02
PMS$GL_RCVBUFFL                   ********     X  02
PMS$GL_TRCNGLOS                   ********     X  02
POST                              000010EE R      02
PR$_IPL                           ********     X  02
PR$_SIRR                          ********     X  02
QUICK_SOL                         000004FD R      02
RANGE                             00000AF7 R      02
RCB$B_ACT_TIMER                 =  0000008F
RCB$B_AQB_CNT                   =  000000A9
RCB$B_CNT_APL                   =  00000095
RCB$B_CNT_NOL                   =  00000094
RCB$B_CNT_OPL                   =  00000096
RCB$B_CNT_PFE                   =  00000097
RCB$B_ETY                       =  0000008A
RCB$B_HOMEAREA                  =  0000008B
RCB$B_LSN_ADJ                   =  000000A8
```

```
RCB$B_MAX_AREA          =  0000008C
RCB$B_MAX_LPD           =  0000005C
RCB$B_MAX_VISIT         =  0000005E
RCB$B_STATUS            =  0000000B
RCB$B_STI               =  00000061
RCB$L_AQB               =  00000010
RCB$L_PTR_ADJ           =  0000002C
RCB$L_PTR_AOA           =  00000020
RCB$L_PTR_JNX           =  00000018
RCB$L_PTR_LPD           =  00000028
RCB$L_PTR_OA            =  0000001C
RCB$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$V_ACT               =  00000001
RCB$V_LVL2              =  00000000
RCB$W_ADDR              =  0000000E
RCB$W_ALIAS             =  0000008D
RCB$W_CNT_NUL           =  0000009A
RCB$W_CUR_PKT           =  00000080
RCB$W_DRT               =  000000AA
RCB$W_LVL2              =  000000AC
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_TOTBUFSIZ         =  0000007E
RCB$W_TRANS             =  0000000C
RCV_DIO_BIO               00000822 R      02
REACH                     00000AE7 R      02
REACT_RCV                 000000BA R      02
RETRY_TIMER             =  00000004
ROUTE                     00000BBF R      02
SCAN_CACHE                00000710 R      02
SCH$WAKE                  ********   X    02
SEND_HELLO                000000A0 R      02
SIZ...                  =  00000001
SOL_AREA                  000006CD R      02
SOL_NW                    0000049B R      02
SOL_PH4                   00000677 R      02
SOL_PH4N                  0000069C R      02
SOL_WAIT                  000004C6 R      02
SS$_DEVACTIVE             ********   X    02
SS$_NORMAL                ********   X    02
TALKER                    00000349 R      02
TEMP                    =  00000001
TEST_ADJ                  0000005D R      02
TO_ACP                    00000B0A R      02
TR$ADJUST_IRP             00001106 R      02
TR$ALLOCATE               00001177 R      02
TR$ALLOC_IRP              00001148 R      02
TR$C_MAXADR             =  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
TR$DENY                   0000052A R      02
TR$GET_ADJ                00000599 RG     02
TR$GIVE_TO_ACP            00001043 RG     02
TR$GRANT                  00000535 R      02
TR$KILL_LOC_LPD           000001D6 RG     02
TR$LOC_DLL_RCV            000010B3 R      02
TR$LOC_DLL_XMT            000010A1 R      02
TR$QUE_IRP_AQB            00001048 R      02
TR$QUE_WQE_AQB            0000102E R      02
TR$RCV_BIO_DATA           000007CF RG     02
TR$RCV_DIO_DATA           0000072B RG     02
TR$RTRN_XMT_ECL           00000F06 RG     02
TR$RTRN_XMT_RTH           00000EE0 RG     02
TR$RTRN_XMT_TLK           00000E9A RG     02
TR$SOLICIT                0000048C RG     02
TR$TEST_REACH             00000589 RG     02
TR$TIMER                  00000217 RG     02
TR$UPDATE                 00000040 RG     02
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
TR_ECL                            00000A89 R      02
TR_FILL_JNX                       0000118E R      02
TR_LPD_DOWN                       00000FD7 R      02
TR_RTHDR                          000009B9 R      02
TR_RTHRU                          00000B28 R      02
TR_RTRN_IRP                       00000F5E R      02
UNR                               00000B07 R      02
UPDATE_CACHE                      00000E4D R      02
VA$M_BYTE                       = 000001FF
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
WQE$B_EVT                       = 00000010
WQE$B_TYPE                      = 0000000A
WQE$C_LENGTH                    = 00000024
WQE$L_PM2                       = 00000014
WQE$W_ADJ_INX                   = 00000020
WQE$W_REQIDT                    = 00000012
XFER                              000010C5 R      02
XM$M_STS_ACTIVE                 = 00000800
XM$M_STS_BUFFAIL                = 00001000
XM$V_STS_BUFFAIL                = 0000000C
XPT_C_CACHETIMEOUT              = 00000046
XPT_C_CACHETIMER                = 0000000A
_$$_                            = 00000000
```

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	00000057 (87.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	000011D0 (4560.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$116_DRIVER	00000001 (1.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.09	00:00:00.85
Command processing	177	00:00:01.10	00:00:07.08
Pass 1	531	00:00:23.18	00:00:47.01
Symbol table sort	0	00:00:02.07	00:00:04.00
Pass 2	519	00:00:08.61	00:00:20.97
Symbol table output	5	00:00:00.37	00:00:00.72
Psect synopsis output	3	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1274	00:00:35.45	00:01:20.66

The working set limit was 900 pages.
130558 bytes (255 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1167 non-local and 270 local symbols.
3762 source lines were read in Pass 1, producing 29 object records in Pass 2.
53 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	2
_255DUA28:[NETACP.OBJ]NET.MLB;1	11
_255DUA28:[SYS.OBJ]LIB.MLB;1	12
_255DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	32

1360 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

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

NETDRVXPT
LIS

NETOPCOM
LIS

NETLLCNT
LIS

NETEVTLOG
LIS

NETPROCRE
LIS